Министерство образования и науки Российской Федерации
Государственное образовательное учреждение
высшего профессионального образования
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЛЕСОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра Информационных систем и технологий

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ: ТЕОРИЯ И ПРАКТИКА

С б о р н и к н а у ч н ы х т р у д о в

Выпуск 6

Санкт-Петербург
2014

Рассмотрен и рекомендован к изданию
Ученым советом лесотехнического факультета
Санкт-Петербургского государственного лесотехнического университета
имени С.М.Кирова
«_____»_____2014 г.

С о с т а в и т е л ь

М.А. Шубина, кандидат технических наук, доцент (отв. секретарь)

УДК 630

Информационные системы и технологии: теория и практика: сб.
Научн. Тр. Вып.6./отв. ред. А.М.Заяц.-СПб.: СПБГЛТУ, 2014. – 85 с.

ISBN 978-5-9239-0163-4

Сборник подготовлен по материалам кафедры вуза, представленным на научно-технической конференции лесохозяйственного факультета СПбГЛТУ в январе 2014 года, и практических работ, выполненных ее сотрудниками

Темплан 2014 г. Изд.№
ISBN 978-5-9239-0163-4

А.М. Заяц, кандидат технических наук, профессор
В.А. Пресняков, кандидат технических наук, доцент

ВИРТУАЛЬНАЯ СРЕДА В ИНФРАСТРУКТУРЕ КАФЕДРЫ

Вся почти пятидесятилетняя история кафедры связана не только с реализацией учебных планов по подготовке студентов в области вычислительной техники, информационных систем и информационных технологий, но и активным внедрением инфокоммуникационных технологий (ИКТ) в учебный процесс.

Опыт использования компьютерной техники на кафедре позволяет резюмировать следующее:

- парк компьютерной техники большой, но состоит из устаревших и различных по своим возможностям образцов с малым числом необходимого современного периферийного оборудования и мобильных устройств;
- сетевая инфраструктура базируется на старых технологиях – конечные пользователи «толстые» клиенты;
- гетерогенность не только аппаратного, но и программного обеспечения;
- отсутствие новых аппаратно-программных и информационно – образовательных сред для изучения и освоения новых быстроразвивающихся систем и информационных технологий;
- завершение технической поддержки ОС Windows XP с апреля 2014 года, составляющей основу системного ПО кафедры;
- необходимость интеграции средств и технологий автоматизации учебного процесса и деятельности вуза с новыми информационно-образовательными технологиями и программными продуктами.

Существующее положение свидетельствует о необходимости кардинального изменения в развитии и модернизации информационных систем и технологий, используемых в вузах.

На изменение подходов в информатизации образовательного процесса оказывает влияние и то, что во - первых в настоящее время в вузах обучаются студенты, повседневная жизнь которых уже не мыслима без использования различных гаджетов, как принято сейчас называть технические приспособления, обладающие инфокоммуникационными воз-

можностями, во – вторых выросло новое, так называемое Y - поколение не представляющее повседневность без персональных компьютеров, интернета, социальных сетей, различных мобильных устройств и приложений.

В вузах формируются коллективы преподавателей выросших и олучивших образование в годы интенсивного внедрения ИКТ в учебный процесс, информатизации общества и всех сфер человеческой деятельности в целом.

На смену «старой гвардии» приходят преподаватели нового поколения − преподаватели, самостоятельно занимающиеся разработкой необходимых им электронных информационных ресурсов, что предполагает приближение их к уровню подготовки квалифицированных пользователей или даже программистов. И этот процесс не дань моде, а движение в этом направлении уже не удел одиночек - преподавателей, а сформировавшаяся тенденция.

При этом нормативная и методическая документация «заточены» под старые образовательные формы, использование новых электронных образовательных технологий почти не учитывается в планировании учебным процессом и зачастую только декларируется в ФГОСах. Разрабатываемые учебно – методические пособия по форме представления материала ориентированы на старые – «доэлектронные» требования.

Модернизация аппаратно – программного обеспечения систем обычно предполагает его обновление, однако практика показывает, что введение в структуру ИС новых более совершенных и мощных компьютеров и серверов не всегда означает их более эффективную работу в решении существующих и новых задач в системе.

Дело в том, что обычно каждая машина используется для выполнения только одного приложения в конкретный момент времени. Исследования показывают, что зачастую серверы и рабочие станции в системе используются только на 15-20% от их реальных возможностей.

Для устранения этого с целью повышения эффективности использования компьютерной техники при выполнении сложных приложений приобретается дорогие и мощные компьютеры.

Но каждый из них, если не принимать специальных решений не может одновременно выполнять сразу нескольких приложений и реализовывать их в разных операционных средах. При таком положении приходится покупать больше аппаратов, расходы растут и теряется эффект от использования современного программного обеспечения, и смысл модернизации компьютерной техники.

Разрешение этой парадоксальной ситуации обеспечивает виртуализация ресурсов ИТ на основе развертывания нескольких полноценных

виртуальных машин на одной аппаратной, работающих одновременно как на одной, так и на различных операционных системах.

Процессорные характеристики, организация и объемы памяти современных компьютеров позволяют разместить на одном физическом сервере несколько виртуальных, установить на каждый необходимую операционную систему и запускать одновременно приложения с каждого из них. Таким образом, решается важная проблема использования мощности современного оборудования в полном объеме без лишних затрат.

При этом упрощается процедуры и снижаются затраты на администрирование информационных систем реализующих принципы виртуализации. Одним из системных достоинств использования виртуальных машин (ВМ) является возможность их объединения в сети, что позволяет на одном компьютере моделировать поведение распределенных систем, состоящих как из приложений для конечного пользователя, так и различного рода серверов в гетерогенной среде.

Гибкость виртуальных машин в отношении выделяемых им ресурсов, широкие возможности по оптимизации их производительности позволяет легко управлять множеством разных конфигураций ВМ и создавать независимые от оборудования приложения, «упакованные» в виртуальные машины. Затем эти компоненты, состоящие из виртуальных машин, могут быть в различных вариантах объединены в сеть для моделирования различных систем.

Создание виртуальной инфраструктуры должно осуществляться в рамках единого проекта с учетом и использованием тех наработок, которые существуют и учитывают специфику конкретного ВУЗа.

Опыт кафедры позволяет решать эти задачи на ее информационно – коммуникационных средствах использую существующую информационную систему (не нарушая процедуры ее использования в существующем образовательном процессе) как полигон создания и тестирования новых информационных структур и технологий во всем спектре задач информатизации вуза с дальнейшим переносом всего положительного в инфраструктуру университета.

Эта идея может и должна проводиться на основе реальной реструктуризации программно-технической базы кафедры информационных систем и технологий включающая:

- постепенный переход на новую организацию учебного процесса для подготовки специалистов в современной информационно-образовательной среде на платформе виртуализации и облачных технологий с переводом всех ресурсов на кластер (ферму) серверов;
- на начальном этапе выделение одного физического сервера под контроллер домена, в который вводятся все необходимые серверы и

рабочие станции: Web, ftp, DNS, SQL; сервер приложений; хранилище данных; рабочие станции преподавателей; рабочие станции студентов; принтеры и другое оборудование. Это позволит централизованно определять политику безопасности и распределения ресурсами;

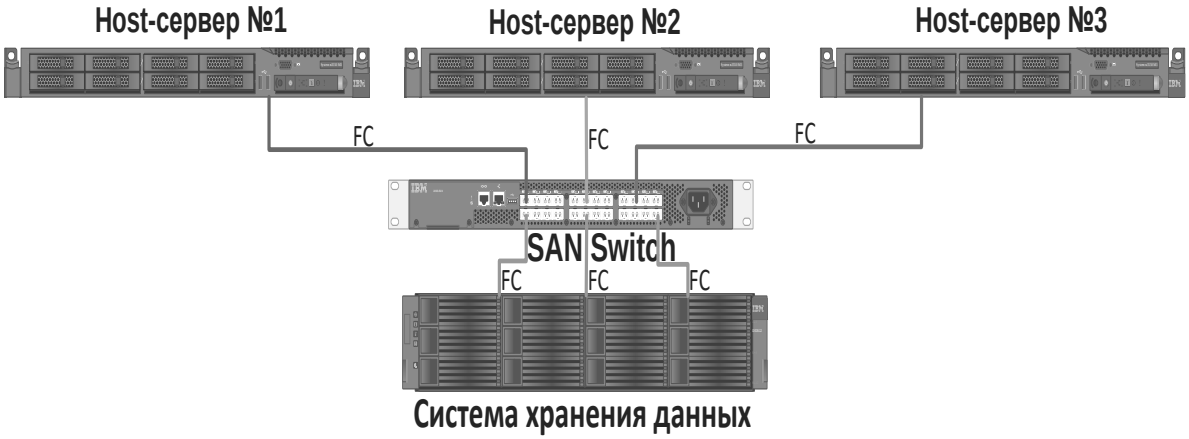
- обеспечение индивидуальным информационно – сервисным пространством (ИИСП) каждого студента и преподавателя в рамках виртуальных машин (ВМ) с централизованным доступом к ресурсам и управления ими;
- усовершенствование структурированной кабельной системы (СКС) информационной системы кафедры;
- создание модели (стенда) виртуальной лаборатории и электронного обучения;
- создание модели (стенда) организации учебного процесса и документооборота всего ВУЗа на базе ИС кафедры.

При этом парк существующих ПК лишь минимально может быть дополнен функциональными компьютерами.

Дополнительно приобретается всего лишь один новый компьютер для развертывания платформы виртуализации. В качестве ОС на него устанавливается один из гипервизоров выполняющий функцию управления созданием и работой виртуальных машин.

В дальнейшем возможным расширением проектного варианта может быть следующая структура, предложенная специалистами фирмы РАМЕК (http://www.ramec.ru/).

Три хост сервера объедены через Fibre Channel к системе хранения данных (СХД) по топологии сети - однокоммутаторная структура. Все элементы архитектуры подключаются через SAN Switch.



Компоненты

Лицензии VMware:

- Лицензии vSphere 5.5
- Сетевой коммутатор (на 40 конкурентных лицензий)

Аппаратная часть на базе IBM:

- Серверы (3 шт.) Intel Xeon 10C Processor, 256 Gb.
- Система хранения IBM System Storage DS3512, 36 TB.
- Оптический коммутатор.
- Сетевой коммутатор.

Существующие ПК:

- FDDI или CDDI топология ЛВС;
- 3 сервера, 1 СХД с платформой виртуализации.

В качестве терминалов доступа используются существующие компьютеры, на которых в идеале удаляется все ПО, «урезается» до минимума ОС и фактически нужен всего ли один сервис - это терминальная связь, фактических протокол RDP. Более того, для всех задач или для части задач, к виртуальным рабочим столам можно подключаться с ноутбуков или планшетов студентов и преподавателей.

При этом загрузка сети остается минимальной, так все вычисления, загрузка программ происходят на основном физическом сервере, на котором работают виртуальные машины с серверными и другими приложениями и даже результаты тяжелых запросов после формирования их, например, на стороне виртуальной машины с SQL сервером, возвращаются опять же на виртуальную рабочую станцию. Таким образом, на существующий компьютер, ноутбук или планшет идет только «перерисовка» экрана.

Наряду с отмеченными преимуществами при такой структуре можно сократить число учебно – вспомогательного персонала (УВП), т.к. с технологией виртуализации на всех учебных рабочих станциях может стоять один образ ОС с клиентом доступа к серверу – такой образ делается один раз, а дальше просто копируется, при этом не требуется нахождение рядом с физическим дисплейным классом системного администратора высокого уровня квалификации.

На виртуальной машине студенты могут свободно устанавливать и удалять операционные системы, программное обеспечение, изменять настройки системы и выполнять любые дисковые операции без нарушения работоспособности «реального» компьютера. Это очень важно, поскольку количество, а иногда и квалификация УВП не позволяет восстанавливать операционные системы реальных компьютеров до исходного состояния после каждого занятия.

На виртуальной машине обучаемый обладает правами администратора, может настраивать виртуальную компьютерную сеть и в

то же время не имеет возможности изменить настройки, реально существующей компьютерной сети или получить несанкционированный доступ к закрытым ресурсам.

Каждый обучаемый будет иметь «собственную» виртуальную машину с индивидуальными настройками и необходимым программным обеспечением (обучаемые имеют возможность даже переносить такие виртуальные машины с одного компьютера на другой, например, из компьютерного класса на домашний компьютер по сети или на перезаписываемых DVD-дисках или flash-накопителях).

Это позволяет в значительной степени «индивидуализировать» учебный процесс и снять распространенные проблемы, возникающие при обучении нескольких групп в одном компьютерном классе, когда одни обучаемые случайно или умышленно модифицируют или удаляют файлы данных настройки других.

Технология виртуализации также позволяет минимизировать негативный эффект от ОС, находящихся в работе длительное время без должного обслуживания. В такой ОС могут наблюдаться проблемы фрагментации файлов, накопление большого объема журналов (в т.ч. системных), обновлений, временных файлов и т.п., что в совокупности приводит к существенному увеличению времени отклика в системе.

В сфере изучения информационных технологий виртуальные машины можно использовать: для установки программ, несовместимых с ОС реального компьютера; защиты информации; тестирования программного и/или аппаратного обеспечения; создания переносных пользовательских сред, «отвязанных» от конкретного оборудования; запуска вредоносных программ с целью их исследования; эмуляции локальной компьютерной сети.

Работа, провидимая на кафедре, уже сегодня позволяет активно использовать созданные преподавателями электронные информационные ресурсы на платформе виртуализации и облачных технологий в образовательном процессе.

Н.П. Васильев, кандидат технических наук, доцент
Хмарик А.Г., Сластунов Д.Д.

SENCHA EXTJS НА ПРИМЕРЕ РАЗРАБОТКИ ОПРЕДЕЛИТЕЛЯ РАСТЕНИЙ

Почему Sencha ExtJS и что это такое?

В настоящее время, когда HTML-5 и современные WEB технологии поддерживаются практически любым WEB-обозревателем, последние превратились в первоклассную платформу для разработки и реализации приложений. Вот лишь несколько доводов в пользу высказанного положения:

- Межплатформенность. Разработанное на основе указанных технологий приложение будет одинаково хорошо работать на клиентском компьютере с операционной системой от Windows и до iOS X.
- Не требуется установка клиентской части разработанного приложения.
- Современные Ajax фреймворки позволяют создавать насыщенные функциональностью интерфейсы, которые ранее были доступны только в настольных приложениях, разрабатываемых в популярных средах программирования типа Delphi, Builder, Visual C++ и т. д. Эти приложения по этой причине иногда называют RIA – Rich Internet Application. Среди фреймворков лидирующее положение занимает Sencha ExtJS [1]. Однако, он не столь популярен в России, как, например, jQuery (хотя в последнее время все большее количество разработчиков обращается именно к этой библиотеке, наверное, узнав о ее возможностях).
- Использование Ajax библиотек гарантирует поддержку ведущих браузеров: разработка будет одинаково хорошо функционировать и выглядеть в любом браузере, будь то Internet Explorer, Safari или Opera.
- Что касается технологий Sencha, то кроме ExtJS, существует Sencha Touch для мобильных устройств. Эта технология по сути та же и может быть легко освоена разработчиком, как продолжение ExtJS.
- Наконец, все, что нельзя реализовать в рамках клиента, можно запросить у сервера, или серверов, используя Ajax.
- Технология Adobe® AIR® позволяет разработчикам получить из WEB разработки «родное» приложение для Windows, Mac OS, а также iPhone, iPad, Kindle Fire, Nook Tablet, и других устройств на платформе Android™.

Реализация определителя растений. Описанная технология была опробована на примере программной реализации определителя хвойных растений. Основные идеи многовходового определителя изложены авторами в работе [2]. Реализацию можно посмотреть здесь [3].

Требовалось разработать гибкий и компактный интерфейс для выбора значений ряда признаков, по которым и должна была производиться фильтрация растений. Авторами работы [2] накоплен и систематизирован достаточно объемный материал, касающийся отличительных признаков хвойных северо-запада России с большим количеством фотографий и иных иллюстраций. Эти данные существуют в виде электронных таблиц и продолжают корректироваться и накапливаться авторами по настоящее время, с помощью вспомогательных программных средств, разработанных на Visual Basic.

Данные были проанализированы, структурированы и конвертированы в базу данных под управлением SQL сервера. Таким образом, была налажена технология периодической корректировки и дополнения информационной основы определителя, которая составляет серверную часть приложения. Признаки могут иметь ряд фиксированных значений, или могут быть числовыми. Признаки объединены в группы. Растения связаны со значениями признаков. При выборе нескольких признаков - искомое множество суть пересечение множеств растений с заданными одиночными признаками. Авторами работы [2] был предложен оригинальный алгоритм оценки «ошибочности» ввода значений признаков пользователем, который, однако, связан с емкими вычислениями некоторых пересечений множеств таксонов, удовлетворяющих тем или иным признакам. За счет удачно разработанной SQL структуры данных эти вычисления производятся достаточно эффективно, что позволяет их использовать в режиме реального времени.

Следующая задача состояла в разработке компактного, с одной стороны, а с другой стороны, достаточно мощного и гибкого интерфейса для доступа к разработанному серверному программному обеспечению. Эта задача была удачно решена с помощью AJAX фреймворка ExtJS.

Пользователь имеет возможность выбрать произвольный признак из любой группы и получить справочную информацию по этому признаку на любой стадии определения. В результате, в интерактивном режиме пользователь получает отфильтрованный список таксонов. Оперативно рассчитанная «ошибочность» позволяет отсортировать список признаков и состояний признаков в соответствии с их диагностической ценностью.

Достоинством определителя, которое, несомненно, можно отнести к фреймворку ExtJS, является возможность детального сравнения нескольких рисунков или фотографий, иллюстрирующих таксоны,

отфильтрованные в ходе определения. Данный инструмент позволяет эффективно использовать богатый визуальный материал, накопленный авторами работы [2].

Таким образом, описанные клиент серверные технологии, в очередной раз показали свою мощь и эффективность в ходе разработки электронного определителя растений.

Библиографический список

1. http://www.sencha.com/
2. Хмарик А.Г. Орлова Л.В. Васильев Н.П. Сластунов Д.Д. Егоров А.А. Разработка электронного определителя хвойных Северо-Запада России // Труды XIII Съезда русского ботанического общества и конференции "Научные основы охраны и рационального использования растительного покрова Волжского бассейна". Т. 2 : Систематика и география сосудистых растений. Сравнительная флористика. Геоботаника. Тольятти: Кассандра, 2013. – С. 80-81.
3. http://95.31.134.154/DET/markList_tst.html

С.В.Гуров, доктор технических наук, профессор

ОЦЕНКА НАДЕЖНОСТИ ТЕХНИЧЕСКИХ СИСТЕМ НА ЭТАПЕ ИХ ЭКСПЛУАТАЦИИ

1. Эксплуатация системы и изменение закона распределения времени работы до отказа

Надежность является важнейшим параметром любой технической системы. Она во многом определяет такие характеристики системы, как качество, эффективность, безопасность, живучесть, риск. Основными понятиями теории надежности являются «надежность» и «отказ». Надежностью называется свойство технического объекта сохранять свои характеристики (параметры) в определенных пределах при данных условиях эксплуатации [1]. Из этого определения следует, что надежность технической системы тесно связана с условиями ее эксплуатации. Отказом называется событие, после возникновения которого характеристики технического объекта (параметры) выходят за допустимые

пределы. Это понятие субъективно, т. к. допуск на параметры объекта устанавливает пользователь. Отказ — фундаментальное понятие теории надежности. Критерий отказа — отличительный признак или совокупность признаков, согласно которым устанавливается факт возникновения отказа.

Одной из разновидностей модели надежности (или безопасности) технической системы можно считать зависимость ее характеристик от условий эксплуатации или испытываемых нагрузок. Например, наработка объекта до отказа (или между отказами) конкретной технической системы, очевидно, является случайной функцией случайной нагрузки. В еще большей степени это относится к функции распределения времени до отказа системы. Эксплуатация в технике — часть жизненного цикла изделия (технической системы, орудия труда, сооружения и т. п.), на протяжении которого оно используется по назначению [2]. На этапе эксплуатации реализуется, поддерживается и восстанавливается надежность изделия. В общем случае эксплуатация изделия включает в себя также транспортирование, хранение, техническое обслуживание и ремонт.

Условия эксплуатации технических систем имеют различную физико-химическую природу и изменяются в широких пределах. Их можно разделить на климатические, механические, радиационные. Климатические факторы обусловлены изменением температуры и влажности окружающей среды, тепловым ударом, увеличением или уменьшением атмосферного давления, наличием движущихся потоков пыли (песка), присутствием активных веществ в окружающей атмосфере, наличием солнечного облучения, взрывоопасной и воспламеняющейся атмосферой, дождя или брызг. Механические факторы – это воздействие вибрации, ударов, линейного ускорения, акустического удара; наличие невесомости. Радиационные факторы представляют собой космическую радиацию, ядерную радиацию от реакторов, атомных двигателей, облучение потоком гамма-фотонов, быстрыми нейтронами, бета-частицами, альфа-частицами, протонами. Характер и интенсивность воздействия перечисленных факторов зависят от условий использования и назначения технических средств, которые разделяются на стационарные и транспортируемые.

С течением времени в технических системах, как правило, происходят какие-либо изменения, влияющие на длительность их эксплуатации. Эти изменения могут носить как внутренний, так и внешний характер. Внешние изменения связаны, как правило, с изменением условий эксплуатации систем. На систему могут воздействовать изменение температуры, влажности, давления, вибрация, механические удары, электромагнитные помехи, агрессивная химическая среда и др. Работа

системы может происходить в разных условиях (на земле, воде, воздухе), что также приводит к изменению нагрузки на систему и, как следствие, изменению ее надежности. Если рассматриваемая система является элементом системы более высокого уровня, то изменения нагрузки могут быть еще обусловлены отказами (или восстановлением) некоторых других элементов. Данное свойство называется последействием отказов (восстановлений) резервированной системы. Наличие интервалов простоя в работе системы можно расценивать как воздействие на систему нулевой нагрузки.

Влияние внешних факторов во многом определяет возможности нормального функционирования технической системы. Так, повышение рабочей температуры снижает ее надёжность. Колебания температуры могут привести в механических узлах конструкции к изменению типа посадок, вызвать ослабление крепления, температурные напряжения. Воздействие низких температур ухудшает прочностные характеристики материалов, эластичность упругих элементов. Понижение атмосферного давления отрицательно влияет на условия теплоотвода в конструкциях системы, что связано со многими нарушениями нормального ее функционирования. Повышенная влажность может вызвать коррозию деталей и несущих конструкций, которой особо способствуют наличие активных веществ в атмосфере, солнечная радиация, пыль и песок.

Современные сложные системы часто работают в изменяющихся условиях: изменение условий внешней среды, сезонные изменения потока заявок на обслуживание технических средств и их составных частей, возникновение различных аномалий при эксплуатации, изменение экономической ситуации и т.д. Условия эксплуатации технических, и, в частности транспортных средств, представляют совокупность факторов, к которым относятся дорожные и транспортные условия, организация работы подвижного состава, природно-климатические условия, квалификация водителей и ремонтного персонала и другие. Особую роль играют, так называемые, «тяжёлые» условия эксплуатации: нерегулярное использование техники или значительные перерывы в ее эксплуатации; неодновременная работа составных элементов технических систем; непрогретые до рабочей температуры узлы технических средств; режим "включение - выключение", при котором достигается максимальная нагрузка на систему; повышенная нагрузка на элементы при отказе других элементов системы; эксплуатация технического средства в условиях запылённого или загрязнённого воздуха; ослабление, полное прекращение или усиление электропитания технического средства; низкое качество топлива и нерегулярная его замена.

Очевидно, что идеальными условиями эксплуатации транспорта являются регулярные дальние поездки в пустом автомобиле с умеренными скоростями по чистым Европейским дорогам, да ещё к тому же на европейском топливе. Для других условий эксплуатации требуется корректировка рекомендаций производителя автомобиля относительно того, как часто нужно менять моторное масло.

Весьма существенной причиной возникновения дополнительных нагрузок в системе является изменение надежности ее отдельных узлов в зависимости от состояния (рабочего или отказового) других частей системы. Эта зависимость может быть смоделирована с помощью так называемых моделей нагрузки на составляющие элементы. Важным моментом для таких моделей является правило, определяющее изменение интенсивности отказов элементов после отказов других элементов системы.

При изменении условий эксплуатации системы мы выходим за рамки классической теории надежности, предполагающей эти условия неизменными. Следствием может служить изменение закона распределения времени работы системы до отказа.

Формально на случайную величину X - время жизни технической системы в определенные моменты времени воздействует нагрузка $k(t)$, которая может быть как больше, так и меньше 1. Следует считать, что при стабильных условиях $k(t)=1$. Тогда в соответствии с изменяющейся нагрузкой происходит изменение закона распределения времени до отказа системы. В частности, время до отказа (продолжительность жизни) может уменьшиться, или увеличиться.

Примерами возникновения и изменения нагрузки на систему могут служить:

- Распределение нагрузки на системы гражданского строительства, в частности, влияние разрушения сварных соединений на увеличение нагрузки на другие узлы конструкции. Неспособность одного или нескольких сварных соединений осуществить поддержку моста может привести к росту нагрузки на остальные его части, что приводит к более раннему отказу всей конструкции.
- Изменение нагрузки на систему в моменты включения, выключения и в штатном режиме. Каждый двигатель в штатном режиме потребляет электроэнергии на порядок меньше, чем в момент, его включения. Для примера, без технических тонкостей можно привести очень тяжелую телегу, которую тяжело сдвинуть с места, но в дальнейшем не очень тяжело поддерживать её скорость, т.е. чтобы начать движение, необходимо намного больше усилий. Начальная перегрузка длится доли

секунды по времени, поэтому самое главное, чтоб электрическая станция её выдержала и не отключилась или не сломалась.

- Изменение нагрузки на летательные аппараты на различных этапах полета. Примером тому являются модели реактивных двигателей.
- Надежность программного обеспечения на этапе эксплуатации компьютерной системы. Наиболее подходящей мерой надежности является вероятность того, что система выполняет возложенные на нее функции в течение заданного времени при условии взаимодействия аппаратуры и программы. В случае проявления ошибки в программе осуществляется ее исправление, и она продолжает свою работу. Однако, после исправления мы имеем новую улучшенную (или ухудшенную) программу с другими характеристиками надежности. Этот процесс также можно расценивать как изменение нагрузки на систему, когда исправление программных ошибок приводит к изменению работы программ, и, как следствие, изменению закона распределения времени их выполнения.
- В случае значительного колебания суточной или сезонной величины потребляемой мощности дизель-генераторов применяется режим параллельной работы. Особенность их работы заключается в том, что длительная минимальная нагрузка на него не должна быть менее 30%, и поэтому работа одного дизель генератора в этом случае недопустима. В силу этих причин используется комплекс из нескольких станций, работающих синхронно на общую нагрузку и включающихся или отключающихся автоматически, в зависимости от величины нагрузки.

Особо следует сказать о социальных и экономических проблемах, связанных с изменением нагрузки, в частности, о сроке службы нематериальных активов и объектов интеллектуальной собственности [3]. Важными характеристиками срока службы нематериальных активов являются средний срок службы и схема выбытия активов (которую иногда называют темпом изнашивания).

Например, инженерный чертеж служит до тех пор, пока изображенный на нем компоновочный узел остается в производстве, или пока чертеж не устаревает иным образом. Внесение изменений может привести к изменению срока службы чертежа. Выбытие актива происходит, когда он снимается с обслуживания или эксплуатации. Например, в случае инженерного чертежа, если деталь больше не производится по какой бы то ни было причине, то производитель теряет стоимость этого нематериального актива, несмотря на то, что мощности и

возможности для производства детали, изображенной на чертеже, все еще существуют.

В итоге можно сказать, что изменение нашрузки на систему может быть вызвано следующими причинами.

- Зависимость элементов системы. Показатели надежности технической системы обычно определяются в предположении независимости входящих в нее элементов. Это значит, что отказ какого-либо элемента никак не влияет на надежность других элементов. Данное предположение лежит в основе математического аппарата теории надежности. На самом деле, в результате отказов некоторых элементов неизбежно возникает дополнительная нагрузка на элементы, находящиеся в работоспособном состоянии. Как следствие, происходит изменение надежности элементов. Указанному явлению, называемому последействием отказов, в той или иной степени подвержена любая резервированная система. Интенсивности отказов элементов при этом изменяются в зависимости от времени эксплуатации системы, и сами элементы системы оказываются зависимыми друг от друга. В результате эффекта последействия происходит более интенсивное старение системы, и сокращение времени ее «жизни». Методов расчета надежности систем с зависимыми элементами в настоящее время практически не существует. Исключением являются лишь системы с последействием отказов, когда рассматривается только дублирование или троирование одинаковых по надежности элементов с экспоненциальными распределениями времени до отказа. При этом остается открытым вопрос, по каким законам происходит изменение интенсивности отказов элементов.
- Изменение условий эксплуатации системы. Изменение нагрузки на элементы системы и их надежности может быть связано также с условиями эксплуатации системы и влиянием ряда внешних факторов. При этом возможны ситуации, как увеличения, так и уменьшения нагрузки (нагрузочное резервирование). Может иметь место и колебательный характер изменения нагрузки. По признаку зависимости от времени нагрузка может быть дискретной, непрерывной или смешанной. Общая ситуация в научной литературе практически не изучалась. Это связано с тем, что не понятно, каким образом возникающая нагрузка на элементы системы оказывает влияние на законы распределения соответствующих случайных величин. Последние работы в этом направлении, в какой-то степени, сняли указанные препятствия и позволили исследовать надежность систем, как с увеличением, так и с уменьшением нагрузки на их элементы. Надо сказать, что расчет надежности систем с последействием отказов в общем случае невозможно выполнить без применения надлежащих программных средств.

- Многофазная работа и многофазное обслуживание системы. Предположим, что система должна пройти несколько фаз, отличающихся условиями эксплуатации. При этом допускается произвольный порядок прохождения фаз. Требуется определить последовательность прохождения фаз, для которой надежность системы будет максимальной. Конечно, оптимальный порядок будет зависеть от интенсивностей отказов системы на каждой фазе, от моментов смены этих фаз, а также от критерия оптимальности. Например, вычислительный комплекс может работать в различных условиях: на суше, в воздухе, на воде с известными интенсивностями отказов. Известны времена пребывания комплекса в данных условиях, и, значит, моменты изменений этих условий t_1 и t_2. Определить оптимальную последовательность фаз, для которой среднее время безотказной работы комплекса будет максимальным.
- Влияние факторов на продолжительность жизни. Изменение уровня жизни населения, или изменение экономических условий связано с различными факторами, например, принятие соответствующих законов, исполнение этих законов, улучшение (или ухудшение) жизненных условий, и т.п.
- Распределение времени жизни фирмы до банкротства. Имеется стабильно работающая фирма. С течением времени на фирму начинают воздействовать внутренние и (или) внешние причины, усложняющие (упрощающие) ее деятельность. Это влияет на скорость разорения или банкротства фирмы. Указанные отягощающие обстоятельства изменяют распределение времени жизни фирмы до наступления банкротства (развала, распада). Во время кризиса некоторым российским банкам, а также ряду крупных предприятий оказывалась финансовая поддержка (осуществлялись вливания средств и капиталов, инвестиции). За счет этого увеличивалось время жизни соответствующих организаций. На языке математики происходило изменение закона распределения времени жизни предприятий. Такова, например, программа по финансовой поддержке малого и среднего предпринимательства (МСП) в России. Снижение стоимости ресурсов, механизм снижения процентных ставок по действующим договорам и пр.

2. Стыковка распределений и ее обоснование

Функционирование систем часто сопровождается различными внутренними или внешними воздействиями, которые могут влиять на ее надежность. Такие воздействия могут быть вызваны различными условиями эксплуатации системы, изменением нагрузки, функционированием системы в разных климатических зонах, наличием

последействия отказов, вмешательством человека в работу системы и др. Указанные воздействия могут носить дискретный или непрерывный характер.

Мы вынуждены выйти за рамки классической теории надежности, если изменяются условия эксплуатации объекта. Тогда ось времени разбивается на промежутки, на каждом из которых может быть свой закон распределения времени до отказа. В зависимости от числа воздействий на систему различных распределений может быть два или большее количество таких промежутков. Возникает задача о том, как связать между собой эти распределения, как осуществить продолжение функции распределения времени до отказа. Это есть задача о стыковке (конкатенации, сцеплении) функций надежности системы на различных этапах ее жизненного цикла.

Предположим, что работа системы на начальном этапе характеризуется вероятностью безотказной работы $P_0(t) = e^{-\Lambda_0(t)}$. По каким-либо причинам в момент времени t_1 произошло изменение интенсивности отказов. Например, в момент времени t_1 на систему стала действовать дополнительная нагрузка, которая изменила закон распределения времени до отказа. После момента времени t_1 функция надежности системы стала равной $P_1(t) = e^{-\Lambda_1(t)}$. Возникает вопрос о распределении времени до отказа $P_c(t) = e^{-\Lambda_c(t)}$ на всем интервале функционирования системы. В работе [1] этот вопрос решался рассмотрением двух систем: без памяти и с памятью, надежность которых выше или ниже надежности исследуемой системы. Тем самым для реальной ситуации определялись двусторонние границы вероятности безотказной работы системы.

Для получения однозначной функции надежности $P_c(t)$ рассмотрим возможные толкования процесса продолжения.

1. Сдвиг графика функции надежности $y = P_1(t)$ по оси y. Тогда $P_c(t) = P_1(t) + x$, где $x = P_0(t_1) - P_1(t_1)$. Этот вид продолжения недопустим, поскольку при $t \to \infty$ может не выполняться условие $P_c(t) \to 0$.

2. Сдвиг графика функции ресурса $y = \Lambda_1(t)$ по оси y. Тогда $\Lambda_c(t) = \Lambda_1(t) + x$, где $x = \Lambda_0(t_1) - \Lambda_1(t_1)$. В этом случае для функции надежности имеет место соотношение $P_c(t) = cP_1(t)$, где $c = e^x$. Это значит, что должно произойти искажение графика $P_1(t)$, что, вероятно, также не соответствует действительности.

3. Сдвиг графика функции $y = P_1(t)$ по оси времени t. Тогда $P_c(t) = P_1(t - x)$ при $t \geq t_1$, где x таково, что $P_1(t_1 - x) = P_0(t_1)$. Этот вид продолжения обеспечивает одновременный сдвиг, как функции

надежности, так и функции ресурса $\Lambda_1(t_1 - x) = \Lambda_0(t_1)$. Он лишен недостатков первых двух способов продолжения.

В дальнейшем будем предполагать [4], что график функции $P_1(t)$ может перемещаться параллельно самому себе в направлении оси времени, что соответствует возможности его смещения в направлении оси абсцисс (направо или налево). Этот факт можно истолковать как изменение момента времени начала функционирования вспомогательной системы с законом распределения $P_1(t)$. Тогда нужно выбрать такое положение, чтобы график проходил через точку с координатами $(t_1, P_0(t_1))$, обеспечивая тем самым непрерывность функции $P_c(t)$. Приведем графическую иллюстрацию этого свойства.

Пример 1. Пусть $P_0(t) = e^{-0,1t}$, $P_1(t) = e^{-0,2t}$, $t_1 = 7$ ч.

Тогда

$$P_c(t) = \begin{cases} P_0(t), & при\ t < t_1 \\ P_1(t - x), & при\ t \geq t_1 \end{cases}, \qquad (1)$$

где x таково, что

$$P_1(t_1 - x) = P_0(t_1). \qquad (2)$$

Таким образом, величина сдвига определяется из уравнения (2), т.е. $e^{-0,2(7-x)} = e^{-0,1\cdot 7}$, откуда $x = 3,5$ ч. Вероятность безотказной работы системы на рис.1 совпадает с кривой $P_1(t)$ при $t \leq 7$ ч., а при $t > 7$ ч. она совпадает с кривой $P_2(t - x)$.

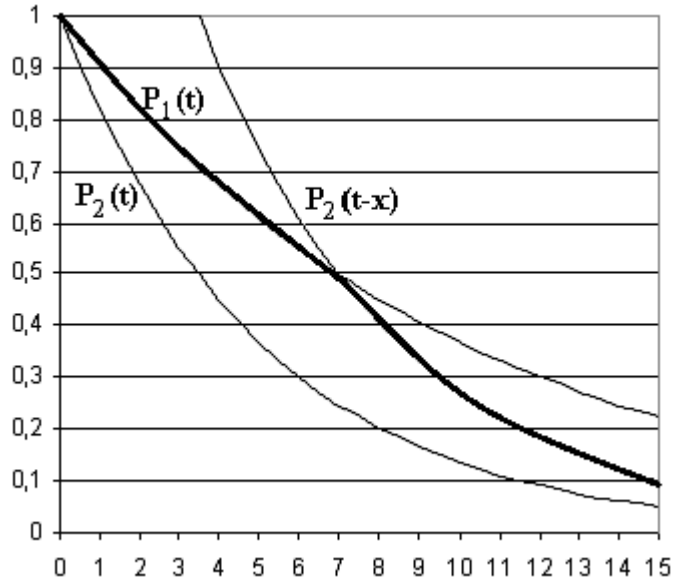


Рис. 1. График функции надежности системы, образованной стыковкой распределений (жирная линия)

Пример 2. Начальный этап работы системы характеризуется нормальным распределением времени до отказа с математическим ожиданием $m = 10$ ч. и средним квадратическим отклонением $\sigma = 2$ ч. Под воздействием некоторых причин в момент времени $t_1 = 9$ ч. (второй вариант $t_1 = 11$ ч.) закон распределения времени до отказа изменился и стал экспоненциальным с параметром $\lambda = 0{,}1\,\text{ч}^{-1}$. Требуется определить результирующий закон распределения времени работы системы с учетом изменившихся условий.

Результирующий закон распределения будем искать по формуле (1), в которой параметр сдвига x вычисляется из соотношения (2). Тогда

$$e^{-\lambda(t_1 - x)} = 1 - \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{t_1} e^{-\frac{(t-m)^2}{2\sigma^2}}\, dt,$$

или, используя обозначение функции Лапласа $\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\frac{x^2}{2}} dx$, получим

$$e^{-\lambda(t_1 - x)} = 1 - \Phi\left(\frac{t_1 - m}{\sigma}\right).$$

Отсюда

$$x = t_1 + \frac{1}{\lambda} \ln\left(1 - \Phi\left(\frac{t_1 - m}{\sigma}\right)\right).$$

Зависимость функции надежности системы $P_c(t)$ от времени представлена на рис.2 для двух значений $t_1 = 9$ ч. и $t_1 = 11$ ч.

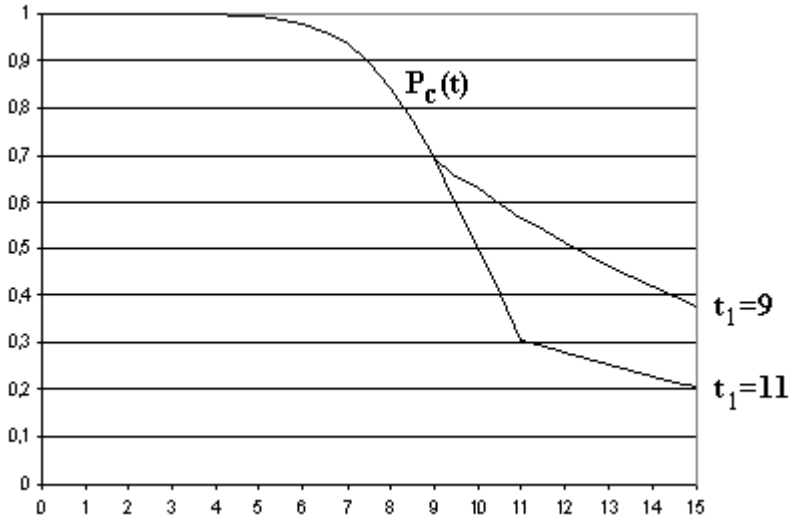


Рис. 2. График функции надежности системы

На рис.2 отчетливо виден переход с кривой Гаусса на график функции экспоненциального распределения. Данный переход соответствует моментам времени 9ч. и 11ч. Наблюдается также

существенная зависимость функции надежности системы от указанных моментов.

1. Надежность системы с нагрузкой, возникающей в случайный момент времени

Ограничимся случаем однократной нагрузки на систему, которая возникает в момент времени x, случайно распределенный на оси времени с плотностью $h(x)$. Согласно (1) вероятность безотказной работы системы с нагрузкой, возникшей в произвольный момент времени x, равна

$$P(t,x)=\begin{cases}P(t), & при\ \ t\le x\\ P_1(t-u(x)), & при\ \ t>x\end{cases},$$

где $u(x)$ – параметр сдвига, определяемый соотношением

$$P(x)=P_1(x-u(x)). \qquad (3)$$

Поскольку момент времени x – случайная величина с плотностью $h(x)$, то вероятность безотказной работы системы равна

$$P_c(t)=\int_0^\infty P(t,x)h(x)dx.$$

Интегрируя, получим

$$P_c(t)=\int_t^\infty P(t)h(x)dx+\int_0^t P_1(t-u(x))h(x)dx,$$

следовательно,

$$P_c(t)=P(t)\overline{H}(t)+\int_0^t P_1(t-u(x))h(x)dx. \qquad (4)$$

Пример 3. Предположим, что интенсивности отказов до возникновения и после возникновения нагрузки на систему постоянны и равны λ и λ_1 соответственно. Момент возникновения нагрузки является случайным с плотностью $h(x)$. Определить функцию надежности системы, учитывающую переменные условия эксплуатации.

По условию имеем $P(t)=e^{-\lambda t}$, $P_1(t)=e^{-\lambda_1 t}$. Тогда по формуле (4) получим функцию надежности системы

$$P_c(t)=e^{-\lambda t}\overline{H}(t)+\int_0^t e^{-\lambda_1(t-u(x))}h(x)dx,$$

где функция сдвига, как следует из (3), удовлетворяет соотношению

$$e^{-\lambda x}=e^{-\lambda_1(x-u(x))}.$$

Отсюда следует, что

$$u(x)=\left(1-\frac{\lambda}{\lambda_1}\right)x.$$

Подставляя это выражение в функцию надежности системы, получим

$$P_c(t) = e^{-\lambda t}\overline{H}(t) + \int_0^t e^{-\lambda_1 t + (\lambda_1 - \lambda)x} h(x)dx$$

или

$$P_c(t) = e^{-\lambda t}\overline{H}(t) + e^{-\lambda_1 t}\int_0^t e^{(\lambda_1 - \lambda)x} h(x)dx. \qquad (5)$$

Из формулы (5) нетрудно показать, что вероятность безотказной работы системы уменьшается по сравнению с первоначальной вероятностью до отказа, т.е. $P_c(t) < P(t)$, если вторичная интенсивность отказа λ_1 больше исходной интенсивности отказа λ.

Найдем среднюю наработку до отказа системы

$$T_c = \int_0^\infty P_c(t)dt = \frac{1}{\lambda} + \left(\frac{1}{\lambda_1} - \frac{1}{\lambda}\right)\hat{h}(\lambda), \qquad (6)$$

где $\hat{h}(\lambda) = \int_0^\infty h(x)e^{-\lambda x}dx$ – преобразование Лапласа плотности $h(x)$.

Если момент появления нагрузки имеет равномерное распределение на интервале $[0, b]$, то $\hat{h}(\lambda) = \frac{1 - e^{-\lambda b}}{\lambda b}$, и средняя наработка до отказа в зависимости от значения b варьируется в пределах от $\frac{1}{\lambda_1}$ до $\frac{1}{\lambda}$. Графическая иллюстрация для значений $\lambda = 0{,}01\,\text{ч}^{-1}$ и $\lambda_1 = 0{,}02\,\text{ч}^{-1}$ приведена на рис. 3.

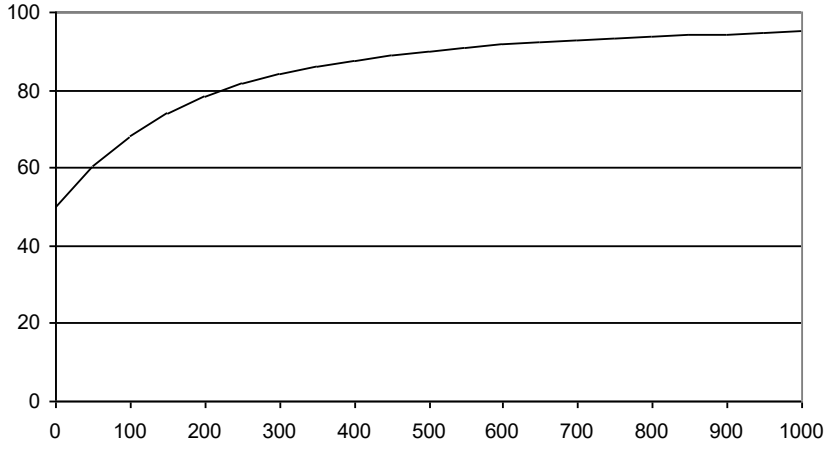


Рис. 3. Зависимость средней наработки до отказа T_c от носителя равномерного распределения (значения b)

Так, при $b=0$ средняя наработка до отказа $T_c=\frac{1}{\lambda_1}=50$ ч, а при $b\to\infty$ средняя наработка до отказа $T_c\to\frac{1}{\lambda}=100$ ч.

Пример 4. Пусть время безотказной работы имеет распределение Вейбулла с параметрами α и β до момента возникновения нагрузки и параметрами α и β_1 после возникновения нагрузки. Момент появления нагрузки на систему имеет равномерное распределение на интервале от 0 до b. Из соотношения (3) для распределения Вейбулла получим

$$\left(\frac{x}{\beta}\right)^{\alpha}=\left(\frac{x-u(x)}{\beta_1}\right)^{\alpha},$$

откуда функция сдвига равна

$$u(x)=\frac{\beta-\beta_1}{\beta}x.$$

На основе (4) получим

$$P_c(t)=\begin{cases}e^{-\left(\frac{t}{\beta}\right)^{\alpha}}\left(1-\frac{t}{b}\right)+\frac{1}{b}\int_0^t e^{-\left(\frac{t-u(x)}{\beta_1}\right)^{\alpha}}dx, & при\ \ t\le b\\ \frac{1}{b}\int_0^b e^{-\left(\frac{t-u(x)}{\beta_1}\right)^{\alpha}}dx, & при\ \ t>b\end{cases}.$$

Преобразуя это выражение и используя обозначения для неполной гамма-функции, представим функцию надежности в виде

$$P_c(t)=\begin{cases}e^{-\left(\frac{t}{\beta}\right)^{\alpha}}\left(1-\frac{t}{b}\right)+\frac{\beta\beta_1}{b(\beta-\beta_1)}\Gamma\left(1+\frac{1}{\alpha}\right)\left(I\left(\frac{1}{\alpha},\left(\frac{t}{\beta_1}\right)^{\alpha}\right)-I\left(\frac{1}{\alpha},\left(\frac{t}{\beta}\right)^{\alpha}\right)\right), & при\ \ t\le b\\ \frac{\beta\beta_1}{b(\beta-\beta_1)}\Gamma\left(1+\frac{1}{\alpha}\right)\left(I\left(\frac{1}{\alpha},\left(\frac{t}{\beta_1}\right)^{\alpha}\right)-I\left(\frac{1}{\alpha},\left(\frac{t-\frac{\beta-\beta_1}{\beta}b}{\beta_1}\right)^{\alpha}\right)\right), & при\ \ t>b\end{cases}.\quad(7)$$

Предположим, что система имеет распределение Вейбулла с параметрами $\alpha=3$ и $\beta=1000$ час. Под воздействием некоторой нагрузки, произошедшей в случайный момент времени, равномерно распределенной на промежутке от 0 до b час., параметр масштаба изменился и стал равен $\beta_1=700$ час. Требуется вычислить и сравнить значения функции надежности системы без учета и с учетом случайной нагрузки для двух значений параметра $b=100$ час. и $b=400$ час.

Расчеты, проведенные по формуле (7), позволяют определить вероятность безотказной работы системы с учетом воздействия случайной нагрузки, подчиненной равномерному распределению на интервале от

нуля до b час. Графическая иллюстрация этих расчетов представлена на рис. 4.

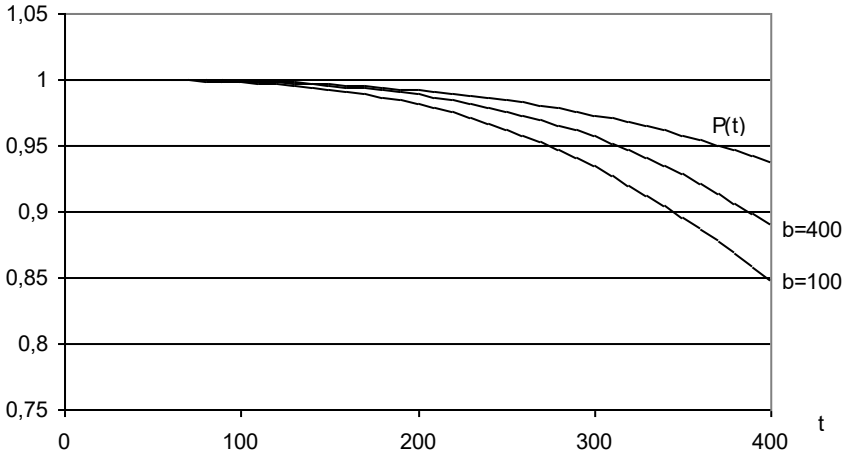


Рис. 4. Влияние случайной нагрузки на надежность системы

Очевидно, что случайный момент изменения закона распределения существенно влияет на функцию надежности системы $P_c(t)$. С уменьшением этого момента надежность системы падает. Так, график функции $P_c(t)$ при $b = 100$ час. ниже аналогичного графика при $b = 400$ час.

Явная аналитическая зависимость функции надежности от времени позволяет также установить влияние других параметров. Можно показать, что влияние нагрузки усиливается с ростом показателя формы.

Библиографический список

1. Половко А.М., Гуров С.В. Основы теории надежности. Санкт-Петербург, БХВ-Петербург, 2006, 702 с.

2. ГОСТ 25866-83 Эксплуатация техники. Термины и определения. http://zakonrus.ru/gost/g25866-83.htm.

3. Р. Рейли, Р. Швайс. Оценка нематериальных активов. Квинто-консалтинг, 2005, 792 с.

4. S.V.Gurov, L.V.Utkin, Load-share reliability models with the piecewise constant load: Int. J. of Reliability and Safety, 2012 Vol.6, No.4, pp.338 – 353.

Ю.А. Жук, кандидат технических наук, доцент

РОБАСТНЫЕ МОДЕЛИ КЛАССИФИКАЦИИ И ИХ АНАЛИЗ

Введение

Обнаружение аномальных наблюдений – это одна из задач статистического машинного обучения, которая заключается в идентификации новых или неизвестных данных. Идентификация аномалий используется в различных областях (обнаружение неисправностей, несанкционированного доступа, идентификация заболеваний и т.д.).

Особенностью задачи обнаружения аномальных наблюдений является отсутствие информации о классах, которым принадлежат объекты соответствующих наблюдений. С этой точки зрения Задачу обнаружения аномальных наблюдений можно рассматривать как частный случай задачи кластеризации, поскольку отсутствует информации о классах, т.е. когда основные данные образуют один кластер, а аномальные данные другой кластер или несколько других кластеров. В то же время задачу обнаружения аномальных наблюдений можно рассматривать как одноклассовую классификацию (ОКК), в которой имеется только один класс, а все наблюдения, не принадлежащие этому классу, являются аномальными.

Для учета выбросов и различных засорений, было предложено большое количество робастных моделей классификации [1,4,5,7]. Робастность в задачах классификации предполагает, что положение (или координата, значение признака) элемента обучающей выборки на самом деле является неточным и находится в некоторой области. Робастные модели классификации различаются в зависимости от вида области и ее размера. Рассмотрим основные элементы построения модели ОКК или модели обнаружения аномалий.

Основные элементы построения модели одноклассовой классификации или обнаружения аномалий

Пусть имеется обучающая выборка $\mathbf{x}_1,...,\mathbf{x}_n \subset \mathrm{X}$, где n – количество наблюдений в обучающей выборке, X – некоторое множество, например, компактное подмножество R^m.

Существует три основных подхода к обнаружению аномалий на основе метода опорных векторов. Основная идея, заложенная в ряде моделей обнаружения аномальных наблюдений или одноклассовой

классификации, заключается в отображении точек обучающей выборки в некоторое пространство *G*, в котором разделяющая функция *f* будет линейной. Такое отображение также называется спрямляющим.

Модель Шелкопфа

Первый подход основан на модели, предложенной в работах [8,9]. Ниже эта модель для определенности и краткости будет называться моделью Шелкопфа, хотя имеется несколько авторов этой модели.

Пусть ϕ – такое отображение $X \to G$, что точки обучающей выборки отображаются в пространство признаков большей размерности *G*. В задачах одноклассовой классификации отображение ϕ обычно переводит пространство-прообраз в часть сферы с центром в точке $0 \in G$.

Для построения отображения ϕ может использоваться произвольное ядро Мерсера $K(\mathbf{x},\mathbf{y}) = (\phi(\mathbf{x}),\phi(\mathbf{y}))$. Однако в ряде работ предпочтение отдается по ряду причин гауссову ядру, имеющему вид:

$$K(\mathbf{x},\mathbf{y}) = \exp\left(-\sigma \cdot \|\mathbf{x}-\mathbf{y}\|^2\right).$$

Здесь σ – параметр ядра, определяющий геометрическую структуру выборки в пространстве *G*. В работе [12] отмечается, что задача выбора подходящего параметра σ является чрезвычайно важной. Если используются очень малые значения параметра 1/σ (1/σ→0), то $K(\mathbf{x},\mathbf{y}) \to 0$ для всех $\mathbf{x} \neq \mathbf{y}$, и точки в пространстве признаков стремятся быть ортогональными друг к другу, несмотря на их принадлежность нормальным или аномальным наблюдениям. В этом случае, расстояние между ними очень большое независимо от того, как они располагались в исходном пространстве X, что не позволяет их разделить. С другой стороны, когда используются большие значения 1/σ $(1/\sigma^2 \to \infty)$, $K(\mathbf{x},\mathbf{y}) \to 1$ и все отображенные точки сводятся к одной точке. Очевидно, что такой подход также не может быть использован. Поэтому выбор параметра σ должен учитывать приведенные особенности отображения.

В работе [2] показано, что при использовании гауссова ядра отображенные точки обучающей выборки ложатся на поверхность сферы в пространстве признаков *G* единичного радиуса, так как $\phi(\mathbf{x}) \cdot \phi(\mathbf{x}) = K(\mathbf{x},\mathbf{x}) = 1$. Таким образом, их можно отсечь от аномальных точек при помощи плоскости $f(\mathbf{x},\mathbf{w}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle - \rho = 0$, которая является линейной. При этом наша цель – отсечь как можно более компактную область сферы, т.е. сделать параметр ρ как можно большим, так чтобы

объем полупространства, определяемого ограничением $\langle \mathbf{w}, \phi(\mathbf{x}) \rangle \geq \rho$, был как можно меньше.

Введем параметр $\nu \in [0;1]$, который обозначает долю всех данных обучающей выборки, для которых выполняется условие $\langle \mathbf{w}, \phi(\mathbf{x}) \rangle \geq \rho$. Для разделения данных необходимо решить задачу квадратичной оптимизации:

$$\min_{\mathbf{w}, \xi_i, \rho} \left(\frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i - \rho \right),$$

при ограничениях

$$\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle \geq \rho - \xi_i,$$

$\xi_i \geq 0, \ i = 1, ..., n.$

Вспомогательные переменные ξ_i используются для ослабления слишком жесткого условия, согласно которому все точки должны лежать на отсеченной полусфере.

Принятие решения осуществляется на основе решающей функции

$$g(\mathbf{x}, \mathbf{w}, \rho) = \mathbf{sgn}\big(f(\mathbf{x}, \mathbf{w}) \big) = \mathbf{sgn}\big(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle - \rho \big),$$

которая положительна в области нормальных наблюдений.

Используя множители Лагранжа $\alpha_i, \beta_i \geq 0$, запишем лагранжиан

$$L(\mathbf{w}, \xi, \rho, \alpha, \beta) = \frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i - \rho - \\ - \sum_{i=1}^{n} \alpha_i \left(\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle - \rho + \xi_i \right) - \sum_{i=1}^{n} \beta_i \cdot \xi_i.$$

В результате несложных преобразований и вычисления производных по переменным $\mathbf{w}, \rho, \xi_i$, получаем двойственную задачу квадратичной оптимизации в виде:

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \cdot \alpha_j \cdot K(\mathbf{x}_i, \mathbf{x}_j),$$

при ограничениях

$$0 \leq \alpha_i \leq \frac{1}{\nu n}, \ \sum_{i=1}^{n} \alpha_i = 1.$$

Значение ρ вычисляется для любого j от 1 до n как

$$\rho = \left(\langle \mathbf{w}, \phi(\mathbf{x}_j) \rangle \right) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}_j).$$

После подстановки полученного решения в выражение для решающей функции g получаем

$$g(\mathbf{x},\mathbf{w},\rho)=\mathbf{sgn}\left(\sum_{i=1}^{n}\alpha_i K(\mathbf{x}_i,\mathbf{x})-\rho\right).$$

Модель обнаружения аномалий Тэкса и Дьюина

Тэкс и Дьюин в работах [10,11] предложили находить часть сферы, ограничивающей шар, с минимальным объемом, содержащую все или большинство точек обучающей выборки. Поскольку объем определяется его радиусом *r* или квадратом радиуса, то объем шара минимизируется путем минимизации радиуса. С другой стороны, полученная сфера должна содержать все объекты обучающей выборки $\mathbf{x}_i$. Это приводит к следующей задаче оптимизации:

$$F(r,\mathbf{a})=\min_{r,\mathbf{a}} r^2$$

при ограничениях

$$\left\|\mathbf{x}_i-\mathbf{a}\right\|^2\le r^2,\ i=1,...,n.$$

Здесь переменная оптимизации $\mathbf{a}$ является центром сферы. Ограничения означают, что каждая точка $\mathbf{x}_i$ находится внутри сферы. Однако данное условие является слишком строгим и может привести к очень большим сферам, поэтому допускается, что некоторые точки обучающей выборки могут быть вне сферы. Это достигается введением вспомогательных переменных $\xi_i\ge 0$. Отсюда получаем следующую задачу оптимизации:

$$F(r,\mathbf{a})=\min_{r,\mathbf{a}}\left(r^2+C\sum_{i=1}^{n}\xi_i\right),$$

при ограничениях

$$\left\|\mathbf{x}_i-\mathbf{a}\right\|^2\le r^2+\xi_i,\ \xi_i\ge 0,\ \forall i=1,...,n.$$

Здесь параметр C определяет компромисс между объемом сферы и количеством ошибок (числом «выброшенных» точек данных).

Двойственная целевая функция (лагранжиан), может быть записана в виде:

$$L(r,\mathbf{a},\alpha_i,\gamma_i,\xi_i) = r^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\gamma_i\xi_i -$$
$$-\sum_{i=1}^{n}\alpha_i\left(r^2+\xi_i-\left(\|\mathbf{x}_i\|^2 - 2\langle\mathbf{a},\mathbf{x}_i\rangle + \|\mathbf{a}\|^2\right)\right).$$

Здесь α_i, γ_i, $i=1,...,n$ - множители Лагранжа. Отсюда, двойственные переменные должны удовлетворять условию неотрицательности $\alpha_i \geq 0, \gamma_i \geq 0$ для всех $i=1,...,n$.

Тэксом и Дьюином [10,11] было показано, что двойственная задача имеет вид:

$$L(r,\mathbf{a},\alpha_i,\gamma_i,\xi_i) = \sum_{i=1}^{n}\alpha_i\cdot\langle\mathbf{x}_i,\mathbf{x}_i\rangle - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j\cdot\langle\mathbf{x}_i,\mathbf{x}_j\rangle$$

при ограничениях

$$0\leq\alpha_i\leq C,\ i=1,...,n,\ \sum_{i=1}^{n}\alpha_i = 1.$$

Оптимальное множество параметров α_i соответствует максимальному значению L. Радиус сферы r вычисляется как расстояние от центра сферы до опорных векторов с весом меньшим, чем C, т.е.,

$$r^2 = (\mathbf{x}_k\cdot\mathbf{x}_k) - 2\sum_{i=1}^{n}\alpha_i\cdot\langle\mathbf{x}_i,\mathbf{x}_k\rangle + \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j\cdot\langle\mathbf{x}_i,\mathbf{x}_j\rangle$$

для любого опорного вектора $\mathbf{x}_k$ так что $\alpha_k < C$.

Новый объект $\mathbf{z}$ относят к аномальным, если $\|\mathbf{z}-\mathbf{a}\|^2 \geq r^2$. С точки зрения опорных векторов, объекты рассматриваются как нормальные, когда

$$\langle\mathbf{z},\mathbf{z}\rangle - 2\sum_{i=1}^{n}\alpha_i\cdot\langle\mathbf{z},\mathbf{x}_i\rangle + \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j\cdot\langle\mathbf{x}_i,\mathbf{x}_j\rangle \leq r^2.$$

Этот метод можно сделать более гибким при замене произведения $\langle\mathbf{x}_i,\mathbf{x}_j\rangle$ ядром $K(\mathbf{x}_i,\mathbf{x}_j) = (\phi(\mathbf{x}_i),\phi(\mathbf{x}_j))$, где ϕ - отображение функции $X\to G$ такое, что точки данных отображаются в альтернативное многомерное пространство признаков G.

Линейная модель Кэмпбела и Беннет

Рассмотрим подход на основе линейного программирования для обнаружения аномалий, предложенный в работе [2]. Авторы работы

начинают со случая, когда любая точка обучающей выборки $\mathbf{x}_j$, которая лежит за пределами предполагаемой поверхности, ограничивающей обучающие точки, принимается как аномальная. Эта поверхность определяется как множество уровня $f(\mathbf{z}) = 0$ некоторой линейной функции. В пространстве признаков, функция $f(\mathbf{z}) = \sum_i \phi_i \cdot K(\mathbf{z}, \mathbf{x}_i) + b$ соответствует гиперплоскости, которая «натягивается» на отображенные точки данных, с ограничением, что отступ всегда остается положительным или нулевым. Здесь $\phi = (\phi_n, ..., \phi_n)$ параметры функции f в пространстве признаков или множители Лагранжа.

Критерий для построения оптимальной функции $f(\mathbf{z})$, предлагаемый Кэмпбел и Беннет, минимизирует среднее значение этой функции во всех точках, т.е., минимизирует сумму $\sum_i f(\mathbf{x}_i)$. Это достигается за счет минимизации функции

$$W(\phi, b) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \phi_j \cdot K(\mathbf{x}_i, \mathbf{x}_j) + b \right)$$

при ограничениях

$$\sum_{j=1}^{n} \phi_j \cdot K(\mathbf{x}_i, \mathbf{x}_j) + b \geq 0,\ i = 1, ..., n, \quad \sum_{i=1}^{n} \phi_i = 1,\ \phi_i \geq 0.$$

Смещение b – переменная оптимизации. Возможны и другие ограничения на класс функций, например $\|\phi\|_1 = 1$ без ограничений на знак ϕ_i.

Для регулирования выбросов, вводятся нежесткие отступы по аналогии с обычным подходом в рамках метода опорных векторов. В этом случае, минимизируется следующая функция:

$$W(\phi, b) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \phi_j \cdot K(\mathbf{x}_i, \mathbf{x}_j) + b \right) + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i,$$

при ограничениях

$$\sum_{j=1}^{n} \phi_j \cdot K(\mathbf{x}_i, \mathbf{x}_j) + b \geq -\xi_i,\ \xi_i \geq 0,\ i = 1, ..., n,$$

$$\sum_{i=1}^{n} \phi_i = 1,\ \phi_i \geq 0.$$

Параметр $\nu \in [0;1]$ контролирует ошибку отступа (чем меньше ν, тем меньше выбросов игнорируются: $\nu \to 0$ соответствует жесткому ограничению отступа). Этот параметр аналогичен ν для стандартного

метода опорных векторов. Вспомогательные переменные ξ_i используется, для «смягчения» условий нарушения отступа.

Заключение

В статье рассмотрены основные модели одноклассовой классификации или обнаружения аномалий. При решении задач интеллектуального анализа данных и, в частности, при вычислении оценок параметров распределений, проблема наличия в выборке аномальных измерений имеет большое значение. Присутствие единственного выделяющегося наблюдения может приводить к оценкам, которые совершенно не согласуются с выборочными данными.

Предлагаемые модели образует множество распределений вероятностей на элементах обучающей выборки. Для решения задачи ОКК необходимо выбрать одно распределение из множества, которое максимизирует функционал риска, и одно распределение, которое минимизирует этот функционал.

Все основные модели [2,3,8,10,11] могут быть эффективны, т.е. с высокой точностью обнаруживать аномальные наблюдения, когда имеется большое количество наблюдений в обучающей выборке. Это условие не всегда выполняется. Поэтому одним из направлений дальнейших исследований является модификация основных моделей ОКК с использованием известной робастной модели ε-засорения [6]. Кроме того, предполагается разработка нечеткой модели ОКК, которая будет построена на использовании модели ε-засорения. Ее главным достоинством будет являться отсутствие необходимости задания параметра засорения ε, что не всегда возможно.

Библиографический список

1. Bouveyron, C. Robust supervised classification with mixture models: Learning from data with uncertain labels / C. Bouveyron, S. Girard // Pattern Recognition, 2009. - Vol. 42, № 11. - Pp. 2649 - 2658.
2. Campbell, C. A linear programming approach to novelty detection / C. Campbell, K. Bennett // Advances in Neural Information Processing Systems. - MIT Press, 2001. - Vol. 13. - Pp. 395-401.
3. Campbell, C. Kernel methods: a survey of current techniques // Neurocomputing, 2002. - Vol. 48, № 1-4. - Pp. 63 - 84.
4. Cerioli, A. Robust classification with categorical variables /A. Cerioli, M. Riani, A.C. Atkinson // Proceedings in Computational Statistics. – Compstat, Physica-Verlag HD, 2006. - Pp. 507-519

5. Ghaoui, L. Robust classification with interval data / L. Ghaoui, G. Lanckriet, G. Natsoulis. – California, University of California, 2003.
6. Huber, P.J. Robust Statistics. - New York: Wiley, 1981.
7. Lanckriet, G. A robust minimax approach to classification / G. Lanckriet, L. Ghaoui, C. Bhattacharyya, M. Jordan // Journal of Machine Learning Research, 2002. - №3. - Pp. 555-582
8. Scholkopf B. Support vector method for novelty detection / B. Scholkopf, R. Williamson, A. Smola et al. // Advances in Neural Information Processing Systems, 2000. - Pp. 526–532.
9. Scholkopf, B. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond / B. Scholkopf, A. Smola. – Cambridge, Massachusetts: The MIT Press, 2002.
10. Tax D. Support vector data description / D. Tax, R. Duin // Machine Learning, 2004. - Vol. 54. - Pp. 45-66.
11. Tax, D.M.J. Support vector domain description / D.M.J. Tax, R.P.W. Duin. - Pattern Recognition Letters, 1999. - Vol. 12. - Pp. 1191-1999.
12. Wang, J. Gaussian kernel optimization for pattern classification / J. Wang, H. Lu, K. Plataniotis, J. Lu // Pattern Recognition, 2009. - Vol. 42 , № 7. - Pp. 1237 - 1247.

А.М.Заяц, кандидат технических наук, профессор
З.Н. Андреева, студент

ИНТЕРПОЛЯЦИОННАЯ ОЦЕНКА РАЗВИТИЯ ДРЕВОСТОЯ С УЧЕТОМ ВЕТРОВАЛОВ И ЛЕСНЫХ ПОЖАРОВ

Ведение устойчивого лесопользования невозможно без наличия достоверной информации о состоянии леса.

Для получения такой информации используются таксационные измерения, проводимые в соответствии с требованиями Лесного кодекса и Лесоустроительной инструкции.

Результатами таких измерений является общегосударственная система нормативов таксации леса. Этими нормативами руководствуются все организации и предприятия лесного комплекса. В таксационных справочниках приводятся таблицы хода роста древостоев различного сорта и плотности древесины, сортиментные и товарные таблицы, таблицы выхода сырья и т.п.

Таблицы получены в результате многолетних обширных исследований десятков тысяч деревьев, произрастающих на многих

тысячах площадей леса. Выполнен анализ статистических данных и получены математические модели среднего диаметра древостоев по группам насаждений, относительных их полноты и запаса, хода роста древостоев и других таксационных показателей.

Задачи таксации столь обширны, что полученные математические модели не могут охватить все закономерности жизни леса.

Важным вопросом устойчивого лесопользования является доход, который можно определить как равенство скоростей истощения и прироста лесных ресурсов. Критерий устойчивого дохода показывает, что стоимость основного капитала леса (всех полезностей леса) не снижается, если скорость его уменьшения (истощения) равна скорости прироста (накопления) в этот же период:

$$V^t \text{ ис} = V^t \text{ пр} ,$$

Величина истощения лесных ресурсов в год (скорость) вычисляется как разность между уменьшением объема ресурсов и их приростом по формуле:

$$V^t \text{ ис} = Q_t + D_t - V_t \cdot q,$$

где $V_{ис}$ – физическая величина запасов лесных ресурсов в определенный период времени t, $м^3$;

Q_t – количество заготавливаемого леса в этот период t, $м^3$;

D_t – урон от пожаров и насекомых, ветровала, причиненный лесам в период t, $м^3$;

$V_t \cdot q$- скорость прироста лесных ресурсов, $м^3$/год;

$(Q_t + D_t)$- скорость истощения лесных ресурсов, $м^3$/год.

Отсюда стоит задача определения урона от пожаров и ветровала, причиненного лесам.

Решение такой задачи может осуществляться разными способами.

В лесной таксации главным образом это выражается в том, что таксаторы сами непосредственно ходят по лесосекам и делают пересчёт деревьев. Неудобство заключается в том, что если после пересчёта деревьев и полной обработки данных случается какой-либо катаклизм, который меняет состояние древостоя и объём существовавшей древесины, необходимо заново выполнять все обходы и пересчёты, а это довольно трудоёмкий и долгий процесс.

Другим решением является построение математической модели, которая покажет новое состояние древостоя по общим данным, без необходимости проводить новый пересчёт деревьев.

Предлагается один из возможных подходов в оценке развития древостоя с учетом последствий ветровалов и пожаров на основе

математического моделирования с использованием аппарата интерполирования.

Модель строится с использованием общедоступных данных о последствиях пожаров и ветровалов а также мер противопожарной охраны и существующих подходов к оценке интенсивности лесных пожаров.

В интернет - ресурсах, в общем доступе, найти какие- либо конкретные статистические данные о пожарах и ветровалах, на основе которых можно построить оценочную модель, практически невозможно. Однако есть общие сведения о пожарах, на основе которых можно получить необходимые данные для моделирования.

Данные, имеющиеся в открытом доступе на сайте межведомственной информационно-статистической системы, представлены в таблице 1.

Таблица 1

Оценка изменений запасов лесных ресурсов Ленинградской области за 2009-2014 года по кварталам в результате лесных пожаров

	2009	2010	2011	2012	2013
S1, тыс. га	5 997,4	6 032,5	6 032.3	6 036.8	6 036.6
S2, тыс. га	4 978,6	6 032,5	5 006.3	5 005.8	5 004.6
S3, тыс. га	4 771,8	4 808,6	4 810.2	4 804.6	4 794.3
Лесистость	56,9	57,3	57.3	57.3	51.7
Запас древ, млн.м^3	828,79	827,95	821.86	831.18	822.29
Sпожара, га	265	245	101	23.76	124.36
Количество пожаров	237	256	206	65	159
S восстан, га	19 562	17 409,8	20 007,5	16 546	18 714
Класс пожароопасности	4	5	2-3	2	3-4

S1 – земли, на которых расположены леса, S2- лесные земли, S3 – земли, покрытые растительностью.

На основе содержимого таблицы находим промежуточные данные и необходимые аргументы для создания модели. Результаты расчётов приведены в таблице 2.

1. На основе данных об общем запасе, общей площади лесных земель и площади пожаров находим объём сгоревшей древесины, обозначенный как **Vf.**
2. Находим процент площади, покрытой растительностью, по отношению к площади земли, на которой расположены леса, обозначенный как **L**.
3. На основе данных о размерах площади восстановления и плотности насаждения, находим объём восстановленной древесины, обозначенный как **V восст.**
4. **Mt** - запас на 1 тысячу гектар.

Т а б л и ц а 2.

Результаты проводимых расчётов с общедоступными данными по Ленинградской области за 2009-2013гг.

Год	**Mt, млн.м^3**	**Vf, млн.м^3**	**Vf,%**	**L, %**	**V восст, млн.м^3**
2009	0,13819	0.0366	0.004416	0.7956	2.15
2010	0,13725	0.0336	0.004421	0.7971	1.901
2011	0,13624	0.0138	0.001679	0.7974	2.174
2012	0,13769	0.0033	0.000397	0.7959	1.813
2013	0,13622	0.0169	0.002055	0.7942	2.025

Для создания модели необходимо также определить и представить данные о противопожарных мерах и классу пожароопасности.

Противопожарные меры включают в себя такие мероприятия как:

- Авиапатрулирование
- Наземная охрана
- Расчистка сухостоев и ветровалов
- Пропаганда охраны лесов среди людей

Пропаганда среди населения, как одна из основных мер обеспечивающих выполнение противопожарных мероприятий включает:

1. Лекции в школах и на местах работы для донесения населению информации о мерах противопожарной охраны и действиях граждан для предотвращения и сокращения пожаров.
2. Публикации в газетах, программы на радио, фильмы и передачи на телевидении об опасности лесных пожаров.

3. Социально-профилактические плакаты при въезде в леса о вреде халатного обращения с огнём и последствиях лесных пожаров.
4. Создание специальных площадок отдыха населения в лесах с соответствующим оборудованием и частым патрулированием.
5. Запрет на посещение леса населением в особо пожароопасные периоды.

Для учета этого в модели ведем условную шкалу успешности проведения профилактических мер.

Максимальное значение профилактических мер оценивается как 1, когда соблюдены абсолютно все возможные аспекты. В реальной жизни она оценивается меньше единицы, так как ограничение на посещения леса вводится только в экстренных случаях и не каждый год.

Основной причиной лесных пожаров является человеческий фактор, халатное обращение населения с огнём на территории леса, поэтому большая часть усилий должна приходиться именно на предотвращения этой причины. Анализ показывает, что значение этой величины обычно не более 0,6.

Также существуют проблемы финансирования противопожарных мероприятий, пока недостаточное техническое оснащение, и в ближайшее время ситуация вряд ли существенно изменится. Наземное патрулирование осуществляется только 70% территории Ленинградской области. Шкала осуществления противопожарных мер по годам представлено в таблице 3.

Т а б л и ц а 3

наименование		2009	2010	2011	2012	2013
Авиапатруль		0.03	0.03	0.03	0.03	0.03
Наземная охр		0.09	0.09	0.1	0.12	0.13
Ветровал		0.09	0.02	0.04	0.06	0.09
пропаганда	П1	0.06	0.08	0.09	0.1	0.1
	П2	0.08	0.1	0.12	0.13	0.14
	П3	0.07	0.08	0.09	0.09	0.09
	П4	0.09	0.09	0.1	0.12	0.14
	П5	0	0.1	0	0	0
	Итог	0.3	0.45	0.37	0.44	0.47
Планы		0.1	0.1	0.11	0.11	0.12
Итог G		0.52	69	65	76	84

Для уточнения значений класса пожароопасности вводится шкала по годам, где максимальное значение – 1. Данные приведены в таблице 4.

Классам пожароопасности присваиваются значения:

1 класс: 0-02
2 класс: 0.22-0.4
3 класс: 0.4-0.6
4 класс: 0.6-0.8
5класс:0.8-1

Т а б л и ц а 4

Год	Уровень пожароопасности K
2009	0.7
2010	0.9
2011	0.4
2012	0.35
2013	0.6

Полученные данные обеспечивают создание интерполяционной модели с использованием функции многих переменных в математическом пакете Derive.

Требуемые аргументы представлены в таблице 5.

Т а б л и ц а 5

G	**K**	**Vf%**
0.52	0.7	0.00004416
0.69	0.9	0.00004421
0.65	0.4	0.00001679
0.76	0.35	0.00000397
0.84	0.6	0.00002055

Модель зависимости объёма сгоревшей древесины строится на таких показателях, как коэффициент профилактики и коэффициент пожароопасности.

Строится линейная модель в пакете Derive 6

$Vf = b0 + b1 \cdot g + b2 \cdot k$

g	k	b0 + b1·g + b2·k
0.52	0.7	0.00004416
0.69	0.9	0.00004421
0.65	0.4	0.00001679
0.76	0.35	0.00000397
0.84	0.6	0.00002055

FIT

g	k	b0 + b1·g + b2·k
0.52	0.7	0.00004416
0.69	0.9	0.00004421
0.65	0.4	0.00001679
0.76	0.35	0.00000397
0.84	0.6	0.00002055

$$-5.817291810\cdot10^{-5}\cdot g + 6.479967802\cdot10^{-5}\cdot k + 2.795984929\cdot10^{-5}$$

Была получена линейная функция:

$-5.817291810\cdot10^{-5}\cdot g + 6.479967802\cdot10^{-5}\cdot k + 2.795984929\cdot10^{-5}$

Производится проверка полученной функции в пакете MathCad

$$G := \begin{pmatrix} 0.52 \\ 0.69 \\ 0.65 \\ 0.76 \\ 0.84 \end{pmatrix} \qquad K := \begin{pmatrix} 0.7 \\ 0.9 \\ 0.4 \\ 0.35 \\ 0.6 \end{pmatrix}$$

$$V := -5.817291810\cdot10^{-5}\cdot G + 6.479967802\cdot10^{-5}\cdot K + 2.795984929585994001\cdot10^{-5}$$

$$V = \begin{pmatrix} 4.307\times10^{-5} \\ 4.614\times10^{-5} \\ 1.607\times10^{-5} \\ 6.428\times10^{-6} \\ 1.797\times10^{-5} \end{pmatrix}$$

Из проверки видно, что полученная функция достаточно точна.
По интерполяционным кривым, полученным в результате обработки данных статистики, осуществляется прогноз ущерба на будущий период и рассчитывается скорость истощения лесных ресурсов.

Модель может быть использована для автоматизации таксационных методов оценки древостоя; экологического мониторинга последствий пожаров и ветровалов в лесных массивах; прогнозирования силы пожара и своевременного принятие профилактических мер.

Работа выполнена в рамках гранта КНВШ - **КОНКУРС ГРАНТОВ 2014 ГОДА ДЛЯ СТУДЕНТОВ ВУЗОВ, РАСПОЛОЖЕННЫХ НА ТЕРРИТОРИИ САНКТ-ПЕТЕРБУРГА, АСПИРАНТОВ ВУЗОВ, ОТРАСЛЕВЫХ И АКАДЕМИЧЕСКИХ ИНСТИТУТОВ, РАСПОЛОЖЕННЫХ НА ТЕРРИТОРИИ САНКТ-ПЕТЕРБУРГА**
Направление конкурса 2.1 - «Математика и информатика»

А.М.Заяц, кандидат технических наук, профессор
А.В.Ульянов, магистрант
Л.А.Яловка, студент

ИНФОРМАЦИОННЫЙ КИОСК КАФЕДРЫ ИСИТ

В статье представлен материал по разработке кафедрального информационного киоска (КИК), обеспечивающего терминальный доступа к информационной системе университета, на котором размещается учебная, административная и научная информация СПбГЛТУ и в часности кафедры информационных систем и технологий: новости, расписание занятий, результаты экзаменов, приказы и распоряжения, информация о ППС и научных направления и т.п.

Предполагаемое место расположения КИКа представлено на рис. 1.

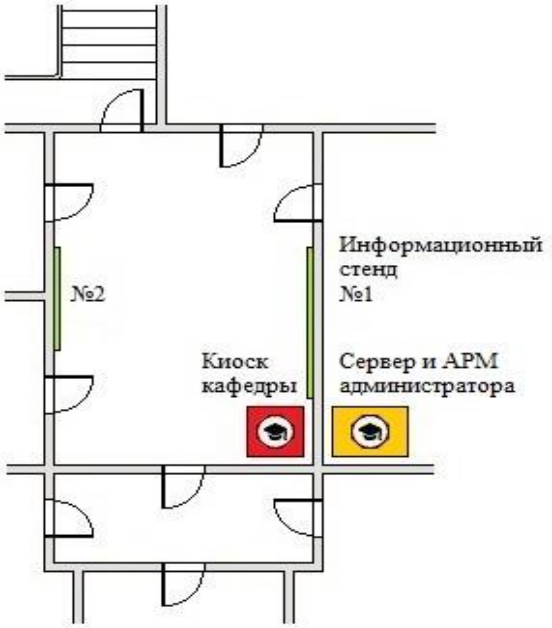


Рис.1. Расположение информационного киоска на кафедре ИСиТ

Наполнение контента КИК осуществляется по схеме рис.2 и включает:

- данные, не требующие частой актуализации (статическая информация);
- часто меняющиеся данные;
- данные изменяющиеся в реальном масштабе времени (например, новости).

Рис. 2. Схема наполнения информационного киоска

Программное обеспечение КИКа реализовано в программной среде Delphi и обеспечивает формирование различных разделов контента киоска в ранее описанных режимах. На рис. 3 представлены скриншоты главной страницы «О КАФЕДРЕ» и раздела «Сотрудники кафедры».

Рис.3. Примеры разделов информационного киоска

Меню разделов информационного киоска обеспечивает навигацию по разделам, содержащим историческую справку и современную информация о кафедре ИСиТ и ее сотрудниках, ФГОСы, учебные рабочие планы расписание занятий и учебные программы дисциплины изучаемых на кафедре, расписания занятий, научные исследования (все о научной деятельности кафедры), действующую учебно – методическую документацию кафедры, данные об образовательных ресурсах кафедры, СПбГЛТУ, сети Интернет и многое другое.

Н.В Лушкин, кандидат технических наук, доцент

АППРОКСИМАЦИЯ ГРАНИЦ ГРАФИЧЕСКИХ ОБЪЕКТОВ ОКРУЖНОСТЬЮ И ТОЧКОВКА ЛЕСА

Точковка леса это измерение объёма срубленных деревьев. Для определения объёма круглого леса используют таблицу исчисления

объёмов круглого леса, где по заданному диаметру и длине бревна определяется объём. В предлагаемой вниманию статье исследуется возможность определения объёма срубленного леса, используя графическое изображение торцов брёвен и алгоритм определения связности графических объектов [1].

Так как торцы брёвен имеют круглую форму, будем считать, что границы изображений торцов брёвен имеют вид близкую к окружности, т.е. для определения площади торцов брёвен необходимо определить диаметр окружности графического изображения. Изображения брёвен могут быть как изолированными, так, и объединены в группы связных областей по несколько брёвен (Рис.1). Ставится задача оценить диаметры брёвен находящихся на изображении графического файла. Предлагается следующие этапы решения поставленной задачи:

1) последовательно для каждого связанного объекта часть границы объекта аппроксимируем окружностью;
2) определяем координаты центров и диаметры полученных окружностей;
3) используя исходный файл с графическими объектами, уточняем полученные параметры окружностей;
4) удаляем из исходного файла объекты ограниченные полученными окружностями;
5) выполняем пункты 1-4 до тех пор, пока не переберём все объекты (брёвна).

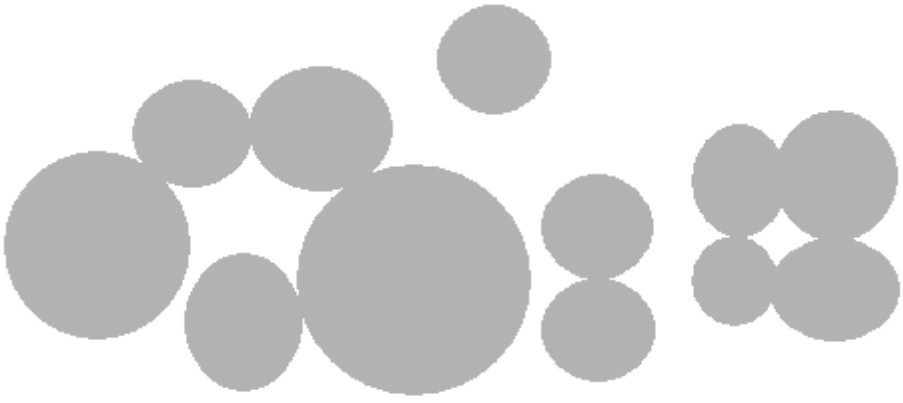


Рис. 1. Группы связных областей по несколько брёвен

Рассмотрим окружность радиуса Rmin < R < Rmax, где Rmin, Rmax – минимальный и максимальный радиусы рассматриваемых брёвен (Рис.2).

Точки N, A1, A2, ..., Ak, B1, B2, …, Bk принадлежат границе графического объекта (волнистая линия). Середины отрезков A1B1, A2B2,

…, AkBk должны принадлежать области Δ, ограниченной вертикальными линиями, так как рассматривается аппроксимация окружностью.

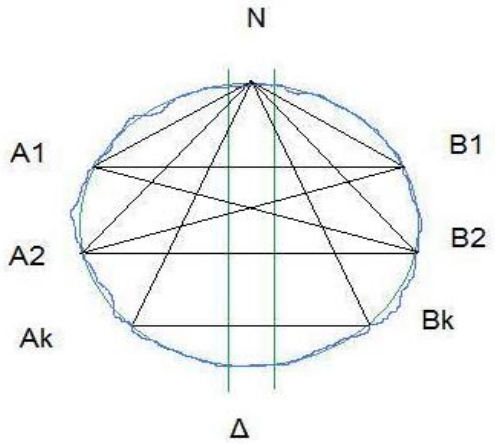


Рис.2. Аппроксимация окружностью границы объекта

Рассмотрим различные способы построения окружностей вокруг треугольников образованных тройками точек на границе графического объекта.

1) строим окружности вокруг тех треугольников, NA1B1, NA2B2, …, NAkBk, если середины их оснований A1B1, A2B2, …, AkBk, принадлежат области Δ, радиус находится в заданных пределах (Rmin < R < Rmax) и координаты центра окружности находятся внутри графической области. Находим средний радиус и средние координаты центра окружности;

2) для любого отрезка A1B1, A2B2, …, AkBk, середина которого принадлежат области Δ, строим всевозможные окружности, с токами N, A1, A2, …, Ak, B1, B2, …, Bk, если радиус находится в заданных пределах (Rmin < R < Rmax) и координаты центра окружности находятся внутри графической области. Находим средний радиус и средние координаты центра окружности.

Моделирования описанного метода на MATHCAD

- Рассмотрим окружность с центром (A,B) и радиуса R.
- Rnd(2)*RN(y)- случайное возмущение.

$$R := 30$$

$$A := 150$$

$$B := 180$$

$$\Delta := 1$$

$$y := B - R .. B + R$$

$$RN(y) := \sin\left[\pi \cdot \frac{[[y - (B - R)]]}{2 \cdot R}\right]$$

$$x2(y) := A - \sqrt{R^2 - (B - y)^2} + rnd(2) \cdot RN(y)$$

$$x3(y) := A + \sqrt{R^2 - (B - y)^2} + rnd(2) \cdot RN(y)$$

$$sered(y) := \frac{x2(y) + x3(y)}{2}$$

В результате моделирования получим множество центров окружностей (AA(y), BB(y)) и радиусов RR(y). Усредненные координаты центра окружности (153,183). Средний радиус SR=30.39. Площадь круга радиуса R=30, S=5655 в пикселях. Площадь круга Smod=5803 в пикселях. На рис. 3 отражена граница объекта и середины отрезков границы объекта.

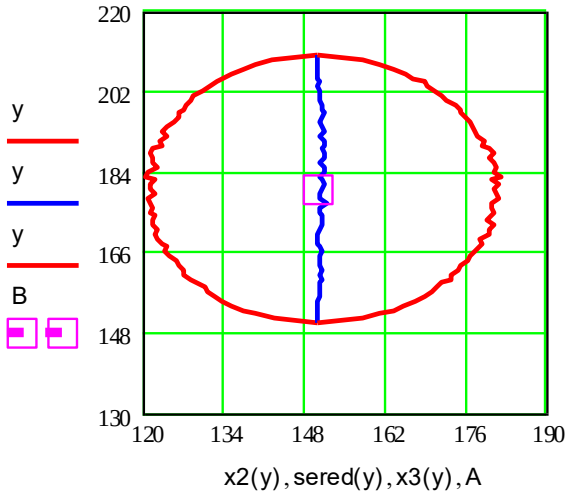


Рис. 3. Граница объекта и середины отрезков границы объекта

На рис. 4 показана зависимость от *y* координаты центра окружности (AA(y), BB(y)) и радиуса RR(y).

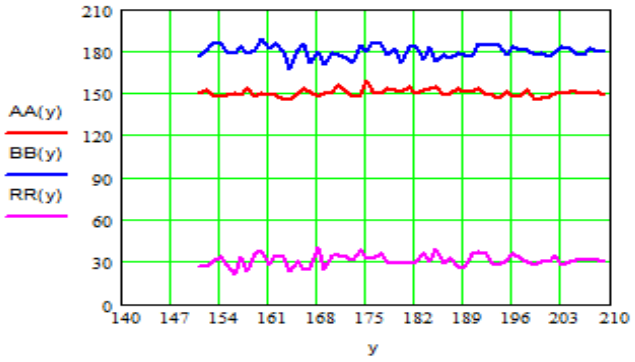


Рис. 4. Зависимость от *y* координаты центра окружности (AA(y), BB(y)) и радиуса RR(y)

Граница графического объекта может состоять из нескольких окружностей, при этом центры границ находятся на одной вертикали с погрешностью Δ.

Аналогично, как и с одной окружностью, строим окружности вокруг тех треугольников, центры отрезков которых находятся в зоне Δ.

На рис. 5 представлена модель, отражающее расположение объекта, состоящее из двух окружностей.

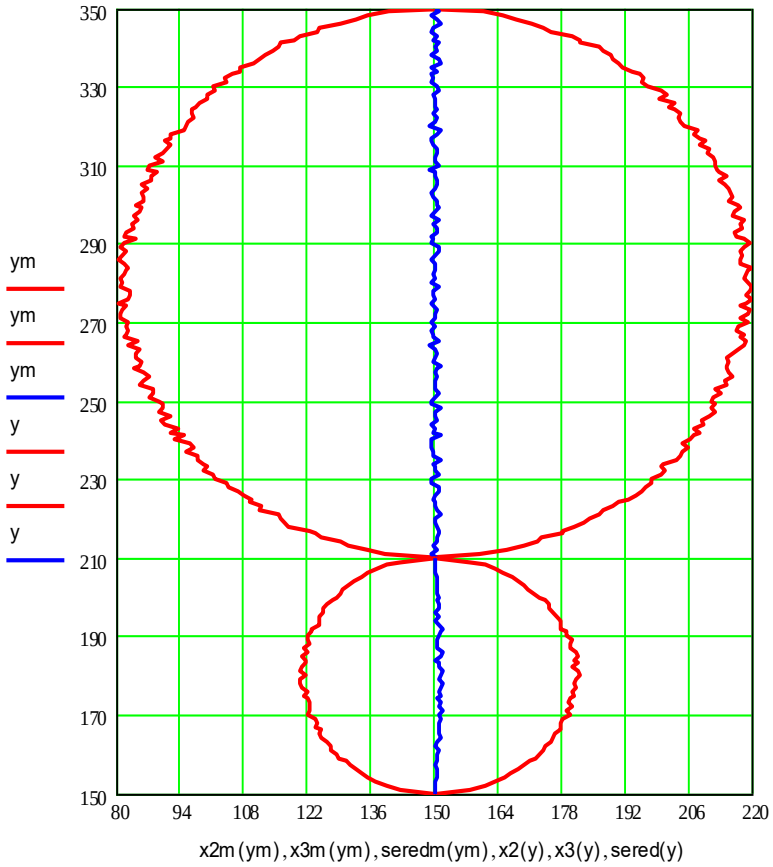


Рис. 5. Модель объекта из двух окружностей

В результате моделирования получим множество центров окружностей (AA(y), BB(y)) и радиусов RR(y). Из графиков видно (Рис.6), что при переходе построения треугольников от одной окружности к другой, происходит резкое изменение длины радиуса и координаты центра по ***y***, что является признаком остановки построения окружностей.

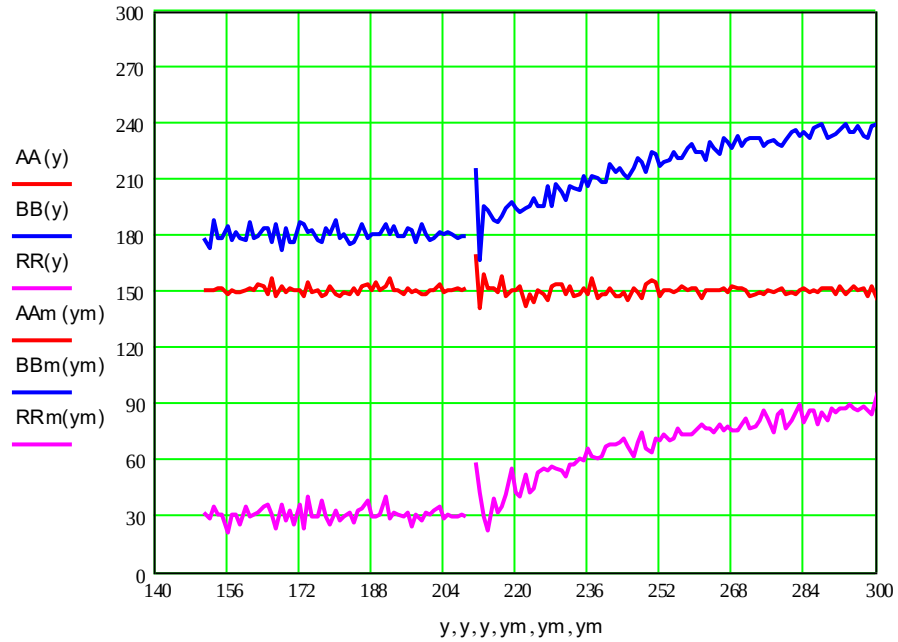


Рис. 6. Изменение длины радиуса и координаты центра по ***y***

На рис. 7 представлен тестовый результат выполнения описанного метода, где изображены четыре связных графических объекта, в котором имитируется одно бревно, два бревна, четыре и пять, при этом в каждом из графических объектов аппроксимирована часть границы окружностью.

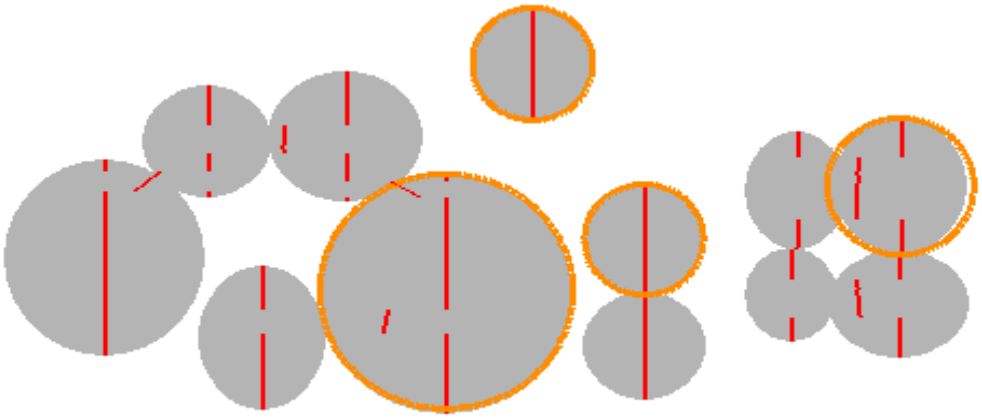


Рис. 7. Тестовый результат выполнения программы описанного метода

Работа алгоритма определения связности графических объектов

На рис. 8 представлена фотография графического объекта, где графический объект состоит из множества брёвен.

Рис.8. Фотография графического объекта

В результате работы алгоритма и программы определения связности графических объектов, по заданному критерию для исходного графического файла, получаем объекты с их границами. Часть объектов брёвен) получаются связанными между собой, часть не связанными (Рис. 9).

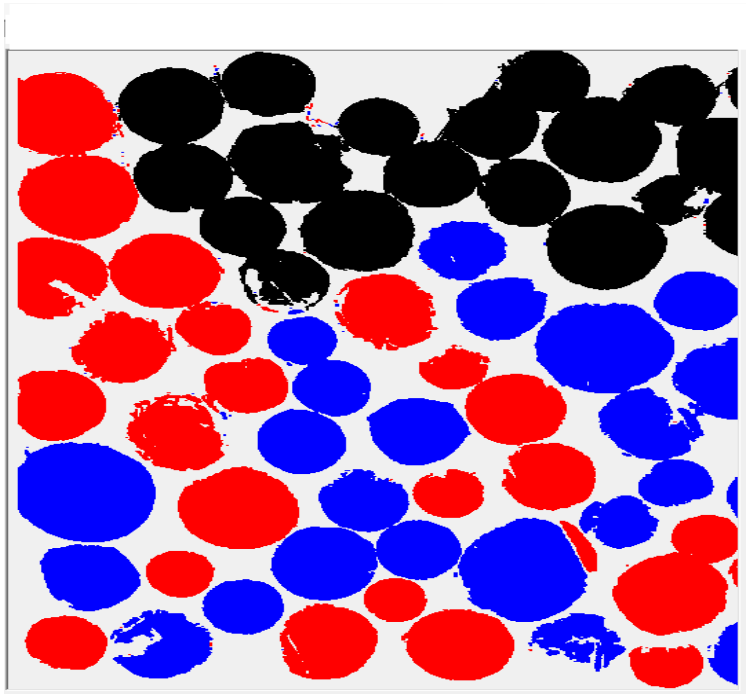


Рис. 9. Связные объекты с границами

Демонстрация первого этапа аппроксимации окружностью.

На рис. 10 представлен результат выполнения описанного метода, где изображены несколько связных графических объекта, при этом в каждом связном объекте аппроксимирована часть границы окружностью. На рисунке для каждого связного объекта построена одна окружность, используя первый способ аппроксимации.

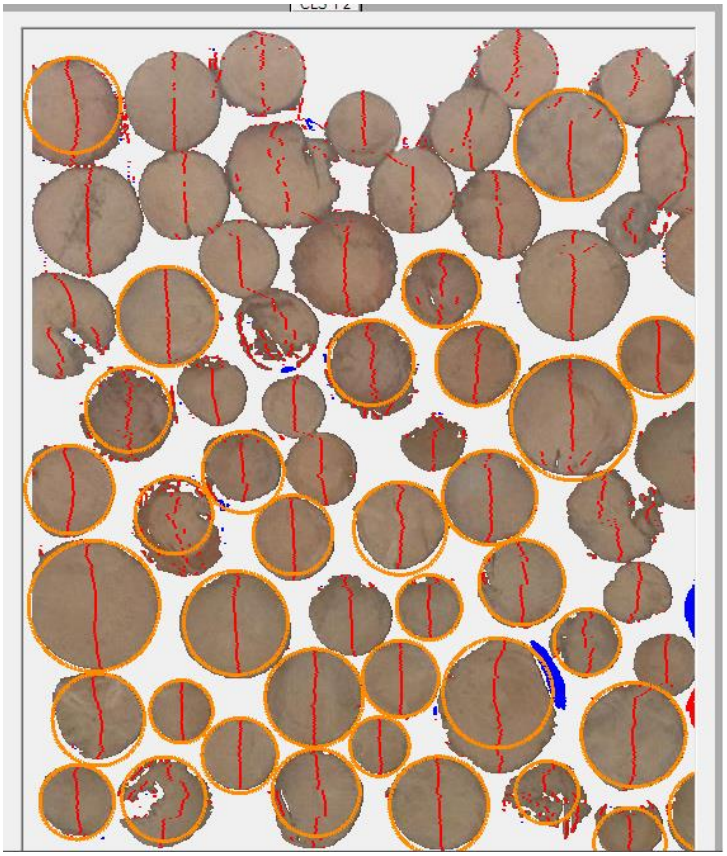


Рис. 10. Результат выполнения первого этапа программы аппроксимации.

Демонстрация второго этапа аппроксимации окружностью

После работы первого этапа алгоритма, для каждого связного объекта, для которого построена окружность с центром с координатами (A, B) и радиусом R, выполняем следующее:

- по критерию для квадратного участка, который включает данную окружность исходного графического файла, получаем объекты с их границами;

- строим уточненную окружность для данного графического объекта.

Результат работы второго этапа аппроксимации окружностью представлен на рис. 11.

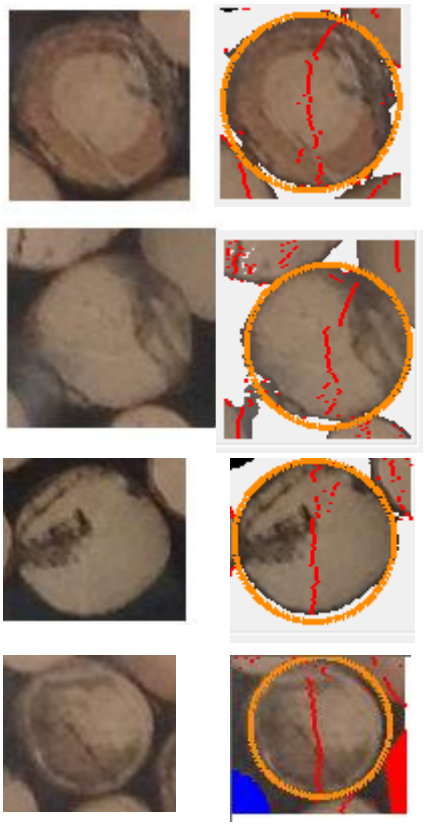


Рис. 11. Результат работы второго этапа аппроксимации окружностью

Описание следующего этапа аппроксимации окружностью

- На следующем этапе с исходного графического файла удаляются области ограниченные уточненными окружностями.
- Запускается алгоритм определения связности графических объектов.
- Переходим к первому этапу аппроксимации окружностью.
- Продолжаем данный процесс до тех пор, пока не получим все аппроксимации границ графических объектов (брёвен) с исходным графическим файлом.

Заключение. Заявленный подход, при обработке графических файлов, позволяет выявить актуальные направления и обратить внимание на развитие компьютерного анализа объектов лесной отрасли.

Представленный метод позволяет создать систему оперативной обработки и оценки объёмов лесозаготовок в лесном комплексе.

Библиографический список

2. Лушкин Н.В. Алгоритм определения связности графических объектов. - Труды Санкт-Петербургской Государственной лесотехнической академии. Актуальные проблемы развития высшей школы Санкт-Петербург. 2010. Стр. 308-309.

В.А.Пресняков, кандидат технических наук, доцент

ТЕКУЩИЕ ПРОБЛЕМЫ АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ УЧЕБНЫМ ПРОЦЕССОМ В СПБ ЛГТУ

В данной статье изложено текущее состояние автоматизации учебного процесса Санкт-Петербургского государственного университета. Рассматриваются вопросы изменения информационной инфраструктуры университета с целью повышения надежности и объединения ресурсов.

1. Текущее состояние АУП ЛТУ

В настоящее время целей автоматизации управления учебным процессом в настоящее время используются один физический компьютер. Будем его называть **сервер АУП**.

На сервере АУП установлена операционная система (ОС) Microsoft Windows Server 2008 R2, под управлением которой в автоматическом режиме работает ряд программ. Некоторые из них являются частью ОС, другие установлены дополнительно. В результате сервер АУП выполняет одновременно несколько ролей.

Роли сервера АУП

- Роль контроллера домена (КД). Эта функция позволяет контролировать (управлять) учетными записями пользователей, компьютерами, принтерами, общими ресурсами.
- Роль сервера доменных имен (DNS)
- Роль файлового сервера (FS).
- Сервер АУП (вернее его накопители) используется как хранилище данных (ХД).
- На сервере АУП развернута система управления базами данных (СУБД) Microsoft SQL Server 2012.

Базы данных

На сервере АУП развернуты следующие базы данных:

1. БД «АУП» – в ней все данные о контингенте, нагрузке, РПД, и т.п.
2. БД «Абитуриенты»
3. БД «Тестирование студентов»

Рабочие станции

В домен включены компьютеры деканатов, кафедр, отдела кадров, учебно-методического управления, а также приемной комиссии университета. В настоящее время в общую инфраструктуру АУП включено около 125 компьютеров и, возможно, в ближайшее время добавятся еще дополнительные компьютеры кафедр и компьютеры учебных лабораторий для проведения тестирования. Подключенные подразделения приведены на рис.1.

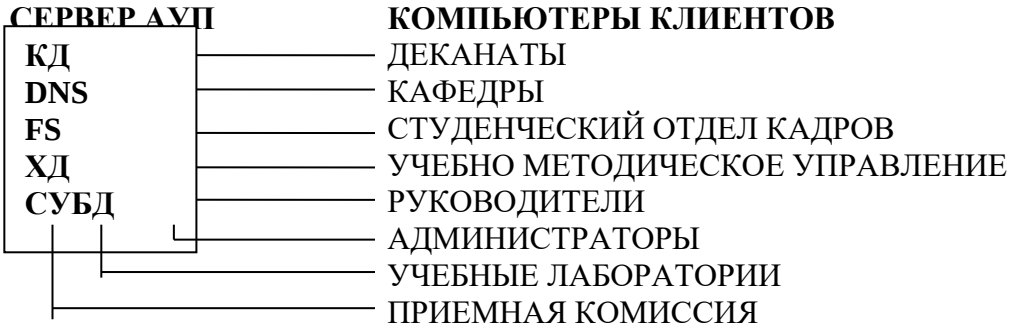


Рис.1. Текущая структура АУП

Программное обеспечение

На компьютерах пользователей установлены клиентские программы, работающие с базами данных и общими ресурсами на сервере АУП. В основном это программы, приобретенные у ООО «Лаборатория математического моделирования и информационных систем» (**ММИС**), г. Шахты [1], а также некоторые собственные разработки СПбГЛТУ. Перечень решаемых задач в настоящее время:

- учет контингента студентов,
- расчет и распределение нагрузки,
- ведение учебных планов,
- электронные ведомости кафедр,
- печать дипломов,
- назначение стипендий,
- приемная комиссия,
- рабочие программы дисциплин,
- индивидуальные планы преподавателей.

Общие ресурсы

На сервере АУП развернуты общие ресурсы (в дополнение к базам данных), доступные пользователям домена с учетом предоставленных им прав доступа. Некоторые их них:

- Учебные планы. Изменения доступны в реальном времени всем зарегистрированным пользователям. Последние версии планов выкладываются сотрудниками учебно-методического управления.
- Ведомости кафедр (Используются сотрудниками деканатов)
- Новые версии ПО (Предназначены для автоматического обновления программ, установленных на компьютерах пользователей)
- Библиотечный фонд (Импорт данных из ИС ИРБИС для подготовки рабочих программ дисциплин)

2. Необходимость резервирования

Разделение ролей сервера.

На стабильность работы системы автоматизации учебного процесса в ее текущей конфигурации может повлиять множество факторов. Некоторые перечислены ниже.

- Выход из строя аппаратного сервера АУП – отказ в работе одной из критических аппаратных составляющих сервера.
- Сбой операционной системы – выход из строя сервера в результате программного сбоя.
- Умышленное или случайное изменение данных непосредственно в базе данных
- Сбой в результате ошибок разработчиков ПО.
- Сбой электропитания – отключение электричества в результате ремонта здания, сбоя на подстанции, повреждение линий электропередач.

Избавиться от первых двух проблем можно, развернув еще один точно такой же физический компьютер-сервер, дублируя на нем информацию в режиме реального времени и обеспечив мгновенный переход на этот резервный сервер при возникновении проблем. Однако для большей независимости следует развернуть приложения и СУБД SQL на третьем физическом сервере. Кроме того, резервное управление компьютерами в домене желательно вынести на отдельный (уже четвертый) компьютер.

Более привлекательным представляется вариант с использованием платформы виртуализации с построением на одном физическом компьютере множества виртуальных машин, с развертыванием на них необходимого числа серверов. Вопросы обеспечения максимальной отказоустойчивости и одна из возможных схем построения соответствующей структуры рассмотрены в [2]. Применительно к ЛТУ

затронутые вопросы необходимо рассматривать в комплексе с общей инфраструктурой, существующей в настоящее время в лесотехническом университете. На рис.2 предлагается вариант постепенного перехода от размещения всех ресурсов на одном физическом сервере к структуре, которая позволит виртуальные компьютеры и продублировать ресурсы.

Примерная схема планируемого изменения структуры в ЛТУ приведена на рис.2.

Как уже было упомянуто, для целей АУП используется один физический компьютер, названный сервером АУП (на рисунке он обозначен, как Сервер 1). Ни одного виртуального компьютера в настоящее время НЕ развернуто. Таким образом, текущему состоянию АУП на рисунке соответствует секция (**А)**.

Для решения поставленной задачи резервирования предполагается использование одного дополнительного физического компьютера (на рисунке – Сервер 2), на котором будут развернуты виртуальные компьютеры для резервирования всех текущих ролей сервера АУП. Этот этап представлен на рисунке секцией (**Б)**.

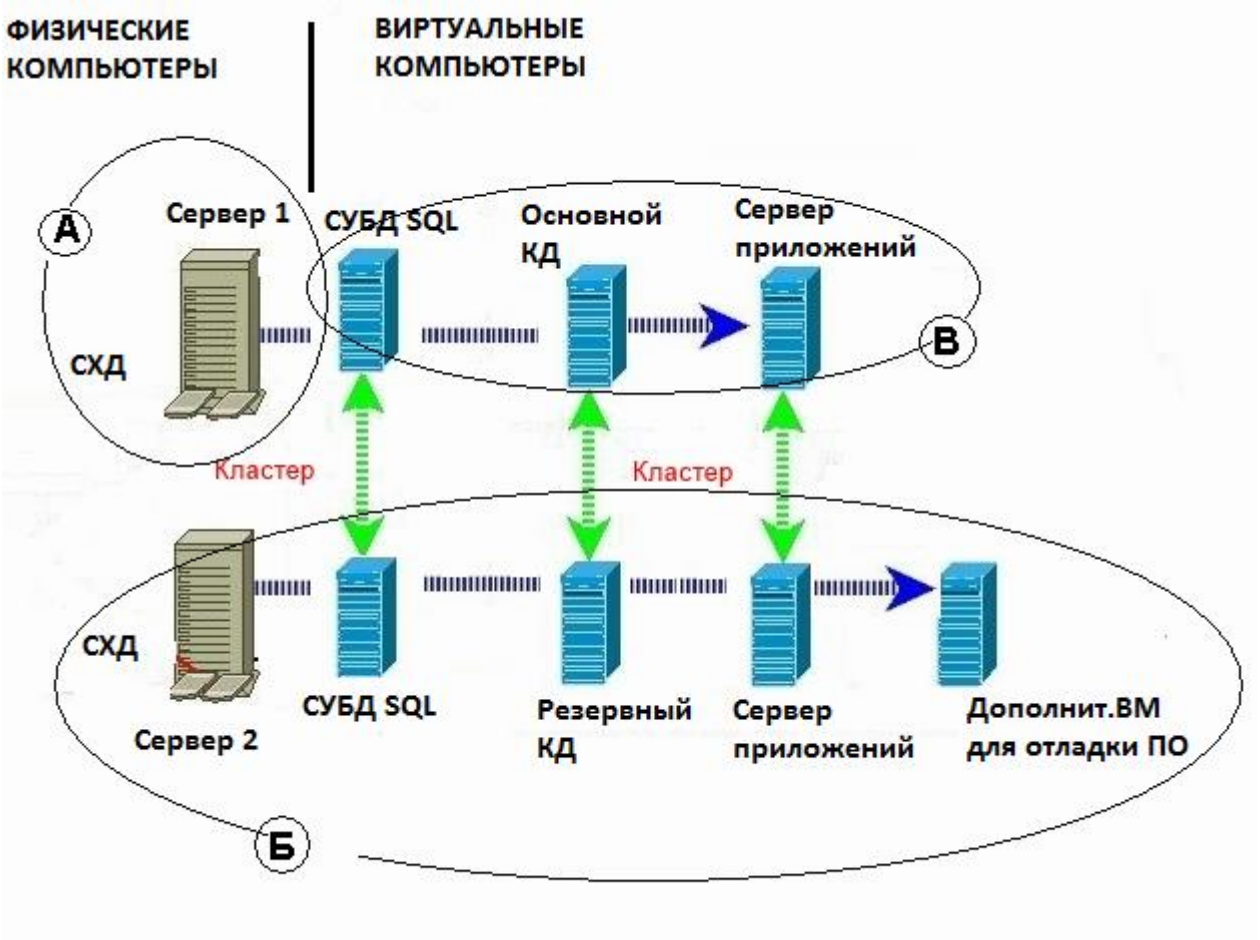


Рис.2. Схема последовательного изменения структуры АУП

После проведения всех работ по введению в действие перечисленных виртуальных компьютеров на Сервере 2, можно будет расширить объем оперативной памяти на Сервере 1 и создать на нем виртуальные компьютеры (секция (**В**)).

Примечание. Горизонтальные стрелки характеризуют процесс виртуализации, вертикальные – кластеризацию.

В результате имеем достаточно надежную систему с оптимальным распределением нагрузки и ресурсов с возможностью легкого резервного копирования и восстановления виртуальных машин. В плане развития, следует отметить то, что виртуальные компьютеры легко переносимы на другие физические компьютеры, например, при внедрении в инфраструктуру университета более мощных компьютеров будущих поколений.

3. Единая информационная система университета

Существует ряд задач, которые требуют непосредственной передачи данных из одной системы в другую. Одним из примеров является передача информации из базы данных АУП в систему ПАРУС [3] (список абитуриентов, данные о студенческих стипендиях, реквизиты для банковских систем и т.п.). Другим примеров является импортирование данных о библиотечном фонде из ИС ИРБИС [4] в базу данных АУП (этот вопрос рассматривается ниже). Информационное обеспечение осуществляется также при помощи университетского сайта [5]

Также существует проблема объединения различных программных средств и единообразие при использовании использование хранилищ данных.

Решение этих и подобных задач могло бы быть осуществлено при помощи единой информационной системы (ИС), предусматривающей единую авторизацию пользователей разных категорий, которым будут предоставляться те или иные права доступа к данным и функциональным возможностям. Такая единая ИС позволила бы объединить задачи информационного обеспечения и дистанционного обучения.

4. Использование базы данных библиотеки

Здесь речь идет об обмене данными между базой данных библиотечного фонда и базой данных АУП. В составе программ лаборатории ММИС [1] имеется модуль, предназначенный для ведения рабочих программ дисциплин. С другой стороны, база данных

библиотечного фонда формируется при помощи информационной системы ИРБИС [3]. При подготовке рабочих программ дисциплин большое значение имеет доступ к современному состоянию библиотечного фонда. Поэтому, естественным образом возникает задача синхронизации база данных, используемых в этих двух программах. Эта задача решается довольно просто при помощи процедур экспорта-импорта данных. В системе ИРБИС предусмотрено резервное копирование всей базы данных или же ее части по запросу. При формировании резервной копии имеется набор параметров, позволяющий выбрать стандарт, в котором необходимо подготовить данные, в том числе RUSMARC (ISO-2709). После того, как файл сформирован, полученные данные импортируются в базу данных АУП при помощи модуля «Рабочие программы дисциплины» (РПД).

Программа РПД позволяет настроить и сгруппировать (установить зависимость) книг библиотечного фонда с кафедрами и читаемыми дисциплинами. Управление и добавление новых поступлений в библиотеку и занесенных в ИРБИС автоматически отслеживается при импортировании в программу РПД (добавляются только новые позиции).

5. Разделение данных при тестировании студентов

При создании или использовании локальных систем для проведения тестирования студентов возникает вопрос о том, где хранить перечень вопросов и ответов к тестовым заданиям, каким образом получать доступ к информации о текущем состоянии контингента студентов.

Один из способов – это создание централизованной базы данных, которая содержала бы информацию о заданиях и результатах тестирования всех студентов университета. Такой подход имеет как положительные, так и отрицательные стороны. Несомненно, что при таком способе упрощено администрирование, однако администратор получает доступ ко всему множеству информации, которая представляет большой объем и разностороннюю направленность. В связи с последним, возникает идея разбиения базы данных с целью раздельного администрирования и управления только локальными данными, необходимыми отдельным подразделениям (факультетам или кафедрам, лабораториям или даже отдельным преподавателям).

Решение такой задачи разделения представляется возможным с использованием двух аспектов.

Первый – построение дочерних доменов с формированием дочерних контроллеров доменов в необходимом количестве для отдельных подразделений, например, для лабораторий, имеющих в своем составе

достаточное количество компьютеров для проведения тестирования. В этом случае администратор локального контроллера домена определяет политику доступа и учетные записи для своих локальных ресурсов и при необходимости устанавливает нужные связи и делегирование прав с родительским контроллером домена.

Второй аспект касается непосредственно работы с базами данных, размещенных, например, на SQL серверах. Для решения задачи управления базами данных с учетом разделения администрирования для локальных подразделений предлагается два варианта организации формирования SQL серверов. А) – Все данные хранятся и обрабатываются на основном SQL сервере ЛТУ. Б) – SQL серверы дополнительно разворачиваются в каждом подразделении. В случае варианта Б разделение администрирование уже предусмотрено, в том числе и организация резервного копирования на своих локальных ресурсах. Тем не менее, разделение администрирование можно провести и на основном сервере ЛТУ. Для этого следует использовать возможность создания определенного множества экземпляров SQL сервера, в соответствии с числом подразделений, участвующих в создании своих локальных баз данных, в том числе базы для проведения тестирования. При использовании экземпляров SQL сервера системные администраторы управляют только своим экземпляром и самостоятельно настраивают политику учетных записей экземпляра и резервного копирования. Как в случае варианта А, так и в случае варианта Б, общая база данных контингента студентов размещается на основном сервере ЛТУ. Поэтому текущие изменения, и, следовательно, текущее состояние групп и студентов, доступны всем локальным подразделениям, как из своих локальных SQL серверов, так и из экземпляров основного SQL сервера.

6. Побочные проблемы

При решении рассмотренных выше вопросов, необходимо рассматривать построение всей инфраструктуры университета в целом. Кроме общих проблем необходимо учитывать различные нюансы. Некоторые из них перечислены ниже.

Подключение дочерних доменов

При развертывании корневого контроллера домена, а также дополнительного (резервного) контроллера домена необходимо сразу учитывать и планировать работу с оборудованием и операционными системами, которые будут использоваться в инфраструктурах. Например, если планируется подключать дочерние домены, развернутые под операционной системой MS Windows 2003 Server, необходимо заранее это

предусмотреть при построении основного домена на операционной системе MS Windows Server 2012 R2.

Проблема медленных сегментов сети

Несмотря на то, что в целом сеть функционирует достаточно удовлетворительно, имеются некоторые сегменты внутренней локальной сети университета, которые работают нестабильно и имеют очень низкую пропускную способность. Из-за этого, в частности, возникают проблемы с автоматическим обновлением установленных программ на компьютерах пользователей. Например, обновление программы «Нагрузка ВУЗа» на указанных сегментах длиться более 10 минут. Прерывание процесса может привести к конфликтам в работе программ.

Неправильные версии операционных систем и местные настройки.

Имеется некоторое число компьютеров в подразделениях и на кафедрах, на которых установлены упрощенные версии операционных систем, не позволяющие включить эти компьютеры к домену АУП.

Библиографический список

[1] – http://www.mmis.ru – лаборатория ММИС: Информационные системы.
[2] - Лев Корольков. Как обеспечить максимальную отказоустойчивость информационной системы? http://efsol.ru/articles/system-fault-tolerance.html
[3] – http://www.parus.com/solutions/middle/products/parus7/ - ПАРУС. Предприятие 7.
[4] http://www.elnit.org – официальный сайт Ассоциации ЭБНИТ, разработчиков системы ИРБИС.
[5] http://www.sbpftu.ru – сайт Санкт-Петербургского государственного университета им. С.М.Кирова

С.П. Хабаров, кандидат технических наук, доцент

ИСПОЛЬЗОВАНИЕ ГРАФИЧЕСКИХ ОБЪЕКТОВ XPCE В СРЕДЕ SWI-PROLOG

Язык Prolog, как правило, ассоциируется с искусственным интеллектом, базами данных и знаний, обработкой естественно языковых конструкций и другими подобными задачами. Реальные интерактивные программы могут включать в свой состав подзадачи, для которых Prolog

является «the tool-to-use». Такие приложения часто реализуется на других языках, а Prolog используется в качестве встроенного движка для решения этих задач. Что же касается SWI-Prolog'а, то следует отметить, что в его дистрибутиве есть средство, которое позволяет разрабатывать, пусть и не очень сложный, но все же графический интерфейс пользователя. Таким средством в составе SWI-Prolog'а является XPCE.

XPCE – это платформо-независимый GUI тулкит для SWI-Prolog'а, Lisp и других интерактивных динамически типизированных языков. Хотя XPCE замышлялся, как не привязанный к конкретному языку программирования, наибольшую популярность этот фреймворк получил именно с Prolog'ом. Его развитие идет с 1987 года, с момента начала работ над SWI-Prolog'ом.

Являясь объектно-ориентированной библиотекой для разработки GUI, XPCE поддерживает окна, кнопки, меню, слайдеры, вкладки и другие базовые GUI виджеты. Ее разработчиками являются Anjo Anjewierden и Ян Wielemaker из отдела SWI Амстердамского университета.

Для того, чтобы ядро SWI-Prolog'а могло бы взаимодействовать с объектами XPCE и управлять ими из своей среды, в SWI-Prolog добавлены несколько предикатов, основными из которых являются:

- new(?Reference, +Class(...Arg...)) – создает новый объект как экземпляр одного из классов XPCE (Class) с набором аргументов, используемых для инициализации объекта. Новому экземпляру объекта может быть присвоен указатель, доступный по ссылке (Reference) из SWI-Prolog'а.
- send(+Reference, +Method(...Arg...)) – позволяет для объекта, которой указан ссылкой на него (Reference), вызвать нужный метод (Method) с требуемыми значениями аргументов.
- get(+Reference, +Method(...Arg...), -Result) – позволяет для объекта, указанного ссылкой (Reference), получить значения ряда параметров (Result), определенных в качестве аргументов (Arg) конкретного метода (Method).
- free(+Reference) – уничтожает объект, указанный ссылкой на него.

Естественно, что в этой небольшой статье автор не ставит перед собой задачу подробного изучения классов, объектов и методов XPCE, а также способами взаимодействия с ними. Тем более, что имеется достаточно подробная документация по этому тулкиту [1,2]. Цель статьи значительно скромнее – показать возможности SWI-Prolog/XPCE по разработке графических интерфейсов.

С этой целью рассмотрим небольшой пример. Пусть имеется простейшая база знаний, которая хранится в файле D:/Prolog/prog_1.pl

```
/* Программа 1 */
```

```
likes('Иван','Марья').                  % факт
likes('Петр','футбол').                 % факт
likes("Иван",X) :- likes("Петр",X).     % правило
```

Требуется разработать графический интерфейс позволяющий формировать стандартные запросы к этой базе знаний.

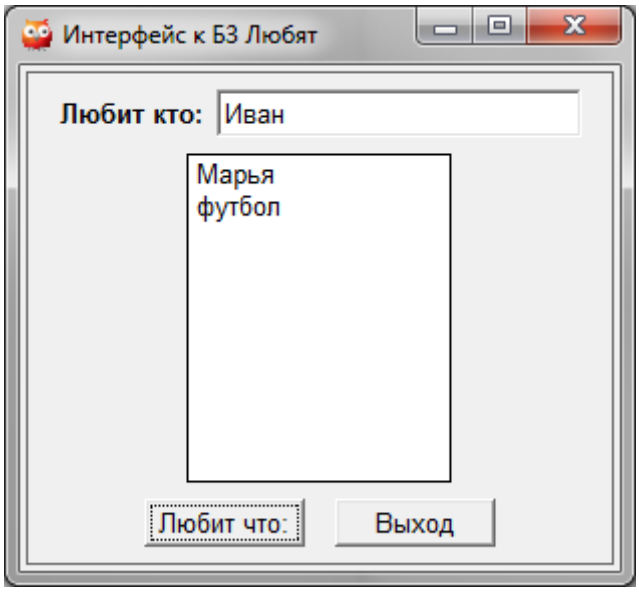


Рис. 1. Графический интерфейс к программе 1.

Как видно из рис. 1, проектируемый интерфейс будет представлять собой окно диалога, на котором размещены четыре элемента управления: текстовое поле для ввода запроса, список – для вывода результатов и две кнопки. Одна из них запускает запрос на выполнение, а другая завершает программу. Один из возможных вариантов реализации этой задачи может иметь вид:

```
/* Программа 2 */
 1 :- include('D:/Prolog/prog_1.pl').
 2
 3 gui_to_likes :-
 4    new(MyWin, dialog('Интерфейс к БЗ Любят')),
 5    send_list(MyWin, append, [
 6         new(Who,text_item('Любит кто')),
 7         new(MyList, list_browser),
 8         button('Любит что:', message(@prolog,
 9                 output, MyList, Who?selection )),
10         button(выход, message(MyWin, destroy))
11    ]),
12    send(MyList, alignment, center),
```

```
13    send(MyWin, open(point(100,400))).
14
15 output(FrmList,X) :-
16             send(FrmList, clear),
17             likes(X,W),
18             send(FrmList, append, W),
19             fail.
```

В этой программе выполняется текстовое подключение исходной базы знаний (строка 1) и определяются два предиката: gui_to_likes/0 и output/2. Первый из них формирует экранную форму, второй выполняет запрос и его результаты выводит в элемент управления список этой формы.

Определение предиката gui_to_likes/0 включает в свой состав четыре соединенных запятыми предиката. Первый предикат (строка 4) – создает новый объект класса окна диалога, связывает ссылку на него с переменной MyWin и определяет текст заголовка этого окна.

Второй предикат (строки 5-11) – вызывает для объекта, ссылка на которой определена в MyWin, метод append со списком вновь создаваемых объектов, определенных в XPCE классов.

Третий предикат (строка 12) – центрует положение на форме объекта ссылка на который определена в MyList. Этот объект представляет собой элемент управления список, созданный в строке 7 программы.

Четвертый предикат (строка 13) – отображет на экране окно, которое было созданно в памяти компьютера на предыдущих шагах программы, и размещает его на экране так, что левый верхний угол окна будет на экране иметь координаты x=100 и y=400.

Более подродно остановимся на списке новь создаваемых объектов экранной формы, которые перечислены в методе append второго предиката:

- В строке 6 создается объект определенного в XPCE класса text_item. Это текстовое поле, позволяющее вводить информацию. При его определении указывается, что ссылка на этот объект сохраняется в переменной Who, что позволяет в дальнейшем по этой ссылки обращаться к объекту и вызывать его методы. Значение аргумента label этого объекта определяется как 'Любит кто'.
- В строке 7 создается объект класса list_browser (список). Ссылка на него сохраняется в MyList. Именно в этот объект будут выводиться результаты запроса.
- В строке 8 и 10 создаются объекты класса button (кнопка). Для них определены значения аргументов label, отображаемые в виде текста на кнопке, и обработчики событий, возникающих при нажатии на эти кнопки: message(…).

Обработчик события message(MyWin,destroy) приводит к вызову метода destroy (уничтожение) для объекта указанного ссылкой MyWin. Так как с этой переменной было связано диалоговое окно с размещенными на нем элементами управления, то это окно будет выгружено из памяти, что привет к окончанию работы программы. Обработчик нажатия кнопки, вида

```
message(@prolog, output, MyList, Who?selection ),
```

имеет более сложную структуру, вызывая методы и аргументы объекта, на который указана ссылка в виде @prolog. Это абсолютная ссылка на объект, которым является SWI-Prolog. Этим указано, что при обработке нажатия кнопки следует вызвать определенный в среде Prolog предикат output/2 и передать ему два аргумента:

- MyList – указатель на объект списка в экранной формы и
    - Who?selection – содержимое объекта, ссылка на который указана в переменной Who.

Что касается предиката output(FrmList,X), то его структура аналогична предикату goal из рассмотренных ранее примеров программ. Отличие в том, что вместо предиката write в описании output/2 используем предикат send(FrmList, append, W), которой вызывает метод append с аргументом W для объекта FrmList. Но переменная FrmList при вызове output/2 из обработчика нажатия кнопки унифицирована переменной MyList. Точно также, как переменная X унифицирована значением Who?selection.

Это значит, что переменная FrmList в текущий момент содержит ссылку на объект список экранной формы, а значение переменной X содержит значение, которое было введено в поле ввода экранной формы. Поэтому вызов send(FrmList, clear) приведет к очистке списка на экранной форме, а после выполнения likes(X,W), добавлению в список на экранной форме значений переменной W с помощью предиката send(FrmList, append, W).

Следует отметим, что в XPCE включен достаточно большой набор элементов управления, которые могут быть использованы в окнах диалога. Не вдаваясь в подробности, только перечислим часть предопределенных в XPCE классов: button, text item, int item, slider, menu, menu bar, label, list browser, editor, tab, tab stack, dialog group.

О назначении большинства из них можно догадаться уже по их названиям. Более подробно узнать об определенных в XPCE классах и их иерархии можно по справке, которая встроена в систему. Например, можно в главном меню SWI-Prolog-Editor выбрать опцию XPCE -> Class browser, а затем в поле со списком Class экранной формы ввести или выбрать из списка интересущий нас класс. Это позволит получить его описание и иерархию его определения (рис. 2).

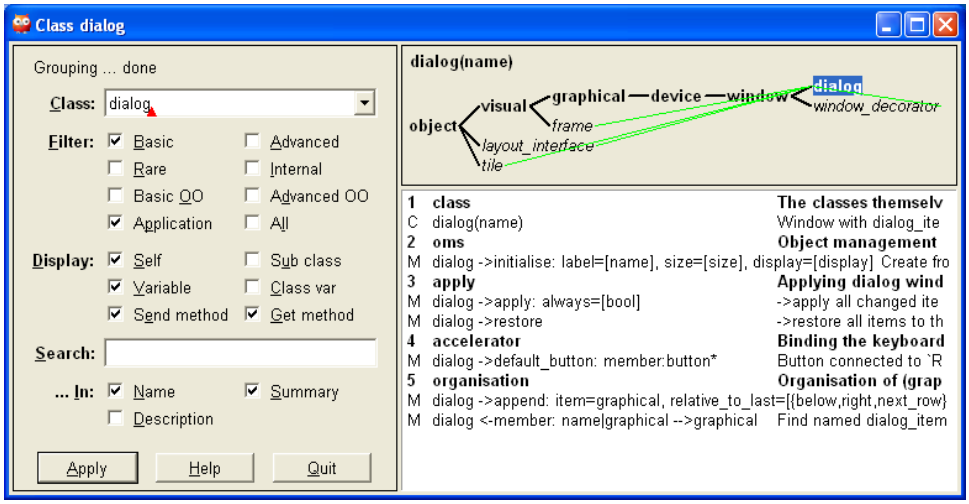


Рис. 2. Описание класса dialog в окне Class browser.

Однако возможен и другой подход, при котором в главном меню SWI-Prolog-Editor выбираем опцию XPCE -> Class hierarchy. Далее, раскрывая дерево оъектов, выбираем интересующий нас класс, правой кнопкой мыши вызываем всплывающее меню. В нем выбираем опцию Documentation для вызова документа с описанием этого класса (рис. 3).

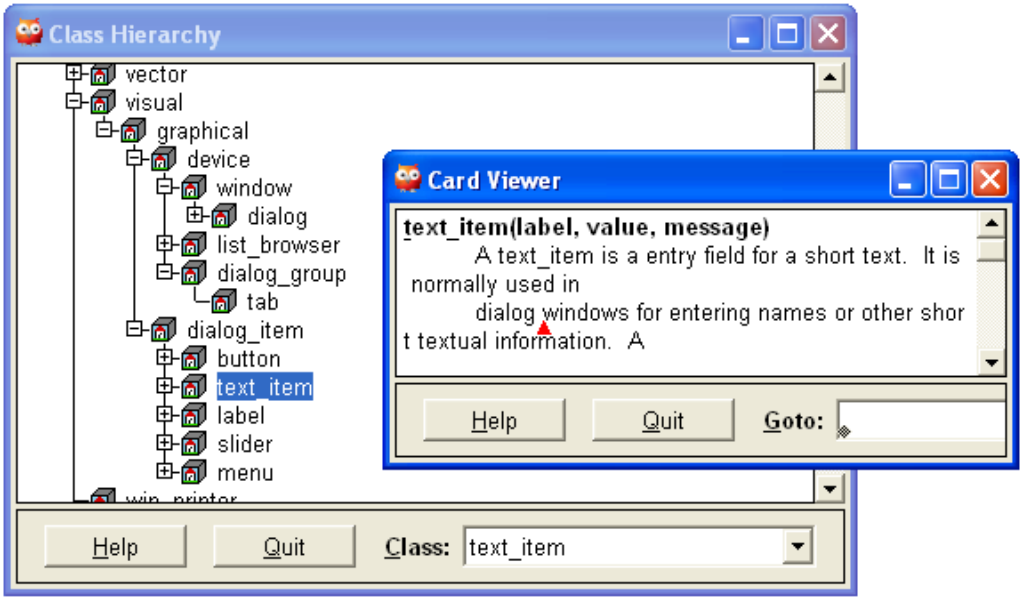


Рис. 3. Дерево классов и окно документации по text_item.

Построение экранных форм на основе окон диалога и компонентов класса dialog_item представляет собой достаточно простую задачу после получения некоторых навыков работы с ними. Этот подход удобен еще и тем, что для простых форм даже можно не указывать месторасположение элементов управления на форме. Это XPCE выполнит сам, но если что-то

не устраиваетт, то можно просто указать, что тот или иной компонент должен располагаться справа или слева от другого, или с новой строки.

Однако в XPCE существует и более развитый набор графических средств, которые позволяют решать широкий класс графических задач. Для этого в XPCE есть класс picture, который на экране отображается в виде окна с полосами прокрутки, а с точки зрения работы с ним разработчика – это практически бесконечная двумерная область, в любом месте которой может быть отображен тот или иной графический примитив. К основным таким примитивам XPCE обчно относят: arrow, bezier, bitmap, pixmap, box, circle, ellipse, arc, line, path, text.

Для знакомства с возможностями XPCE по разработке графических приложений рассмотрим два простейших примера, которые заимствованы из книги Jan Wielemaker, Anjo Anjewierden “Programming in XPCE/Prolog”, но несколько модернизированы.

```
/* Программа 3 */
1 graf_test :-
2   new(Pict, picture('Вывод графических примитивов')),
3   send(Pict,open),
4   send(Pict,display, new(Box,box(100,100))),
5   send(Pict,display, new(Cir,circle(50)), point(25,25)),
6   send(Pict,display, new(BM,bitmap('folder.ico')),
7                      point(100,100) ),
8   send(Pict,display, new(Txt,text('Привет')),
9                      point(120, 50) ),
10  send(Pict,display, new(BZ, bezier_curve(point(50,100),
11                                point(120,132),
12                                point(50, 160),
13                                point(120, 200))) ),
14   send(Pict, display, new(Mes, text('Нажми Enter,
15              пожалуйста ... ')), point(220, 150) ),
16   get0(_), free(Mes),
17   send(Box, radius, 10),
18   send(Cir, fill_pattern, colour(orange)),
19   send(Txt, font, font(times, bold, 18)),
20   send(BZ,  arrows, both).
```

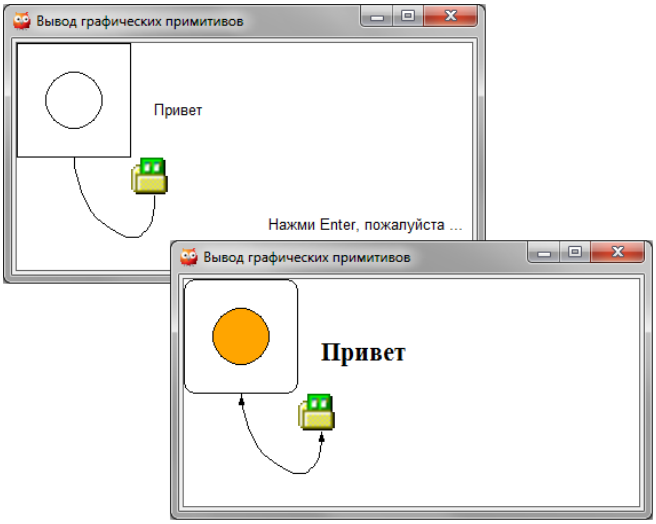


Рис. 4. Результат работы программы 3 до и после нажатия Enter.

Программа 3 состоит из двух частей. Их разделяет строка 16, где ожидается нажатие Enter. В первой части – создается новый объект класса picture, ссылка на который сохраняется в переменной Pict. Затем на этом объекте создаются различные графические примитивы с определенным набором свойств. Вторая часть программы показывает, как в процессе ее работы можно изменить свойства ранее созданных графических объектов или вообще удалить их.

Еще один тестовый пример иллюстрирует совсем не типичную для Prologa'а вычислительную задачу расчета и построения графика функции sin(x) в диапозоне от 0 до 360 градусов (рис. 5).

```
/* Программа 4 */
draw_sin :-
      send(new(Pict, picture('Функция SIN(X)')), open),
      W is 400, H is 220,
      send(Pict, size, size(W,H)),
      send(Pict, display, line(0,H/2,W-20,H/2)),
      send(Pict, display, new(P, path(kind:=poly))),
%     send(Pict, display, new(P, path(kind:=smooth))),
      (  between(0, 360, X),
         Y is (H/2 - sin((X * 6.283185)/360) * 100),
         send(P, append, point(X, Y)),
         fail
      ).
```

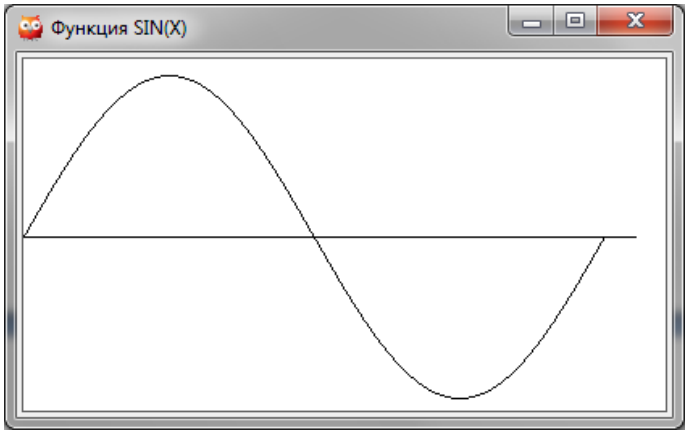


Рис. 5. Результат работы программы 4.

Этот пример приводится исключительно в целях демонстрации графических возможностей XPCE.

Библиографический список

1. SWI-Prolog documentation [Электронный ресурс]. – Режим доступа: открытый ресурс; постоянный адрес в Интернет: – http://www.swi-prolog.org/pldoc/index.html – Загл. с экрана.

2. Wielemaker Jan, Anjewierden Anjo. Programming in XPCE/Prolog [Электронный ресурс]. – Режим доступа: открытый ресурс; постоянный адрес в Интернет: http://www.swi-prolog.org/packages/ xpce/UserGuide/ – Загл. с экрана.

3. *Хабаров, С.П.* Интеллектуальные информационные системы. PROLOG – язык разработки интеллектуальных и экспертных систем: учебное пособие. — СПб.: СПбГЛТУ, 2013. — 140 с.

С.П. Хабаров, кандидат технических наук, доцент

ОБЛАСТИ ПРИМЕНЕНИЯ И СОВРЕМЕННОЕ СОСТОЯНИЕ ЯЗЫКА PROLOG

Интерес к языку Prolog поднимался и затихал несколько раз, энтузиазм сменялся жёстким неприятием. Наиболее высокий интерес к языку Prolog, как к языку будущего, возник в 1980-х годах во время разработки японской национальной программы «компьютеры пятого поколения». В рамках этого проекта разработчики надеялись, что с помощью Prolog можно будет сформулировать новые принципы, которые приведут к созданию

компьютеров более высокого уровня интеллекта. Из этого подхода следовало, что не человека надо обучать мышлению в терминах операций компьютера, а компьютер должен выполнять инструкции, свойственные человеку. И на этом историческом этапе некоторые ученые и инженеры считали подобный путь простым и эффективным.

Prolog – язык программирования, который основан не на алгоритме, а на логике предикатов. Если программа на алгоритмическом (процедурном) языке является последовательностью инструкций, выполняющихся в заданном порядке, то Prolog, будучи декларативным языком, содержит только описание задачи, а Prolog-машина выполняет поиск решения на основе этого описания, используя механизм поиска с возвратом и унификацию.

Разработка языка Prolog началась в 1970 г. Аланом Кулмероэ (Alain Colmerauer) и Филиппом Русселом (Philippe Roussel) в университете города Марсель, Франция. Они хотели создать язык, который мог бы делать логические заключения на основе заданного текста. Их работа была частично мотивирована желанием примирить использование логики в качестве декларативного языка представления знаний с процессуальным подходом к представлению знаний, который был популярен в Северной Америке в конце 1960-х и начале 1970-х годов

Название Prolog было выбрано Филиппом Русселом как аббревиатура от французского PROgrammation en LOGique (PROLOG) и первая реализация этого языка с использованием компилятора Николауса Вирта "Algol-W" была закончена в 1972 году. Основы современного языка были заложены в 1973 году. Prolog постепенно распространялся среди тех, кто занимался логическим программированием, в основном благодаря личным контактам, а не через коммерциализацию продукта.

Только в 1977 году Д. Уоррен и Ф. Перейра в университете Эдинбурга создают очень эффективный компилятор языка Prolog для ЭВМ DEC-10 и переводят методы логического программирования в практическую плоскость. Интересно, что компилятор был написан на самом Prolog. Эта реализация Prolog, известная как "эдинбургская версия", фактически стала первым и единственным стандартом языка. Как правило, если современная Prolog-система и не поддерживает эдинбургский Prolog, то в ее состав входит подсистема, переводящая Prolog-программу в "эдинбургский" вид.

Позднее в 1980 году К. Кларк и Ф. Маккейб в Великобритании разработали версию Prologa для персональных ЭВМ. А в 1996 году международной организация по стандартизации (ISO - International Organization for Standardization) была создана по языку программирования Prolog рабочая группа ISO/IEC JTC1/SC22/WG17 и принят стандарт ISO/IEC 13211-1 (Part 1: General core). Этот стандарт имеет две коррекции

в 2007 и 2012 годах. Кроме этого в 2000 году был принят дополнительный стандарт ISO/IEC 13211-2 (Part 2: Modules), который стандартизует использование в Prolog модулей.

Как видно из приведенных выше стандартов, слухи о безвременной кончине Prolog не являются достоверными [1]. Он и сейчас остается наиболее популярным языком искусственного интеллекта в Японии и Европе. В США, традиционно, более распространен другой язык искусственного интеллекта — язык функционального программирования Лисп.

В развитии языка Prolog наблюдаются очень интересные тенденции. Этот язык быстро приобрел популярность в Европе как инструмент практического программирования. В Японии вокруг языка Prolog были сосредоточены все разработки компьютеров пятого поколения. С другой стороны, в США этот язык в целом был принят с небольшим опозданием в связи с некоторыми историческими причинами [2].

Одна из них состояла в том, что США вначале познакомились с языком Microplanner, который был близок к идее логического программирования, но неэффективно реализован. Определенная доля низкой популярности Prolog в этой стране объясняется также реакцией на существовавшую вначале "ортодоксальную школу" логического программирования, представители которой настаивали на использовании чистой логики и требовали, чтобы логический подход не был "запятнан" практическими средствами, не относящимися к логике.

В прошлом это привело к распространению неверных взглядов на язык Prolog. Некоторые считали, что на этом языке можно программировать только рассуждения с выводом от целей к фактам. Но истина в том, что Prolog является универсальным языком программирования и на нем можно реализовать любой алгоритм. Позиция "ортодоксальной школы" была преодолена практиками, которые использовали более прагматический подход, объединив декларативный подход с традиционным, процедурным. Одной из самых мощных реализаций Prolog в настоящее время является система Visual Prolog, представляя собой полнофункциональную среду программирования. При этом он имеет богатую историю и является развитием и наследником таких систем как Turbo Prolog и PDC Prolog.

Области применения языка Prolog

Базовым принципом языка является равнозначность представления программы и данных (декларативность), отчего утверждения языка одновременно являются и записями, подобными записям в базе данных, и правилами, несущими в себе способы их обработки. Сочетание этих качеств приводит к тому, что по мере работы Prolog-системы знания

накапливаются. Накапливаются как данные, так и правила. Поэтому Prolog-системы считают естественной средой для накопления базы знаний.

База знаний — важный компонент интеллектуальной системы. Они предназначены для поиска способов решения проблем из некоторой предметной области, основываясь на записях базы знаний и на пользовательском описании ситуации.

Простые базы знаний могут использоваться для создания экспертных систем хранения данных в организации: документации, руководств, статей технического обеспечения. Главная цель создания таких баз — помочь менее опытным людям найти уже существующее описание способа решения какой-либо проблемы

Говоря об областях применения Prolog следует сразу оговориться, что он слабо приспособлен для решения задач, связанных с обработкой графики, вычислениями или численными методами. Вместе с тем он с успехом может использоваться в компьютерной алгебре, которая в отличие от численных методов, занимается реализацией аналитических методов решения математических задач на компьютере и предполагает, что исходные данные, как и результаты решения, сформулированы в аналитическом виде. В качестве основных областей применения Prolog можно отметить следующие направления:

- разработка быстрых прототипов прикладных программ;
- управление производственными процессами;
- создание динамических реляционных баз данных;
- перевод с одного языка на другой;
- создание естественно-языковых интерфейсов;
- реализация экспертных систем и оболочек экспертных систем;
- создание пакетов символьных вычислений;
- доказательства теорем и интеллектуальные системы, в которых возможности языка Prolog по обеспечению дедуктивного вывода применяются для проверки различных теорий.

Prolog нашел применение и в ряде других областей, например, при решении задач составления сложных расписаний. Он используется в различных системах, но обычно не в качестве основного языка, а в качестве языка для разработки некоторой части системы. Достаточно часто Prolog используют для написания функций взаимодействия с базами данных.

Используют его и в сложных поисковых системах, которые выполняют не только поиск, но и играют роль некоторой "отвечающей системы" – программного комплекса, который умеет извлекать информацию из большой выборки текстовых файлов и баз данных, а затем вести диалог с

пользователем, отвечая, в обычном понимании этого слова, на его вопросы.

Также Prolog используют при написании новых специфичных языков программирования. Например, функциональный язык Erland постороен на основе Prolog. По сути, Erland является усовершенствованием Prolog для некоторых специфических целей, связанных с задачами реального времени.

В данное время Prolog, несмотря на неоднократные пессимистические прогнозы, продолжает развиваться в разных странах, включая в себя новые технологии и концепции. К ним относятся и парадигмы императивного программирования, при котором процесс вычисления описывают в виде инструкций, изменяющих состояние программы.

Одно из направлений развития языка, в том числе и в России, реализует концепцию интеллектуальных агентов. Под этим термином понимаются разумные сущности, наблюдающие за окружающей средой и действующие в ней. При этом их поведение рационально в том смысле, что они способны к пониманию, а их действия направлены на достижение какой-либо цели. Такой агент может быть как роботом, так и встроенной программой. Об интеллектуальности агента можно говорить, если он взаимодействует с окружающей средой примерно так же, как действовал бы человек.

В качестве примера, иллюстрирующего области использования Prolog можно сослаться на ряд проектов, реализованных датской компанией PDC, домашняя страница которой приведена на рис.1. Вместе с разработкой собственно системы программирования Visual Prolog, компания PDC на основе Visual Prolog разработала и внедрила ряд интеллектуальных продуктов и решений. Основными среди них являются:

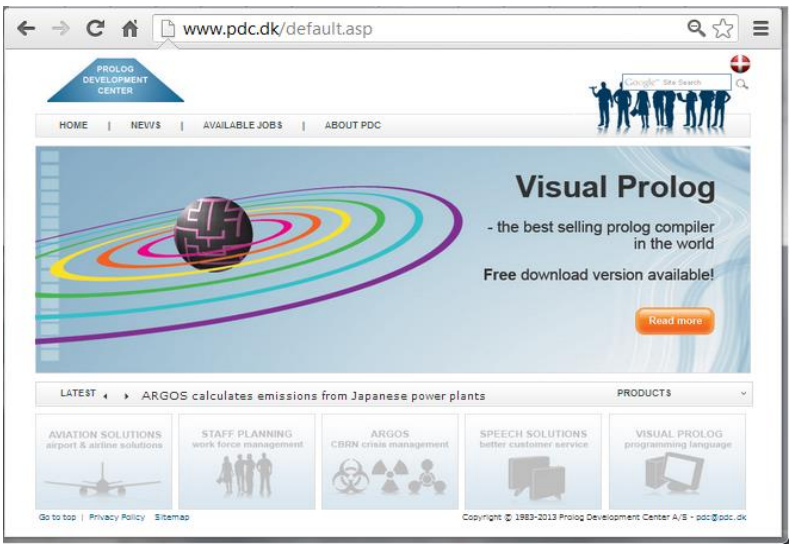


Рис. 1. Домашняя страница компании PDC (www.pdc.dk)

• PDC SCORE и другие – это ИТ-решения в области авиации, которые предназначены для планирования и составления расписаний для бизнес-авиации, а также для бизнес-планирования работы авиакомпаний, аэропортов и наземного обслуживания самолетов. Около 20% мировых авиаперевозок координируется с использованием PDC SCORE. Более 40 международных авиакомпаний и 280 аэропортов используют эти решения.

• ARGOS – информационная система управления и принятия решений в кризисных ситуациях.

• PDC StaffPlan – это ИТ-решение для экономически эффективного кадрового планирования и управления ресурсами. Используется уже около 20 лет в крупных и средних организациях и компаниях в сфере здравоохранения, розничной торговли, аэропортах и промышленности.

• Dictus – это программа распознования и синтеза речи, позволяющая превращать речь в печатный текст с 98% точностью, управлять компьютером с помощью голосовых команд, использовать голосовую передачу и прием SMS-сообщений и т.п.

В этих проектах используется разработанная в PDC технология, которая позволяет применять правила и методы искусственного интеллекта для оптимальной и эффективной поддержки принятия решений. Основу этой технологии составляет Visual Prolog. Сегодня, по мнению PDC, Visual Prolog представляет собой мощный и безопасный язык программирования сочетающий лучшие возможности логического, функционального и объектно-ориентированного программирования.

Если говорить о менее мощных реализациях Prolog, поддерживающих в основном эдинбургский стандарт ISO Prolog, то, вне всякого сомнения, программа на языке Prolog может стать "мозгом" различных чрезвычайно интересных приложений. Но при этом выполнение интерфейсных функций (ввод-вывод, преобразование форматов данных, диалог с пользователем и т.д.) целесообразно возложить на другие языки программирования.

В настоящее время подобная практика разработки прикладных программных комплексов на основе Prolog находит широкое распространение. Этому способствуют и современные реализации языка Prolog, которые хорошо сочетаются с Java, Delphi, C#, C++ и др. Это позволяет реализовать в разрабатываемых приложениях и процедурные, то есть алгоритмические функции, и методы решения декларативных, интеллектуальных задач.

Современные реализации языка Prolog

Принятие стандарта языка Prolog стало стимулом для создания многих независимых реализаций этого языка, как коммерческих, так и бесплатно

предоставляемых широкому кругу пользователей. Постоянно обновляемый обзор новейших реализаций Prolog можно найти в Интернете по адресу:

```
http;//dmoz.org/Computers/Programming/Languages/Prolog/Implementations/
```

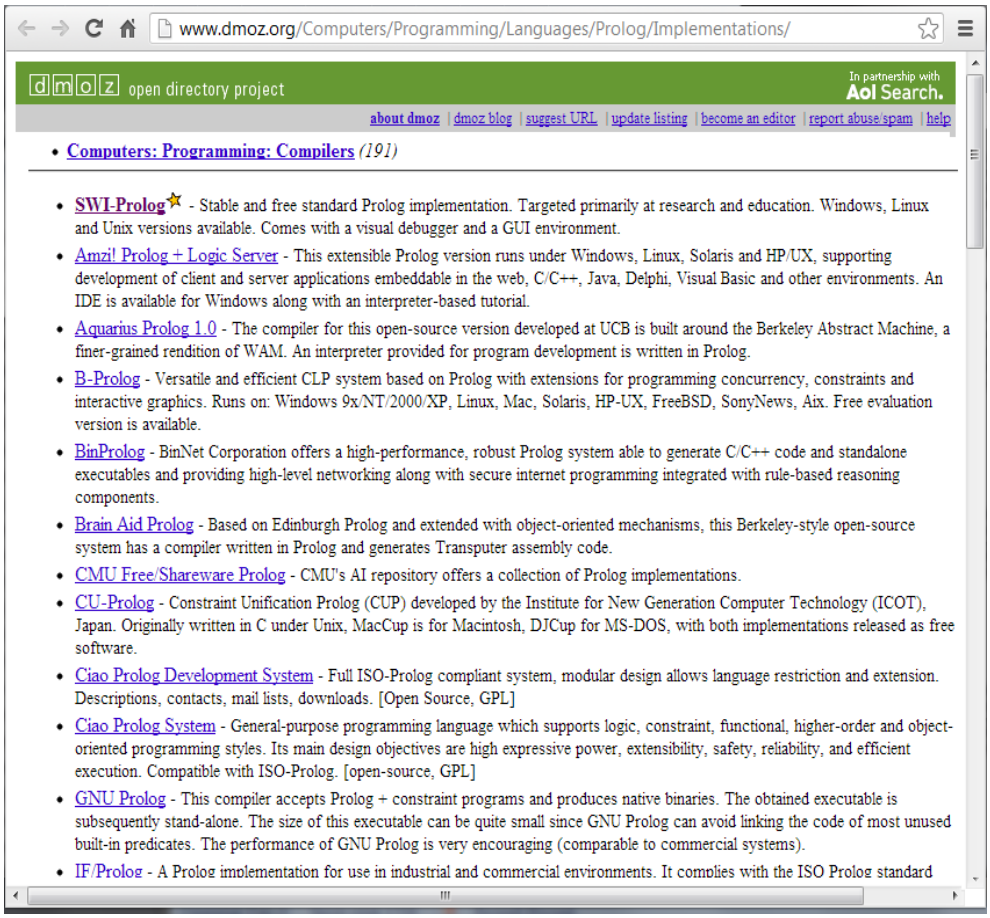


Рис. 2. Список современных дистрибутивов языка Prolog

На момент написания этого обзора этот список (рис. 2) насчитывал 26 реализаций. Среди них присутствуют как коммерческие версии, так и свободно распространяемые. Кроме того, для многих коммерческих дистрибутивов предусмотрены демонстрационные версии, с лицензией на ограниченный, но достаточно большой период времени, а также персональные выпуски (Personal Edition), которые предоставляются бесплатно для личного пользования.

Таблица 1

Сравнение основных характеристик реализаций Prolog

Платформа		Особенности								Инструмент			Механизм
Наименование	Операционная система	Графика	Компилятор	Unicode	ООП	Управление ОС	Автон. исполнение	Интерфейс к С	Интерфейс к Java	Интерпретатор	Отладчик	Профайлер	Синтаксис Prolog
BProlog	U, W, M		+		+	+	+	+	+	+	+	+	ISO
JIProlog	JVM, A	+		+		через Java	+	через Java	+	+	+		ISO
Ciao	U,W,M		+		+	+	+	+	+	+	+	+	ISO_R.
GNU Prolog	U,W,M		+			+	+	+		+	+		ISO
JLog	JVM	+	+						+	+			ISO
JScriptLog	Браузер									+			ISO
jTrolog	JVM			+					+	+	+		ISO
LPA-Prolog	W	+	+	+	+	+	+	+	+	+	+	+	EP_R
Open Prolog	Mac OS										+		
Poplog	L, U, W	+	+			+	+	+		+	+		EP_R
SICStus Prolog	U,L,W,M	+	+	+	+	+	+	+	+	+	+	+	ISO
SWI-Prolog	U,L,W,M	+	+	+		+	+	+	+	+	+	+	ISO, EP
tuProlog	JVM, A	+		+				+	+	+	+		ISO
Visual Prolog	W	+	+	+	+	+	+	+			+	+	
XSB Prolog	L,W,S,M		+			+	+	+	+	+	+	+	ISO
ПЕА-Prolog	L,W,S,M		+	+		+	+	+	+	+	+		EP, ISO

Условные обозначения: U – ОС Unix , W – ОС Windows, M -Mac OS X, L - ОС Linux, S – ОС Solaris, A – ОС Android, JVM - Java Virtual Machine, ISO – ISO Prolog, ISO_R - ISO Prolog с расширениями, EP - Edinburgh Prolog, EP_R - Edinburgh Prolog с расширениями

Достаточно полный обзор основных современных реализаций языка Prolog представлен в Интернете на портале Википедия по адресу: http://en.wikipedia.org/wiki/Comparison_of_Prolog_implementations. Контент этого ресурса позволяет сравнить основные характеристики различных реализаций Prolog (табл. 1) и познакомиться с возможностью поддержки веб-технологий (табл. 2) этими реализациями.

Таблица 2

Поддержка веб-технологий в разных реализацях Prolog

Наименование	Усл. компиляция	Сокеты	Многопоточность	HTTP клиент	HTTP сервер	HTML парсер	Адрес сайта в Интернете
B-Prolog							www.probp.com/
Ciao	+	+	+	+		+	www.ciao-lang.org/
GNU Prolog		+					www.gprolog.org
Jekejeke Prolog		+	+	+	+		www.jekejeke.ch
LPA-Prolog		+		+	+	+	www.lpa.co.uk/win.htm
SICStus Prolog	+	+	+				www.sics.se/projects/sicstus-prolog-leading-prolog-technology
SWI-Prolog	+	+	+	+	+	+	www.swi-prolog.org/
Visual Prolog	+	+	+	+		+	www.visual-prolog.com/
XSB		+	+	+			xsb.sourceforge.net
YAP-Prolog	+	+	+				www.dcc.fc.up.pt/~vsc/Yap/

Наиболее мощные реализации Prolog позволяют использовать средства статистического анализа, а также средства логического программирования в ограничениях (Constraint Logic Programming - CLP). Средства CLP показали себя на практике как исключительно гибкий инструмент для решения задач принятия решений, составления расписаний, планирования материально-технического снабжения и решения комбинаторных задач.

За последние 10-15 лет программирование в ограничениях прошло путь от научной идеи до мощной парадигмы программировнаия и считается в настоящее время одним из стратегических направлений информатики. Наиболее популярные реализации, включающими в свой состав CLP – это B-Prolog, CHIP V5, Ciao Prolog, ECLiPSe, SICStus, GNU Prolog, YAP Prolog.

Так, например, в релиз B-Prolog v.7.8 добавлено моделирование задач линейного и смешанного целочисленного программирования, а также SAT-решатель. В любой реализации Prolog, поддерживающей CLP,

отношения между переменными указываются в форме ограничений. Ограничения могут быть: вида, используемого в задачах удовлетворения ограничений (например, «A или B истинно»), вида, решаемых симплексным алгоритмом (например, «x ≤ 5») и ряд других видов.

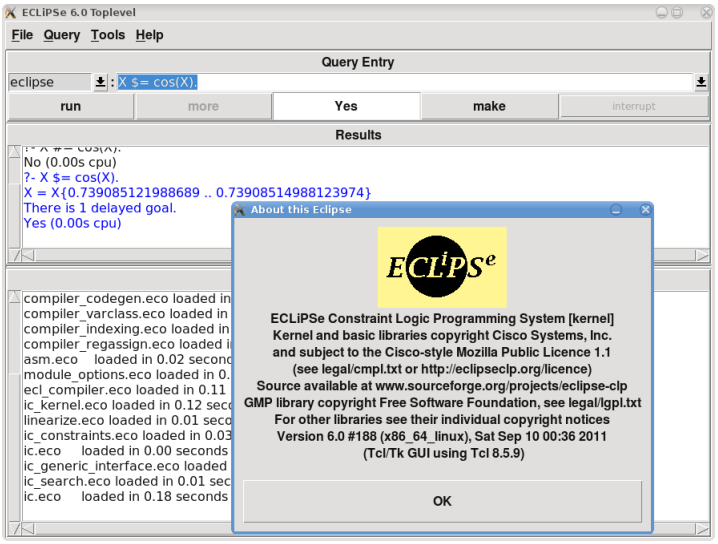


Рис. 3. TkECLiPSe - стандартная GUI-оболочка для ECLiPSe, версия 6.0

Одним из ярких представителей таких систем является ECLiPSe (ECLiPSe Constraint Logic Programming System), интегрирующая различные расширения логической парадигмы программирования, в особенности логического программирования с ограничениями (рис.3). Ее ядро является эффективной реализацией Edinburgh Prolog. В основе лежит инкрементальный компилятор исходных кодов в коды виртуальной машины. Система написана на Prolog и C.

Наряду с реализацией технологии CLP, некоторые версии языка Prolog включают в свой состав и ряд дополнительных расширений. Таких, как:

- Запоминание результатов (tabling) – это техника, которая известна в функциональном программировании как мемоизация (memoiazation). Она позволяет освобождить пользователя от ручного хранения промежуточных результатов.

- Специально построенные базы данных (triplestore), для хранения и поиска триплетов «субъект - предикат - объект». Так, например, утверждение «Небо голубого цвета» будет представлено: «небо» (субъект), «имеет цвет» (предикат) и «голубой» (объект). В отличие от реляционных баз данных, triplestore оптимизирована для хранения и поиска триплетов, который выполняется с помощью своего языка запросов. В дополнение к

запросам, триплеты обычно можно импортировать/экспортировать с помощью Resource Description Framework (RDF) из других форматов.

Отдельно следует отметить наличие в настоящее время реализаций Prolog для мобильных устройств. К их числу относятся ProLog for IPhone 1.0, а также Prolog Mobile 9.5 от Meridian Project Systems, Inc. и tuProlog for Android (рис. 4), JIProlog для J2ME (MIDP 2.0, CLDC 1.0) для мобильных телефонов с поддержкой Java и ряд других.

Рис.4. Скриншоты tuProlog for Android и Prolog Mobile 9.5 на iPad

JIProlog (Java Internet Prolog) — кроссплатформенный интерпретатор языка Prolog, который интегрирует языки Prolog и Java, позволяя вызывать предикаты Prolog из Java и методы Java из Prolog. JIProlog спроектирован так, что он может работать с любой версией Java 1.1 и выше.

На сайте JIProlog есть режим demo (http://www.jiprolog.com/demo.aspx), в котором загружается Java-плагин консоли Prolog. Он позволяет вводить и исследовать небольшие Prolog-задачи прямо из веб-браузера (рис. 5).

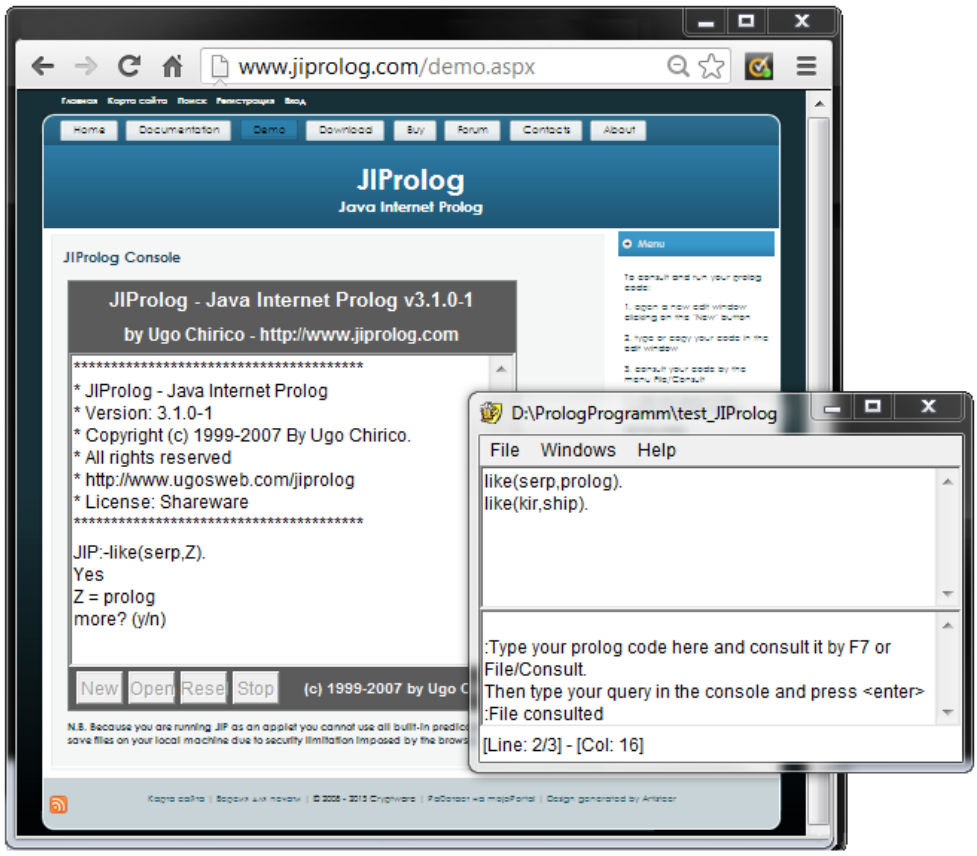


Рис. 5. Редактор и консоль JIProlog в веб-браузере

В это же время разрабатываются множество коммерческих реализаций Prolog практически для всех типов компьютеров. Две коммерческие системы Quintus Prolog (http://www.sics.se/quintus/) и SICStus Prolog (http://www.sics.se/sicstus/) развиваются и поддерживаются SICS (Swedish Institute of Computer Science). Это профессиональные среды разработки с большим набором инструментов и очень обширными библиотеками.

Одним из самых интересных продуктов является Visual Prolog [3,4,5]. Это полнофункциональная среда программирования, которая предлагает полный набор средств, необходимых для разработки критически важных задач, коммерческого класса приложений. Несмотря на своё название это - не реализация Пролога, а совершенно иной язык со строгим контролем типов. Visual Prolog возник не на пустом месте, а является развитием и наследником таких языков программирования как PDC Prolog и Turbo Prolog.

Следует обратить особое внимание на скриншот сайта приведенного на рис. 2. В представленном там списке все реализации Prolog приведены в алфавитном порядке, и только одна из них выбивается из этого ряда. Она помещена в начало списка и отмечена звездочкой.

Это SWI-Prolog – стабильная и бесплатная стандартная реализация языка Prolog, которая ориентирована в первую очередь на научные

исследования и образование, но возможно и ее коммерческое использование. Для SWI-Prolog доступны версии под Windows, Linux и Unix, а также открытий исходный код.

Если на сайте SWI-Prolog обратиться к статистике закачек этого продукта, то можно обнаружить, что в период с 2009 по 2012 годы количество скачиваний в среднем более 10 тыс. в месяц (рис. 6).

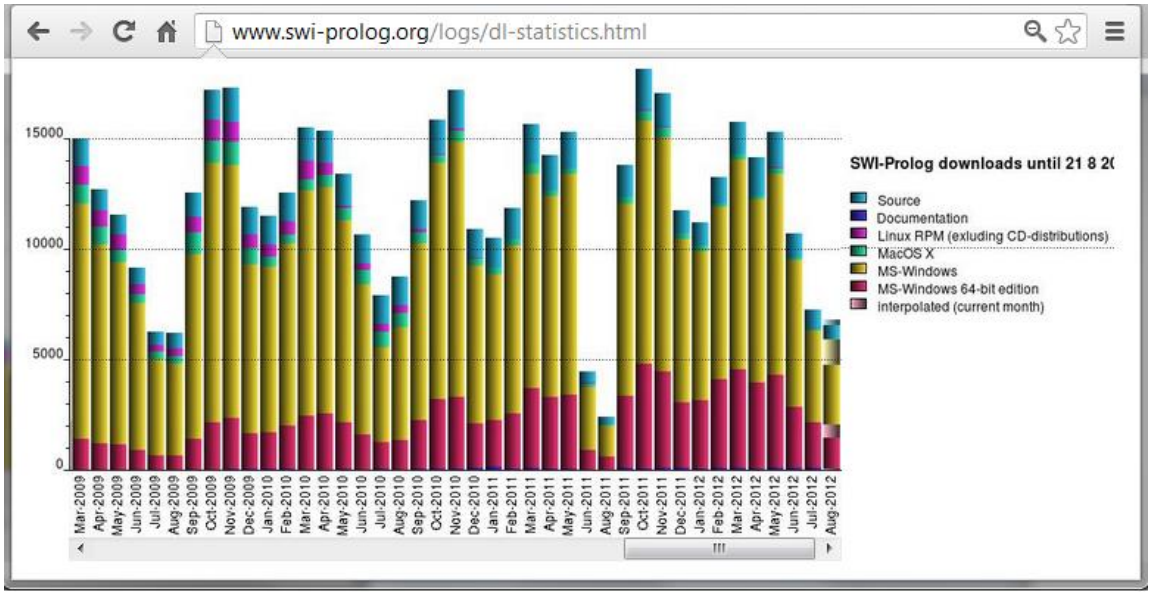


Рис. 6. Статистика закачек дистрибутива с сайта SWI-Prolog

Это говорит о достаточной популярности этого продукта в мире. А если еще сравнить эти значения с количеством закачек в 2001-2005 годах, то из этого сравнения будет очевидно, что количество пользователей SWI-Prolog за последнее время увеличилось почти в 2 раза.

И это в то самое время, когда слышны высказывания о том, что Prolog уже умер. Может те, кто это говорит, просто живут в другом мире – мире дизайнов веб-сайтов и бухгалтерских программ. Но ведь кто-то должен решать и сложные задачи контроля, диагностики, управления и принятия решений в технических, биологических и социальных системах.

Разработка SWI-Prolog проводится в университете города Амстердам, Нидерланды с 1987 года. Его создателем и основным разработчиком является Jan Wielemaker. Название SWI – это аббревиатура от голландского *Sociaal Wetenschappelijke Informatica* («Social Science Informatics»), первоначального названия научной группы университета, в которой работает Jan Wielemaker. Сейчас название этой группы сменилось на новое название HCS (Human Computer Studies).

SWI-Prolog почти как все реализации в основном следует знаменитой "эдинбургской версии", но содержит частично реализованные особенности

ISO Prolog. К числу основных особенностей последних версий SWI-Prolog следует отнести:

- Богатые библиотеки предикатов.
- Возможность написания логических модулей для веб-приложений.
- Поддержка GUI для XPCE и PreEmacs.
- Встроенная командная строка.
- Работа с файлами.

SWI-Prolog содержит развитые средства разработчика, включая интегрированную среду разработки (англ. Integrated Development Environment - IDE) с графическими отладчиком и профилировщиком, и обширную документацию [6,7]. Кроме базовых функций языка, платформа реализует многопоточность, юнит-тестирование, GUI, интерфейс к языку программирования Java, ODBC и т. д. Следует отметить, что SWI-Prolog имет встроенный собственный веб-сервер и работает на Unix, Windows, Macintosh и Linux платформах.

Библиографический список

4. *Хабаров, С.П.* Интеллектуальные информационные системы. PROLOG – язык разработки интеллектуальных и экспертных систем: учебное пособие. — СПб.: СПбГЛТУ, 2013. — 140 с.

5. *Братко, Иван.* Алгоритмы искусственного интеллекта на языке PROLOG, 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2004. — 640 с. :

6. Visual Prolog. Техническая, справочная и обучающая информация от разработчиков (рус.) [Электронный ресурс]. – Режим доступа: открытый ресурс; постоянный адрес в Интернет: – http:://wikiru.visual-prolog.com/index.php

7. *Costa Eduardo*. Visual Prolog для чайников [Электронный ресурс]. – Режим доступа: открытый ресурс; постоянный адрес в Интернет: http://wikiru.visual-prolog.com/index.php?title=Visual_Prolog_for_ Tyros – Загл. с экрана

8. *Thomas W. De Boer*. A Beginners' Guide to Visual Prolog 7.2 [Электронный ресурс]. – Режим доступа: открытый ресурс; постоянный адрес в Интернет: – http://download.pdc.dk/vip/72/books /deBoer /VisualPrologBeginners.pdf – Загл. с экрана.

9. SWI-Prolog documentation [Электронный ресурс]. – Режим доступа: открытый ресурс; постоянный адрес в Интернет: – http://www.swi-prolog.org/pldoc/index.html – Загл. с экрана.

10. Wielemaker Jan, Anjewierden Anjo. Programming in XPCE/Prolog [Электронный ресурс]. – Режим доступа: открытый ресурс; постоянный

адрес в Интернет: http://www.swi-prolog.org/packages/ xpce/UserGuide/ – Загл. с экрана.

М.А.Шубина, кандидат технических наук, доцент

НОВЫЕ ВОЗМОЖНОСТИ ИДЕНТИФИКАЦИИ ЗЕМНЫХ ОБЪЕКТОВ ПО ИЗОБРАЖЕНИЯМ КА LANDSAT-8

Спутниковая информация является объективным средством получения актуальной информации о состоянии поверхности Земли, которая необходима для решения многих хозяйственных задач.

Одним из наиболее популярных спутниковых средств мониторинга Земли является система получения изображений среднего пространственного разрешения Landsat /1/. Мозаика многократных изображений Земли, полученных аппаратами именно этой серии, составили изображение всей поверхности Земли, представленные на известном открытом сайте Google Earth. Многоканальный Landsat обеспечивает среднее пространственное разрешение около 30 м в семи каналах видимого спектра и 15 м в панхроматическом канале. В последние годы изображения Landsat ряда регионов Земли на Google Earth покрываются более дорогими изображениями более высокого пространственного разрешения, а система Landsat стала развиваться в направлении увеличения количества каналов в расширенном спектральном диапазоне, представляющих новую дополнительную информацию.

После выключения Landsat 5 в начале 2013 года, Landsat 7 остался единственным действующим спутником программы Landsat, поставляющим информацию с известными искажениями. Следующий последний аппарат Landsat 8 - восьмой в рамках программы Landsat.

Изначально он назывался *Landsat Data Continuity Mission* (LDCM), создан совместно NASA и USGS. Выведен на орбиту 11 февраля 2013 года. Спутник Landsat 8 продолжает получение данных для программы, используя два набора инструментов, Operational Land Imager (OLI) и Thermal InfraRed Sensor (TIRS). Первый набор получает изображения в 9 диапазонах видимого света и ближнего ИК, второй набор — в 2 диапазонах дальнего (теплового) ИК.

Landsat 8 получает изображения в видимом диапазоне волн, в ближнем ИК и в дальнем ИК, с разрешением снимков от 15 до 100 метров на точку. Производится съемка суши и полярных регионов. В сутки снимается порядка 400 сцен (у предыдущего Landsat-7 было всего 250 сцен в день). По сравнению с Landsat 7 сенсоры OLI и TIRS имеют более

высокое отношение сигнал-шум (SNR) и позволяют снимать до 12 бит на точку. В OLI используется схема *pushbroom*, тогда как в более ранних аппаратах Landsat использовалась схема *whiskbroom*. В схеме pushbroom используются длинные линейные массивы фотодатчиков, снимающие сразу всю ширину поля зрения спутника — 185 километров, тогда как в whiskbroom использовались небольшое количество фотоприемников и дополнительное сканирующее зеркало.

Новая схема требует применения более 6,5 тысяч детекторов для каждого спектрального канала (и 13 тысяч для панхроматического), однако имеет более высокое время экспонирования (4 мс вместо 10 мкс на ETM+) и как следствие большую чувствительность.

OLI работает в 9 спектральных диапазонах, семь из которых близки к тем, которые использовались в более ранних инструментах Thematic Mapper (TM) и Enhanced Thematic Mapper Plus (ETM+) с предыдущих спутников Landsat, за счет чего обеспечивается преемственность и совместимость с ранее накопленным массивом данных Landsat. Добавлено два новых диапазона, канал 1 (темно-синий и фиолетовый) для изучения прибрежных вод и аэрозолей и канал 9 (ближний ИК) для упрощения поиска облаков на снимках.

Инструмент TIRS использует тот же принцип получения изображений *pushbroom*, что и OLI, и также имеет полосу обзора в 185 километров. Получение изображений происходит в двух каналах, 10 и 11, которые, совместно, работают в том же диапазоне, что и канал TIR на более ранних спутниках программы Landsat.

Для питания используются раскрывающиеся солнечные батареи и бортовой аккумулятор NiH2 на 125 ампер-часов. Для хранения данных установлен твердотельный накопитель (флеш-память) объемом 3.14 терабит (порядка 0,4 терабайт). Спутник рассчитан на срок активного существования в 5 лет, однако запас топлива позволяет использовать его до 10 лет.

Основные научные задачи Landsat 8:

- сбор и сохранение много спектральных изображений среднего разрешения (30 метров на точку) в течение не менее чем 5 лет;
- сохранение геометрии, калибровки, покрытия, спектральных характеристик, качества изображений и доступности данных на уровне, аналогичном предыдущим спутникам программы Landsat;
- бесплатное распространение изображений, полученных с помощью Landsat 8

Основные параметры продукции Landsat 8:

Орбита - солнечно-синхронная, приполярная, наклонение - 98,2°, интервал повторения -16 суток.

- Уровень обработки: 1T (коррекция рельефа).
- Формат изображений: GeoTIFF.
- Размер пикселя: 15 метров/30 метров/100 метров (панхроматический канал/ мультиспектральный канал/ дальний ИК).
- Проекция: UTM, также полярная стереографическая для Антарктиды.
- Система координат: WGS 84.
- Точность позиционирования:
  - OLI: КВО 12 метров (90 %)
  - TIRS: КВО 41 метр (90 %)

Сравнительные характеристики **Landsat 7 и 8** приведены в табл. 1.

Таблица 1

Сравнение каналов Landsat 7 и 8				
Каналы	**Landsat-8 Длины волн мкм**	**Разреше-ние м**	**Landsat7 Длины волн, мкм**	**Разреше ние м**
1 - Побережья и аэро-золи (Coastal Aerosol, New Deep Blue)	0.433 — 453	30		
2 - Синий (Blue)	0.450 — 0.515	30	1- Visible 0.45 - 0.52	30
3 - Зеленый (Green)	0.525 — 0.600	30	2- Visible 0.52 - 0.60	30
4 - Красный (Red)	0.630 — 0.680	30	3 -Visible 0.63 - 0.69	30
5 - Ближний ИК (Near Infrared, NIR)	0.845 — 0.885	30	4 -Near-Infrared 0.77 - 0.90	30
6 - Ближний ИК (Short Wavelength Infrared, SWIR 2)	1.560 — 1.660	30	5 -Near-Infrared 1.55 - 1.75	30
7 - Ближний ИК (Short Wavelength Infrared, SWIR 3)	2.100 — 2.300	30	7 -Mid-Infrared 2.08 - 2.35	30
8 - Панхроматический (Panchromatic, PAN)	0.500 — 0.680	15	8 -PAN	15
9 - Перистые облака (Cirrus, SWIR)	1.360 — 1.390	30		
Thermal InfraRed Sensor (TIRS)			Thermal	

10 - Дальний ИК (Long Wavelength Infrared, TIR1)	10.30 — 11.30	100	6 –1 10.40 - 2.50	60 Low Gain
11 - Дальний ИК (Long Wavelength Infrared, TIR2)	11.50 — 12.50	100	6-2 10.40 - 2.50	60 High Gain

Цветные изображения RGB могут быть представлены в зависимости от целей решения задач комбинацией трех каналов. Естественным цветам соответствует (Operational Land Imager, каналы 2 (синий), 3 (зеленый) и 4 (красный). Искусственные цвета (OLI) - каналы 3 (зеленый), 5 (ближний ИК 1) и 7 (ближний ИК 2) изображаются соответственно как синий, зеленый и красный. Удачный выбор каналов при построении цветных изображений может помочь более качественному обнаружению и идентификации природных объектов.

Как видно из таблицы, появился канал, позволяющий выделять аэрозоли, и был расширен диапазон инфракрасного излучения.

Добавление каналов важно для решения задач лесного хозяйства, например, это облегчает задачу различения облачности и дымов от пожаров.

Пожары являются проблемой глобального значения, оказывающей влияние на характер и распространение растительности. Возгорания уничтожают флору и фауну, а также являются причиной хозяйственного и экологического ущерба. Выбросы аэрозолей в атмосферу при крупном пожаре соизмеримы с вулканическим извержением. Степень воспламенения природных ландшафтов имеет свою специфику в разных районах земного шара.

Ежегодно в мире регистрируется до 400 тыс. лесных пожаров, в результате которых повреждается около 0,5 % общей площади лесов, при этом в атмосферу выбрасываются миллионы тонн продуктов сгорания. Каждый год на территории России происходит более 30 тысяч возгораний, в результате которых повреждаются лесные массивы площадью 2 – 3млн. га.

Пожароопасный сезон (по фактической горимости) наступает по мере таяния снега и просыхания поверхности почвы и напочвенного покрова.

Пример изображения зоны пожаров на территории Дальневосточного федерального округа, одного из наиболее

подверженного пожарам района России, представлены на рис. 1, 2. Изображения получены с сайта USGS /2/.

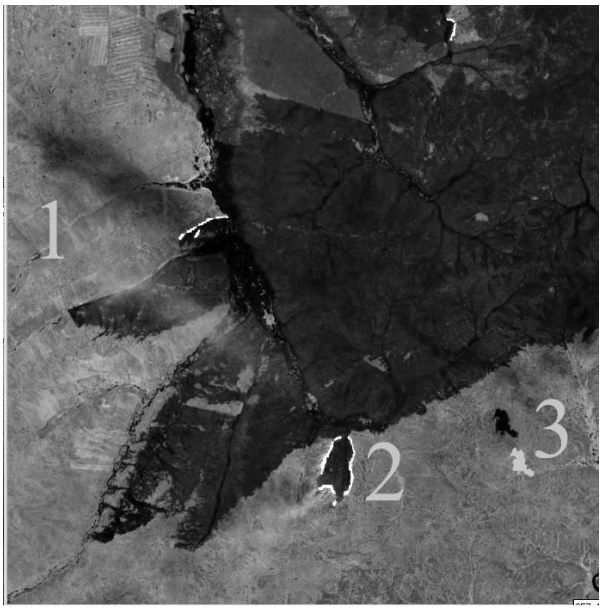


Рис. 1. Изображение Landsat 8 участка долины реки Зея, позволяющее выделить дымы (светлая полоса и ее тень 1), кромки пожаров (белая кайма 2), облака (облако и его тень 3), полученные при сочетании каналов видимого и ближнего ИК диапазонов

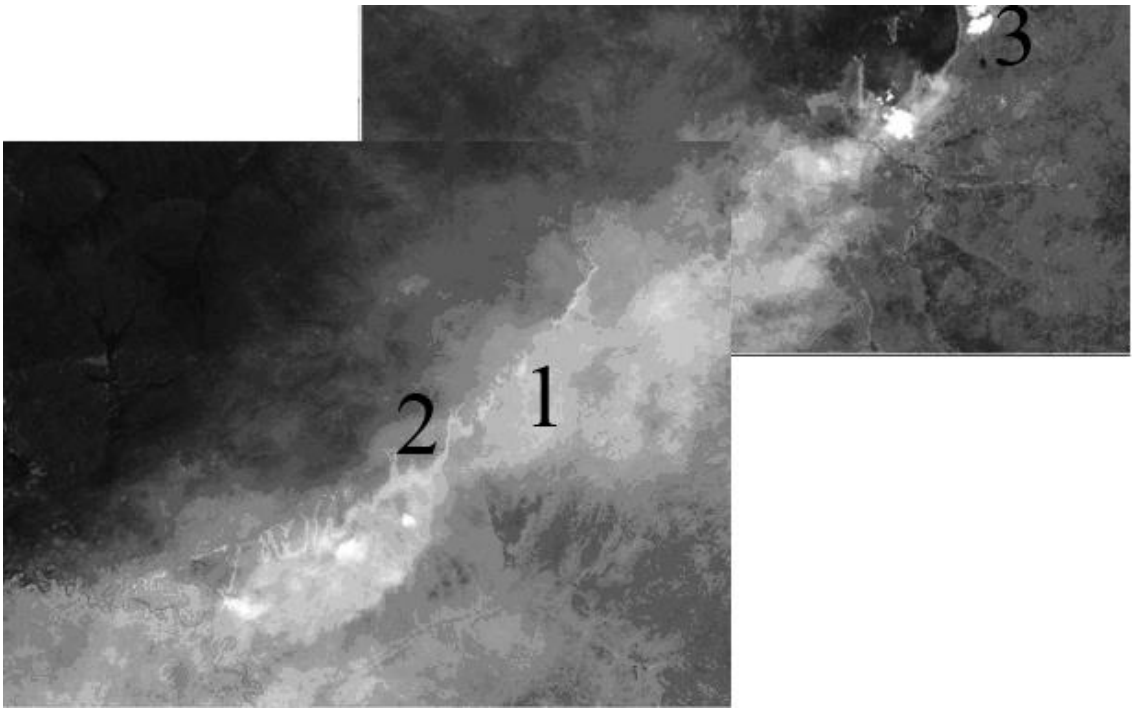


Рис.2. Изображение Landsat 8 участка долины реки Зея, позволяющее выделить дымы 1, кромки пожаров 2, облака 3, полученные при сочетании каналов видимого и дальнего ИК-диапазона

Таким образом, использование каналов ближнего и дальнего инфракрасных диапазонов Landsat-8 позволяет более надежно выделить фронты пожаров и отделить изображение дымов от облачности.

Библиографический список

1. Официальный сайт www.nasa.gov
2. Официальный сайт www.usgs.gov

СОДЕРЖАНИЕ