Санкт-Петербургский государственный университет
Математико-механический факультет

Лаборатория системного программирования и информационных технологий
(СПРИНТ, СПбГУ – Intel)

К. С. Амелин, Н. О. Амелина, О. Н. Граничин, В. И. Кияев

Разработка приложений для мобильных интеллектуальных систем на платформе Intel Atom

Санкт-Петербург
2012

УДК 004.89

Печатается по решению Редакционно-издательского совета Математико-механического факультета Санкт-Петербургского государственного университета

Р е ц е н з е н т ы:

д-р физ.-мат. наук, профессор А.Н.Терехов (С.- Петерб. гос. ун-т)
менеджер по стратегическому развитию Intel И.О.Одинцов

А м е л и н К. С., А м е л и н а Н. О., Г р а н и ч и н О. Н., К и я е в В. И.

А61 Разработка приложений для мобильных интеллектуальных систем на платформе Intel Atom. – СПб.: Издательство ВВМ, 2012. – 211 с.

ISBN 978-5-9651-0713-1

В книге представлены новые результаты авторов по разработке приложений для мобильных интеллектуальных систем на платформе Intel Atom, прочитанные авторами для студентов Математико-механического факультета СПбГУ, стажеров Лаборатории системного программирования и информационных технологий (СПРИНТ, СПбГУ- Intel) и слушателей молодежных образовательных школ вузов Российской Федерации. Представлены также полезные сведения, касающиеся создания и управления интеллектуальной собственностью. Материал сопровождается описанием иллюстративных лабораторных работ для самостоятельного изучения.

Книга предназначена для студентов, аспирантов и преподавателей соответствующих специальностей, а также может быть полезна широкому кругу читателей.

© К.С. Амелин, Н.О. Амелина, 2012
© О.Н. Граничин, В.И. Кияев, 2012

ISBN 978-5-9651-0713-1

Содержание

Введение *5*

1. От встроенных систем к системам интеллектуальным *7*
1.1. Кибернетическая революция 7
1.2. Информация, сигналы, данные, знания и управление 12
1.3. Соединение процессов обработки данных и управления в интеллектуальных встроенных устройствах 22
1.4. Компьютерный континуум Intel и процессор Intel®Atom 26
1.5. Лабораторная работа № 1 «Установка ПО для разработки и тестирования приложений» 30
1.6. Выводы 45
1.7. Контрольные вопросы 46
Список литературы 49

2. Мультиагентные технологии *51*
2.1. Введение 51
2.2. Мультиагентный подход 52
2.3. Мультиагентные системы 55
2.4. Агенты 57
2.7. Лабораторная работа № 2 «Разработка приложения для БПЛА» 64
2.8. Выводы 71
2.9. Контрольные вопросы 72
Список литературы 75

3. Разработка мультиагентных систем *77*
3.1. Мультиагентные системы 77
3.2. Современные международные стандарты создания агентов и платформы МАС 81
3.3. Платформы для разработки МАС 85
3.4. Области применения 86
3.5. Лабораторная работа № 3 «Знакомство с JADE» 102
3.6. Выводы 111
3.7. Контрольные вопросы 112
Список литературы 114

4. Проект «Мультиагентная система для БПЛА» *116*
4.1. Введение 116
4.2. Мультиагентный подход и трёхуровневая система управления БПЛА 122

4.3. БПЛА-агент для автономной группы 126
4.4. Алгоритмы групповой работы сети БПЛА 129
4.5. Программирование бортовых микрокомпьютеров БПЛА..... 131
4.6. Лабораторная работа № 4 «Пример мультиагентного приложения с JADE-агентом на платформе Android»..... 136
4.7. Выводы 142
4.8. Контрольные вопросы..... 142
Список литературы 148

5. Мультиагентное управление 150
5.1. Введение..... 150
5.2. Цели управления 150
5.3. Балансировка загрузки узлов децентрализованной вычислительной сети при неполной информации..... 152
5.4. Задача консенсуса на графах..... 154
5.5. Задача балансировки загруженности сети 157
5.6. Протоколы консенсуса 160
5.7. Лабораторная работа № 5 «Имитационное моделирование алгоритма локального голосования для балансировки загрузки узлов вычислительной сети»..... 163
5.8. Выводы 169
5.9. Контрольные вопросы..... 170
Список литературы 173

6. Создание объектов интеллектуальной собственности и авторское право..... 178
6.1. Введение..... 178
6.2. Общие принципы и основные понятия в сфере интеллектуальной собственности 180
6.3. Законодательство в области права интеллектуальной собственности..... 185
6.4. Юридическая природа, содержание и защита исключительных прав..... 194
6.5. Понятие и принципы патентного права..... 199
6.6. Способы коммерциализации объектов интеллектуальной собственности..... 205
6.7. Базовые принципы управления ОИС..... 209
Список литературы 217

Заключение..... 219

Введение

Олимпийские спортивные призывы: «Быстрее, Выше, Сильнее» в компьютерной области трансформировались в лозунг: «Быстрее, Мощнее, Миниатюрнее»! На самом деле эти противоречивые цели стыкуются в одном: технологии приближаются к созданию «мобильного» искусственного интеллекта – уже скоро умные встроенные миниатюрные вычислительные устройства смогут выполнять те функции, которые несколько лет назад и «не снились» пользователям больших компьютеров. Такие системы немыслимы без современных методов распараллеливания вычислений, а их моделирование и работа в кооперации – без мощных вычислительных устройств, роль и место которых в обществе преобразятся.

Основу книги составили новые результаты авторов, представленные в курсах лекций и мастер-классов по разработке приложений для мобильных интеллектуальных устройств, прочитанных авторами летом и осенью 2012 года для слушателей молодежных школ Intel в Санкт-Петербурге и Волгограде.

Главная цель книги – дать общее представление о некоторых современных тенденциях в процессах разработки приложений для мобильных интеллектуальных систем на платформе Intel Atom.

Книга состоит из шести глав. Первая глава носит общий характер: перспективы развития средств вычислительной техники, переход к интеллектуальным встроенным системам, компьютерный континуум Intel и сведения о процессорах Intel-Atom. Вторая – о современных мультиагентных технологиях, третья – об общих принципах взаимодействия агентов и мультиагентных платформах. В четвертой главе описывается проект авторов по созданию мультиагентной системы управления группой БПЛА (беспилотных летательных аппаратов) и возможная трехуровневая архитектура системы управления мобильным интеллектуальным автономным устройством; в пятой главе детально разобран характерный пример использования мультиагентной парадигмы при решении задачи о ба-

лансировке загрузки узлов децентрализованной вычислительной сети. Шестая глава посвящена особенностям авторского права и патентования разработок интеллектуальных встроенных систем. В лабораторных работах лекционный материал иллюстрируется на работоспособных изолированных примерах приложений.

Общий план и структуру курса разработал О.Н.Граничин, который написал первую и заключительную главы, вторая, третья и пятая главы составлены Н.О. Амелиной; четвертая глава подготовлена К.С.Амелиным; шестая – В.И.Кияевым, который также участвовал в написании первой, второй и третьей глав. Лабораторные работы подготовлены К.С.Амелиным совместно с А.В.Корявко и Н.О.Амелиной.

Дополнительные материалы доступны по ссылке http://www.math.spbu.ru/user/gran/Atom32.

Общее редактирование текстов и подготовку пособия к публикации осуществили О.Н.Граничин и В.И.Кияев. Дизайн обложки выполнен В.И.Кияевым.

Авторы благодарят сотрудников корпорации Intel А.В.Николаева и И.О.Одинцова за инициативу по разработке и созданию курса, активное участие в разработке программы курса, постоянное внимание к продвижению проекта, а также заведующего кафедрой системного программирования СПбГУ, профессора А.Н.Терехова за полезные замечания, которые были учтены при подготовке настоящего издания к печати.

При подготовке пособия были использованы материалы, размещенные на сайтах www.intel.com, www.android.com и др.

Ссылки на программные продукты различных российских и зарубежных фирм не используются в рекламных целях и носят исключительно иллюстративный или справочный характер.

Книга издана при финансовой поддержке Лаборатории СПРИНТ (СПбГУ-Intel).

Книга Норберта Винера 1948 года возвестила
о становлении новой науки КИБЕРНЕТИКИ,
в которой информационно-управленческая связь
в явлениях материального мира
выступает как фундаментальное его свойство.
В. Н. Фомин

1. От встроенных систем к системам интеллектуальным

Переход от простых встроенных (embedded) систем к системам интеллектуальным (intelligence) – закономерная тенденция развития. Компьютерный континуум Intel и процессор Intel® Atom.

1.1. Кибернетическая революция

В истории развития цивилизации произошло несколько *информационных революций* — преобразований общественных отношений из-за кардинальных изменений в сфере обработки информации, информационных технологий. Следствием подобных преобразований являлось всякий раз приобретение человеческим сообществом нового качества.

Начало совпадает по времени с выделения человека из природы, с появления языка, способного оперировать абстрактными понятиями. Первой информационной технологией можно считать способ передачи довольно сложной информации с помощью наскальной «живописи». После изобретения письменности появилась возможность *локального* распространения знаний и сохранения их для передачи последующим поколениям.

Следующий этап связан с изобретением книгопечатания, которое радикальным образом изменило общество, культуру. Книга ста-

ла *универсальным* массовым распространителем и хранителем больших объемов информации.

В эру использования электричества появились телеграф, телефон, радио, телевидение, позволяющие *оперативно* передавать информацию в любые уголки Земли.

Без преувеличения можно сказать, что последние 200 лет были «колыбелью» многих больших и малых революций (рис. 1.1). Промышленная революция XIX-начала XX вв. и Первая мировая война породили волну социальных революций, которые если не перевернули, то основательно потрясли мировые устои. Вторая мировая война и послевоенное развитие экономики, исследования в ядерной и микромолекулярной физике, электронике твердого тела и пограничных явлений, создание первых промышленных вычислительных устройств дали толчок индустриальной революции, которая за четверть века подготовила почву для бурного всплеска развития информационных технологий.

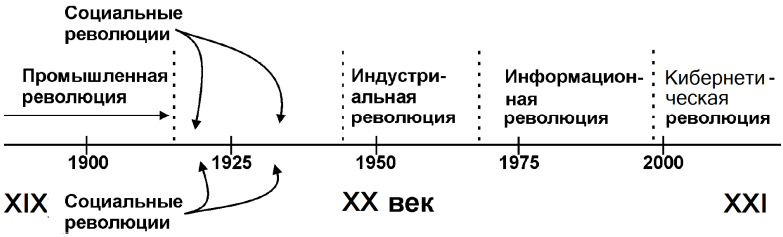


Рис. 1.1. Последовательность революций XIX-XXI веков.

Изобретение компьютера — *универсального, многофункционального, электронного автоматического устройства* для работы с данными и информацией — привело к тому, что компьютерная техника в современном обществе взяла на себя значительную часть работ, связанных с её обработкой, систематизацией и хранением.

В XXI веке мы наблюдаем новый сдвиг в технологиях, который проявляется в разных аспектах (рис. 1.2). Если в последней четверти XX века подчеркивали, что развитие нашего общества находится на *стадии* урбанизации, индустриальной экономики, технологии массовых продуктов, то сейчас её все больше характеризуют глобализация, экономика знаний, цифровые технологии. Ранее *ключевым ресурсом* был капитал, сейчас — знания/информация. В распределении материальных объектов превалировали дороги и транспорт,

сейчас на передовые позиции выходят цифровые сети. Ранее фокус был на региональной сфере, сейчас — на глобальной. Факторами успеха были возможности экономии ресурсов и сокращения затрат при масштабировании производства, теперь таким фактором стала адаптивность — способность быстрого ответа в виде эффективного решения на непредсказуемые изменения (Г. Ржевский, 2012).

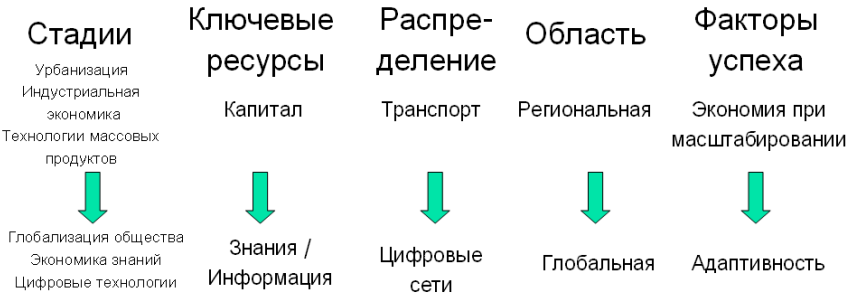


Рис. 1.2. Сдвиги в технологиях XXI века.

При этом за последние полвека стоимость применения цифровых технологий снизилась чрезвычайно сильно в отличие от роста затрат на труд и на использование традиционных механических технологий (рис. 1.3).

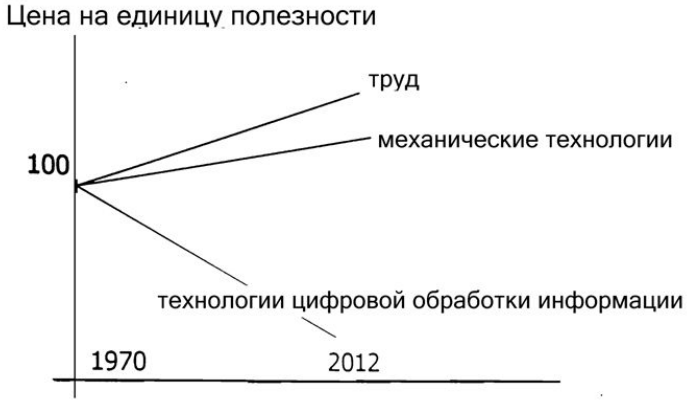


Рис. 1.3. Сравнение стоимости цифровых технологий с механическими.

У людей всегда был интерес к искусственному интеллекту. В конце XX века выделилось два основных подхода к моделированию искусственного интеллекта (*Artificial Intelligence* — AI): машинный интеллект, заключающийся в строгом задании результата функционирования, и искусственный разум, основанный на моделирование внутренней структуры автоматизированной системы, соответствующей строению человеческого мозга.

Но сейчас от теорий человечество делает серьезный шаг к новой реальности: кибернетическому будущему. По многим признакам мы вступили в новую фазу *кибернетической революции*. На диаграмме 1.1 начало этой фазы показано в конце 90х годов XX века. До сих пор в литературе не достаточно изучены причины текущего мирового финансового кризиса, наиболее острая фаза которого пришлась на вторую половину 2008 года. Одним из ключевых технологических факторов стало появление в конце 90х годов XX века первых программ-роботов на финансовых биржах. Непрерывную череду финансовых кризисов XIX и XX вв. в капиталистическом обществе удалось остановить введение в начале 70х годов XX века торговли на биржах «производными» ценными бумагами – опционами. Параллельно с практикой существенные достижения были получены и в теории финансового менеджмента, базирующиеся на гипотезах арбитража (справедливой цены). Теория во многом основывались на эмпирическом опыте и новые механизмы существенно «затруднили» процесс обогащения простыми спекуляциями. Но «программы-роботы» изменили положение дел. В теории традиционно «за скобки» выносили транзакционные издержки (плату за операцию) в связи с их незначительной величиной, но с появлением «программ-роботов» количество операций в единицу времени выросло колоссально и общая «выручка» от них у финансовых учреждений стала существенным фактором сверхприбылей и огромных бонусов руководителей, за которые в итоге заплатили (и продолжают платить) все рядовые граждане развитых стран.

Понять суть происходящих изменений нельзя без переосмысления многих аспектов парадигмы «Что такое вычислительное устройство и что такое вычислительный процесс».

В ближайшее время должны измениться традиционные представления о том, как устроен компьютер, что такое вычислительная

система. Эти процессы принесут изменения в стиль программирования, в то, как будут использоваться вычислительные устройства. Переход к новой парадигме вычислений приведет, наверное, к тому, что архитектура вычислительных устройств «сдвинется» в сторону «*набора одновременно работающих асинхронных моделей взаимодействующих динамических систем (функциональных элементов)*». Среди новых характерных черт будущей парадигмы все более отчетливо проступают следующие: стохастичность, гибридность, асинхронность, кластерность (отсутствие жесткой централизации и динамическая кластеризация на классы связанных моделей).

Естественным этапом на этом переходном пути является вытеснение производства «чистых» процессоров системами на кристалле (System on Chip – SoC). Сейчас корпорация Intel уже начала производство SoC, в которых на чип перенесены контроллеры ввода-вывода, системы обработки видео и ряд других. В «новых» чипах существенно сокращаются объемы вспомогательных обменов в памяти, повышается эффективность, производительность и упрощается конструкция плат. Разработчику архитектуры уже меньше надо думать куда «воткнуть» на плате тот или иной узел. Процессоры, а точнее новые SoC, становятся как бы конечными устройствами, которые можно программировать, не задумываясь о периферии. Это идеальная ситуация, когда программисту не надо думать о целой цепочке посредников, через которые должны пройти его команды и сигналы!

Одним из ключевых моментов современных изменений в ИТ является мобильность! Основная тенденция развития средств вычислительной техники – миниатюризация. Все более хрупкие и миниатюрные вычислительные устройства становятся и более мобильными. Речь идет не только о телефонах и нетбуках, серьезное внимание уделяется широкому спектру встроенных системам, которые могут включаться в разнообразные технологические процессы. О них как раз и будет весь этот курс, о том, как писать для них программы и чем для этого удобно пользоваться.

Для начала надо разобраться в таких кажущихся общепринятыми и понятными терминах как «информация, сигналы, данные, знания и управление».

1.2. Информация, сигналы, данные, знания и управление

В XX веке слово «информация» стало термином во множестве научных областей, получив особые для них определения и толкования (О.Н. Граничин, В.И. Кияев, 2008).

Рис. 1.4. Что означает термин *информация*?

Обычно мы рассматриваем информацию как сообщение о том, что что-то произошло, т.е. одна из важнейших ее черт – изменение чего-то (объекта информации). Но информация возникает у кого-то или в чем-то (субъект информации).

Ценность информации определяется ее способностью «подтолкнуть» субъекта к определенным действиям, т.е. его способности на основании полученной информации сформулировать некоторое управляющее воздействие (табл. 1.1).

На самом деле два крайних термина в заголовке этого раздела неразрывно друг с другом связаны и не могут друг без друга существовать (как двуликий Янус или Инь и Янь в китайской философии). Информация, «не подталкивающая» к действию, – бессмысленна, так же как и бессмысленны какие-то действия без лежащей в их основе информации.

Норберт Винер более шестидесяти лет тому назад провозгласил начало эры новой науки *«Кибернетики»*, одним из первых четко

подметив, что «информационно-управленческая связь – это существенная часть любых явлений в живой и не живой природе».

Таблица. 1.1. Примеры связи информации и управления.

Информация	Управление
человек заболел	принять лекарство
дерево, около которого мы стоим, начинает падать	отойти вбок от оси падения дерева
во всем доме погас электрический свет	принять меры к исправлению повреждения в электроснабжении
наметилась тенденция на существенное снижение курса акций из нашего инвестиционного портфеля	начать продавать падающие акции
зарегистрировано приближение подводной лодки противника к стратегически важному району	отправить отряд кораблей для предотвращения проникновения подводной лодки противника или для контроля ее действий
в запретной зоне для полетов появился самолет противника	постараться сбить вражеский самолет
в определенном регионе у населения существенно возрос интерес к выступлению некоторой эстрадной группы	организовать концерт эстрадного коллектива в этом регионе

Введем для информации обозначение x, а для управления – u. Процесс принятия управленческого решения формализовано может быть записан так

$$u=U(x), \qquad (1)$$

где $U(\cdot)$ – некоторая функция от x. Часто информацию x отождествляют с вектором состояний исследуемой системы. В этом случае говорят, что соотношение (1) задает обратную связь по состоянию.

Связь между информацией и управлением можно представить в виде схемы, изображенной на рис. 1.5. Реализация управления естественно влияет на информацию, способствуя новым изменениям объекта информации. Сформированное управление u поступает в систему и воздействует на состояние x, во многих случаях изменяя его.

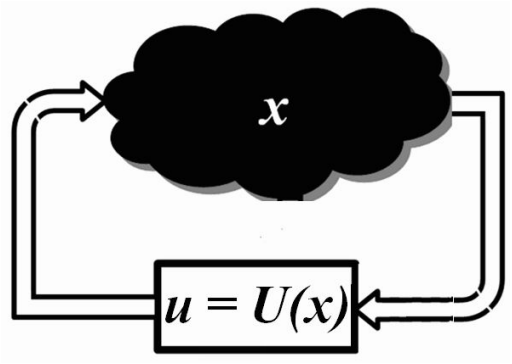


Рис. 1.5. Информация и управление.

Формализация процесса принятия решений приводит к необходимости определения понятий: сигналы, данные и знания.

Информация проявляет себя через изменения в тех или иных физических или социальных явлениях. Например, у заболевшего человека повышается температура тела. При падении дерева органы слуха фиксируют изменения в окружающих нас звуках, а глаза – изменения в отображаемой картинке. При движении подводной лодки от ее гребных винтов распространяются специфические акустические волны. При приближении самолета и попадании его в зону видимости нашего радара от него отражается посылаемая радаром электромагнитная волна. Социальные изменения проявляются через выборочные опросы, голосования и т. п.

Физические и социальные явления и процессы, изменение которых можно зарегистрировать с помощью органов чувств или приборов, называются *сигналами*. Результаты регистрации называются *данными*. Обозначим через f – сигналы и через y – данные. Схема процесса регистрации сигнала показана на рис. 1.6.

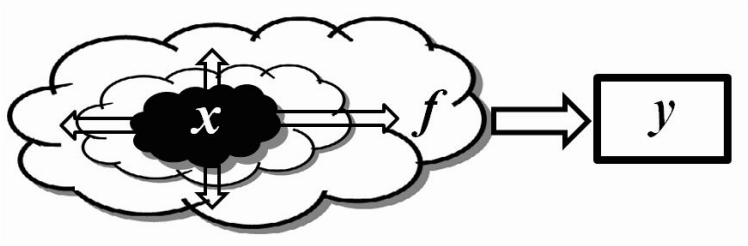


Рис. 1.6. Схема регистрации сигнала (получения данных).

Если нам доступны только данные наблюдений, то процесс принятия решения – это выработка управляющего воздействия на основе зарегистрированных данных

$$u=U(y) \qquad (2)$$

Законы управления такого типа называются обратными связями по наблюдениям.

До сих пор мы оставляли открытым вопрос о том, как выбирать функцию обратной связи. В классической теории управления вводятся понятия наблюдаемости и управляемости, соответствующие возможности восстановить состояние объекта по наблюдениям или перевести объект в произвольное заданное состояние. Здесь мы не будем детально останавливаться на этих вопросах. В последующих разделах книги будет делаться упор на программную и аппаратную реализацию тех или иных управляющих воздействий, способы выбора которых будут отдельно описываться для каждого из случаев.

Осознание связей между зарегистрированными данными и информацией, а также способы выбора управляющих воздействий в зависимости от той или иной информации будем называть знаниями.

Например, регистрация температуры тела человека дала значение более 38,5 градусов Цельсия. Это говорит о том, что пациент болен.

Информация о разливе Нила в определенный день по лунному календарю дала возможность создать первую крупную цивилизацию в Египте, позволив прогнозировать этот процесс и организовать посевы риса в строго определенное время.

В работоспособных системах имеющиеся знания позволяют на основании получаемых данных y формировать управляющие воздействия u, которые или дают какой-то выигрыш или позволяют как-то скомпенсировать негативную информацию. Без базы знаний постановка задачи о выработке обоснованных управляющих воздействий оказывается почти бессмысленной. Уточнение «почти» включено, т.к. при априорном отсутствии знаний их зачастую можно приобрести с течением времени.

Совокупность накопленных знаний называют *онтологией*.

Знания в онтологию поступают как «из вне» в виде постулируемых законов («откровений»), так и формируются «внутри» системы при обработке данных. Процесс извлечения знаний из данных определяет круг задач такой новой бурно развивающейся области как *Data Mining*.

В контексте задач управления онтология служит своеобразной базой данных (знаний), из которой выбирается наиболее адекватная текущей ситуации функция управления *U(·)* (см. рис. 1.7).

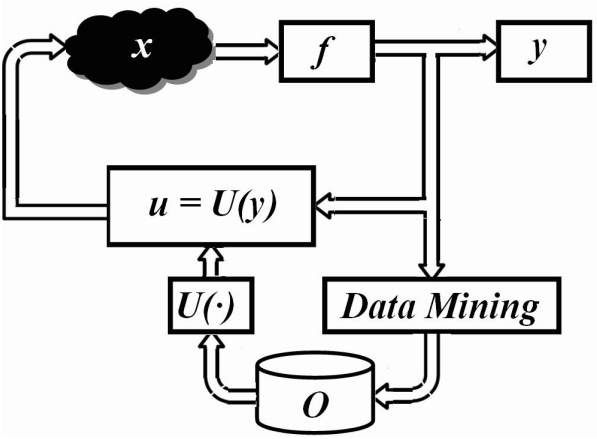


Рис. 1.7. Выбор функции управления *U(·)*.

Изменение набора знаний со временем очень важная черта, позволяющая адаптироваться к изменяющимся условиям.

Реакция на информацию (управляющее воздействие) может быть ранее определена «из вне» в соответствии с некоторыми правилами (законами), записанными в онтологии, либо формироваться «изнутри», адаптируясь к изменениям (т.е. пытаясь найти лучшее решение для поведения в изменившейся ситуации).

Процессы управления и накопления знаний часто являются взаимно противоречивыми. Целью управления обычно является достижение какого-то устойчивого состояния (по возможности не изменяющегося со временем). В этом состоянии «очень мало информации», и, следовательно, невозможно выявить или установить новые связи, значения и т.п. Например, о лежащем в пыли на обочине дороги камне мало что можно узнать при поверхностном осмотре.

Его неизменность дает мало информации (характеристик изменений). Камень надо перевернуть, поднять, толкнуть или расколоть для получения какой-то информации. Это приводит к тому, что при синтезе законов управления часто сталкиваются с проблемой недостаточной *вариативности* последовательности наблюдений. А.А. Фельдбаум сформулировал известный принцип «дуального управления»: *управляющие воздействия должны быть в известной мере изучающими, но, в известной мере, направляющими*. Например, если цель адаптивного управления состоит в минимизации отклонения вектора состояния системы от заданной траектории, то это часто приводит к вырожденной последовательности наблюдений, в то время как для успешного проведения идентификации неизвестных параметров системы должно быть обеспечено «разнообразие» наблюдений.

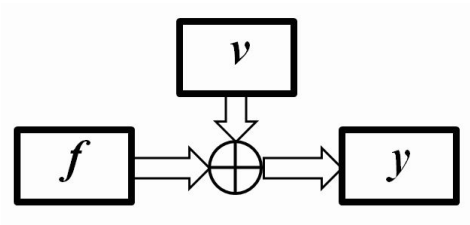


Рис. 1.8. Наблюдения с помехами.

На практике реальная ситуация осложняется еще и тем, что при получении данных в любом регистрационном устройстве к сигналам добавляются некоторые помехи (ошибки) v (см. рис. 1.8).

Структурная схема системы с обратной связью при наблюдениях с помехами приведена на рис. 1.9.

Хорошо поставленный эксперимент при тщательном измерении позволяет в некоторых случаях свести ошибки v к минимуму $v \approx 0$. Если первоначально «чистый» эксперимент не поставить, то стараются сделать его таковым с течением времени, т.е. $v \to 0$. В других случаях в задачах о наблюдении физических явлений часто достаточно обоснованно предполагают статистическую природу помех, но возможны осмысленные постановки задачи и при произвольных внешних помехах (см. О.Н. Граничин, Б.Т. Поляк, 2003).

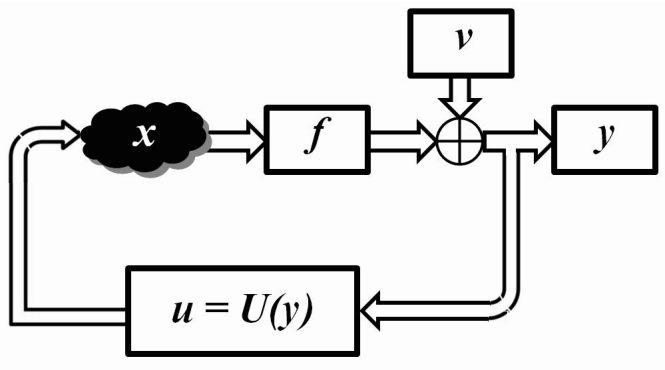


Рис. 1.9. Обратная связь по наблюдениям с помехами.

На фундаментальность понятия «*информация*» указывал еще основоположник кибернетики Норберт Винер. Феномен информации оказался настолько неоднозначным, что по праву считается одной из сложнейших проблем современности. Уже сами попытки подобраться к понятию информации, различные его трактовки в трудах ученых и практиков заставляют задуматься о необычной роли информации в жизни развивающихся систем (В.А. Абчук, С. Ю. Трапицин, В.В. Тимченко, 2006).

Первые попытки осмысления этой категории сделали основоположники теории информации. Еще в 1927 году Р.В.Л. Хартли отмечал, что количество информации, заключенной в любом сообщении, тесно связано с количеством возможностей, данным сообщением исключающихся. Фраза «яблоки красные» несет намного больше информации, чем фразы «фрукты красные» или «яблоки цветные», так как первая фраза исключает все фрукты кроме яблок, и все цвета, кроме красного. Исключение других возможностей повышает информационное содержание (В.С. Готт, А.Д. Урсул, 1971).

У. Эшби предостерегал против попыток рассматривать информацию как «*материальную или индивидуальную вещь*»: «Всякая попытка трактовать информацию как вещь, которая может содержаться в другой вещи, обычно ведет к трудным проблемам, которые никогда не должны были бы возникать» (У.Р. Эшби, 1959).

В 1949 году Шеннон и Уивер представили формулу вычисления количества информации, в которой информация возрастала с уменьшением вероятности отдельного сообщения. В их представлении

информация определяется как мера свободы чьего-либо (или какой-либо системы) выбора в выделении сообщения (К. Шеннон, Л. Уивер, 1956).

Н. Винер определяет информацию как обозначение содержания, полученного из внешнего мира в процессе приспособления к нему наших чувств. Вводя понятие о семантически значимой информации, Винер отмечает количественное и качественное отличие ЭВМ от человека: машины могут правильно работать в том случае, если получают от человека необходимую им информацию и в самой точной форме. Следовательно, характер информации, вводимой в машину, должен быть точно определён и заранее известен человеку. А живые организмы получают необходимую им информацию, благодаря постоянному взаимодействию с природой. Возникновение способности перерабатывать информацию в живых организмах есть исторически развивающийся процесс.

Н. Винер одним из первых показал, что сущность информации заключается в способности систем с памятью к саморазвитию. Другими словами, процессами развития управляет не только внешнее воздействие, но и память. По своей сути, память о прошлом представляет собой информацию, записанную в определенных структурах, которые могут передаваться, запоминаться, воспроизводиться.

В рассуждениях Н. Винера познание — это часть жизни, более того — самая ее суть. «Действенно жить — значит жить, располагая правильной информацией». При этом процесс познания, накопления информации непрерывен и бесконечен. «Я никогда не представлял себе логику, знания и всю умственную деятельность, как завершенную замкнутую картину. Я мог понять эти явления только, как процесс, с помощью которого человек организует свою жизнь таким образом, чтобы она протекала в соответствии с внешней средой. Важна битва за знание, а не победа. За каждой победой, т. е. за всем, что достигает апогея своего, сразу же наступают сумерки богов, в которых само понятие победы растворяется в тот самый момент, когда она достигнута» (Н. Винер,1958).

В 1968 году А.Д. Урсул указал на то, что информация неоднородна, она обладает качественными характеристиками, и два разных типа информации не могут сравниваться. Каждый уровень природы обладает собственной информацией. Для Урсула «человече-

ская» или «социальная» информация была одним типом, и в рамках ее он выделял еще два аспекта: семантический (содержание) и прагматический (ценность) [А.Д. Урсул, 1968].

Семантическая концепция преодолевает чисто вероятностный подход. Здесь количество информации, содержащейся в суждении по некоторой проблеме, определяется тем, насколько доказательство или допущение истинности этого суждения уменьшает энтропию, дезорганизованность системы.

При прагматическом подходе делается попытка установить зависимость между информацией и целью, которую ставит перед собой человек, работающий с информацией. Ценность информации определяется через разность между вероятностями достижения цели до и после получения информации. В соответствие с этим определением, информация измеряется всегда положительной величиной, а ценность ее может быть в известных случаях и отрицательной. Информация ценна, поскольку она способствует достижению поставленной цели. Информация может действительно иметь различную ценность, если критерием оценки избрать ее использование для различных целей.

Вопросом особой значимости является отношение информации и знания, первичности одного к другому. Понятия *«знание»* и *«информация»* часто отождествляются. В то же время знание рассматривают как высшую форму отражения действительности, а последнее, в свою очередь, как особое фундаментальное свойство материи. Говоря, что объект (или субъект) *A* отражает объект *B*, имеют в виду определенные изменения в *A*, которые соответствуют определенным изменениям в *B* и вызываются ими. Говоря об информации, подразумевается, прежде всего, особый способ взаимодействия, через который осуществляется передача изменения от *B* к *A* в процессе отражения, способ, реализующийся через поток сигналов, идущих от объекта к субъекту и особым образом в нем преобразуемых.

Уровень сложности и формы информации зависят, следовательно, от качественных характеристик объекта и субъекта, от типа передающихся сигналов, которые на самом высоком уровне реализуются в форме языковых знаковых систем. Наконец, говоря о знании, мы имеем в виду именно высший уровень информации, функционирующий в человеческом обществе. При этом в качестве знания выступает не вся информация, идущая от *B* и воспринимаемая *A*,

а лишь та ее часть, которая преобразована и переработана *А* (в данном случае — человеком) особым образом.

В процессе переработки информация должна приобрести знаковую форму и выразиться в ней с помощью других знаний, хранящихся в памяти, она должна получить смысл и значение. Следовательно, знание — это всегда информация, но не всякая информация — знание. В превращении информации в знание участвует целый ряд закономерностей, включающих знание в систему общественных связей, в культурный контекст определенной эпохи. Благодаря этому «знание становится достоянием общества, а не только отдельных индивидов» систем (В.А. Абчук, С. Ю. Трапицин, В.В. Тимченко, 2006).

В то же время информация — не просто результат отражения, не просто данные или знание. Только взаимодействуя с потребителем, данные или знание приобретают характер сообщения, сведения, т. е. становится информацией. Информация — это данные, находящиеся в постоянном обороте и движении, это знания, которые собираются, хранятся, перерабатываются, передаются и используются развивающейся системой. Знание только тогда приобретает качество информации, когда оно действует или может действовать как основа регуляции, управления. Информация есть только там, где есть движение, преобразование, использование и управление.

Относительность информации проявляется в том, что системы с различным уровнем организации извлекают из одного и того же источника различную, но всегда соответствующую своему уровню информацию. Положение об извлечении и использовании информации для построения определенных программ поведения и действия принципиально. Именно оно является основой функциональной концепции информации, связывая последнюю с процессами управления. В связи с этим, Н.И. Жуков отмечает, что «информация является не субстанциональным свойством всей материи, а функциональным свойством систем управления, связью управляющих систем с управляемыми».

Между информацией и ее значением (смыслом) можно установить четкое различие. *Информация* — это неотъемлемая данность любых систем, взаимодействующих с человеком. *Значение (смысл)* достигается только тогда, когда обретенная информация помещает-

ся в определенный контекст, то есть информация становится значимой лишь после анализа, сравнения и интеграции с другой, уже существующей в воспринимающей системе информацией.

1.3. Соединение процессов обработки данных и управления в интеллектуальных встроенных устройствах

Сложившаяся к настоящему времени парадигма использования вычислительных устройств базируется на исторически сложившемся разделении процессов обработки данных и принятия управленческих решений (после обработки). Основания этого разделения прослеживаются в истории развития средств вычислительной техники. Первоначально компьютеров было мало и они, занимая огромные пространства, требовали специальных условий для эксплуатации. Формировались особые вычислительные центры для объединенного решения в одном месте множества разных задач, причем до сих пор актуальным остается одно из приоритетных направлений развития – создание суперкомпьютеров. Встроенным устройствам традиционно отводилась роль или устройств для сбора данных, или устройств для реализации определенных управляющих воздействий. В некоторых случаях они использовались как регуляторы в простых контурах обратной связи. Суперкомпьютеры брали на себя выполнение задач Data Mining.

Но надо четко отдавать себе отчет в применимости этой традиционной парадигмы. В природе и обществе все-таки информационно-управленческие связи являются основой всех явлений и процессов. Искусственно разделяя процессы обработки данных и управления, мы существенно снижаем наши потенциальные возможности использования информационно-коммуникационных технологий.

Надо ли разделять процессы обработки данных и управления?

С начала XXI века в теории управления заметен всплеск интереса к тематике управления в сетях, коллективному взаимодействию, мультиагентным технологиям и т.п. Это во многом связано с технологическим прогрессом. Сейчас миниатюризация и быстродействие средств вычислительной техники достигли такого уровня, что стало возможным в миниатюрных встроенных системах реального времени использовать вычислительные блоки, соизмеримые по производительности с мощными компьютерами XX в. Все чаще «простые»

встроенные устройства заменяются на «интеллектуальные встроенные системы».

Новые альтернативы позволяют по-новому взглянуть на ставшую уже традиционной область Data Mining.

В литературе все чаще появляются мысли о возрождении науки «Кибернетика» с большой буквы, о появлении «неокибернетики» (Б.В. Соколов, Р.М. Юсупов). Теория управления, начавшись с регуляторов механических систем в XIX в., пройдя к концу XX в. этап глубокой интеграции с цифровыми технологиями обработки данных и принятия решений, фокусируясь в XXI в. на сетях объектов, выступает «собирателем» трех основных компонент прогресса второй половины XX века:

- теории управления (Control Theory),
- теории коммуникаций (Communication Theory)
- информатики (Computer Science)

(Б.Р. Андриевский, А.С. Матвеев, А.Л. Фрадков).

Может ли дать какое-то новое качество в обработке данных и извлечении знаний применение кибернетической парадигмы, при которой процессы «добычи знаний» и получения информации будут учитывать неразрывную связь информации и управления (и опираться на нее)?

Да, может!!!

Для иллюстрации положительного ответа в статье О.Н. Граничина (2012) рассмотрены несколько примеров повышения эффективности процессов обработки данных и управления при изменении парадигмы, основанных на рандомизации управляющих воздействий и использовании замкнутых стратегий управления в условиях неопределенностей. Остановимся на одном из них.

Эффективность замкнутых стратегий в условиях неопределенностей. Рассмотрим объект управления (ОУ) с входами $u(t)$ и выходами $y(t)$, и предположим, что задано начальное состояние $y(0) = 1$, и динамика объекта при $t = 1,2$ описывается уравнением

$$y(t) + ay(t-1) = u(t-1) + v(t)$$

с неопределенностями $v(t)$ и a двух типов:

- динамические возмущения *v(t)* неизвестны и ограничены для всех *t:* $|v(t)| \le 1$, но могут меняться со временем;
- коэффициент модели *a* также неизвестен и ограничен: $a \in [1;5]$, но он не может изменяться со временем.

Мы можем выбирать входы *u(0)* и *u(1)*. Пусть наша цель – минимизировать |*y(2)*|. Рассмотрим минимаксный функционал качества:

$$J = \sup_{a \in [1;5]} \sup_{|v(1)| \le 1, |v(2)| \le 1} |y(2)| \to \min_{u(0), u(1)} .$$

Конечно, это пример не для супервычислений. Мы специально выбрали достаточно простую модель и рассматриваем всего два шага по времени *t = 1,2*, с тем чтобы решить задачу почти устно. Но если рассмотреть *t = 1,2,3*, то сложность серьезно возрастает, а при больших *t* несколько лет тому назад одному из моих дипломников не хватило времени для решения задачи не только на персональном компьютере, но и с привлечением дополнительных вычислительных ресурсов нашего университета.

Мы сравним качество минимаксной оптимизации для двух классов допустимых стратегий управления

- программное управление,
- замкнутое управление.

Программное управление. Этот класс состоит из всевозможных пар *(u(0), u(1))*. Для того чтобы выбрать *u(0)* и *u(1)*, перепишем функционал качества в соответствии с уравнением ОУ как функцию от *a, v(1), v(2), u(0), u(1):*

$$J = \sup_{a \in [1;5]} \sup_{|v(1)| \le 1, |v(2)| \le 1} \left| a^2 - au(0) + av(1) + u(1) + v(2) \right| .$$

Произведя максимизацию по отношению к *a, v(1)* и *v(2)*, получаем функцию от переменных *u(0)* и *u(1)*. Минимизация этой функции по *u(0)* и *u(1)* дает значения *u(0) = 7, u(1) = 12,25* и

$$J_{pr}^{opt} = 8{,}25 .$$

Замкнутое управление. В момент времени $t = 1$ мы получаем выход *y(1)* и знаем предыдущее управление *u(0)*. Замкнутые стратегии управления определяются парой *u(0)* и функцией обратной связи $U_1(y(1),u(0))$. При фиксированных входе *y(1)* и управлении *u(0)* с силу уравнения *ОУ* и условия $|v(1)| \le 1$ выполняется следующее неравенство:

$$|y(1) + a - u(0)| \le 1.$$

Следовательно, можно определить интервал

$$[a^-;a^+] = [1;5] \bigcap [-1 + u(0) - y(1);1 + u(0) - y(1)],$$

который гарантированно содержит параметр *a*, и его границы вычисляются по формулам:

$$a^- = \max\{1;-1 + u(0) - y(1)\};\ \ a^+ = \min\{1 + u(0) - y(1);5\}.$$

Для оптимального выбора *u(1)* функционал качества можно переписать как функцию от *a, y(1), v(2), u(1):*

$$J = \sup_{a\in[1;5]} \sup_{|v(1)|\le 1} \left(\sup_{a\in[a^-;a^+]} \sup_{|v(2)|\le 1} \left| -ay(1) + u(1) + v(2) \right| \right).$$

Минимизация по *u(1)* приводит к формуле

$$u(1) = U_1(y(1),u(0)) = y(1)(a^- + a^+)/2,$$

учитывая которую, получаем

$$J = \sup_{a\in[1;5]} \sup_{|v(1)|\le 1} \left| y(1)(a^- + a^+)/2 + 1 \right|.$$

После исключения *y(1)* в силу уравнения *ОУ*, произведя операции максимизации, мы можем переписать *J* как функцию от *u(0)*. Минимизация по *u(0)* дает значение *u(0)* = –3 и $J_{cl}^{opt} = 2{,}125$.

В итоге имеем

$$J_{cl}^{opt} = 2{,}125 << J_{pr}^{opt} = 8{,}25.$$

Зависимость качества управления от задания класса неупреждающих стратегий адекватно понимается далеко не всех публикациях. Если все параметры объекта управления известны и помехи отсутствуют, то множества программных и замкнутых стратегий управления оказываются совпадающими.

Основатель кафедры теоретический кибернетики СПбГУ В.А. Якубович любил приводить еще и такую шутку о связи процессов обработки данных и управлений: как-то раз две мышки на молочной ферме одновременно упали с полки с сырами и угодили в неполные бидоны с молоком (в два разных). Задача у них – выжить. Время сильно ограничено, под поверхностью дышать они не могут, а сил «барахтаться» на много не хватит. Возможны две стратегии:

- анализировать ситуацию, смотреть на уходящую вверх ровную стенку бидона и пытаться придумать способ как выжить;
- начать активно «барахтаться», пытаться что-то предпринимать, физически бороться за жизнь.

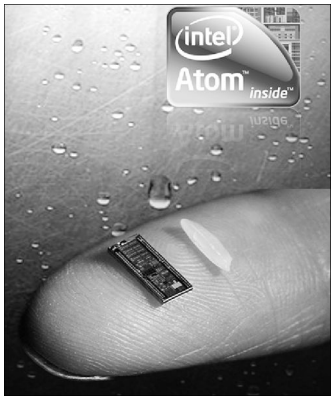
Первая мышка утонула, так и не придумав способ как выжить. У второй молоко в бидоне «взбилось» и превратилось в густую сметану, на поверхности которой мышка прожила пару дней и в конце концов нашла способ выбраться из бидона.

1.4. Компьютерный континуум Intel и процессор Intel® Atom

Основные тенденции полупроводниковой индустрии в настоящее время:

- Закон Мура пока продолжает действовать.
- Растет стоимость разработки новых технологий и материалов, а также затрат на содержание фабрик.

- Производительность процессоров (систем) также растет; ожидается скачок при переходе на 450 мм пластины.



Что есть уже сейчас для реализации умных встроенных устройств? У корпорации Intel уже появился процессор Atom. Это самый маленький процессор Intel для мобильных систем, он построен на самых маленьких транзисторах в мире. У него очень низкое энергопотребление и размером он с зернышко риса! Но – это полноценная архитектура x86.

Где сейчас уже используются, или начнут использоваться в ближайшее время процессоры Intel®Atom? Это – хорошо знакомые всем персональные компьютеры, нетбуки/неттопы, мобильные Интенет-устройства, бытовая электроника с подключением к Интернет (например, IPTV), встраиваемые устройства (особенно для автомобилей – разнообразные системы поддержки движения). Многие автомобилисты уже не представляют себе, как ездить на машине без GPS-навигатора.

Что собой представляют SoC (системы-на-чипе), которые уже сейчас производит корпорация Intel? Выше речь шла о будущих перспективах одновременного выполнения тысяч задач. Сейчас, как минимум, речь идет о переносе на чип контроллеров ввода-вывода, систем обработки видео и ряда других, существенно сокращая объемы вспомогательных обменов в памяти, повышая эффективность, производительность и упрощая конструкции. Разработчику архитектуры уже меньше надо думать куда «воткнуть» на плате тот или иной узел. Процессоры, а точнее новые SoC, становятся как бы конечными устройствами, которые можно программировать, не задумываясь о периферии. Это идеальная ситуация, когда программисту не надо думать о целой цепочке посредников, через которые должны пройти его команды и сигналы.

Что сейчас предлагает Intel на рынке процессоров для встраиваемых устройств и SoC?

Осенью 2010 года корпорация Intel объявила о выпуске семейства «систем-на-чипе» Intel®Atom™ E600 (кодовое наименование Tunnel Creek) для встраиваемых систем и о предстоящем появлении Intel® Atom™ CE4200 (кодовое наименование Groveland) – семейство «систем-на-чипе» третьего поколения, базирующееся на архитектуре Intel. Эти процессоры предназначены для использования в «умном» телевидении, в системах, объединяющих стандартное телевещание с Интернетом, библиотекой контента и мощными функциями поиска. В состав решений входят интегрированные ядро Intel®Atom с частотой 1,2 ГГц и кэш-память второго уровня объемом 512 КБ. Оно предлагает широкие возможности для разработки интерактивных, открытых и персонализированных приложений для запуска на экране телевизора. На базе Intel® Atom™ CE4200 уже разработано решение для потребительской электроники.

Эта «Система-на-чипе» осуществляет многопоточное декодирование и обработку HD-видео, поддерживает 3D, MPEG2, MPEG4-2 и VC-11. Решение оснащено интегрированным декодером HD-видео (H.264), позволяет осуществлять видеозвонки, потоковую передачу материалов на другие устройства, в том числе портативную электронику. Благодаря поддержке различных режимов питания новые решения помогают снизить энергопотребление и создавать устройства, удовлетворяющие промышленным стандартам по энергопотреблению. Планы по созданию цифровых приставок нового поколения на базе компонентов Intel озвучили ADB*, Sagemcom,* Samsung* и Technicolor*.

В ближайшее время будет выпушена новая платформа – Medfield (Penwell SoC and Avantele Passage MSIC) – разрабатываемая по технологии 32 нм с существенным уменьшением общих размеров и потребляемой мощности, с размером высокоинтергированной SoC: 144 кв мм, увеличением до 4X graphics performance.

В 2012 году корпорация Intel запустила в Израиле новую фабрику по производству микросхем по технологии 22 нм. Одним из «неожиданных» следствий закона Мура для развития бизнеса корпорации стал тот факт, что в ближайшее время при очередной смене технологической платформы возможности новых фабрик достигнут такого уровня, что в кратчайшие сроки будет возможно модернизировать все существующие места традиционного использования ком-

пьютеров, активно выйти на рынок планшетников, смартфонов и даже встроенных систем. Сохранение при этом универсальной x86 архитектуры и системы команд в значительной степени послужит явным конкурентным преимуществом.

Все это приводит к изменению роли и места на рынке корпорации Intel. К традиционным процессорам для серверов и персональных компьютеров и ноутбуков добавляются нетбуки, смартфоны, различные персональные устройства, мобильные Интернет-устройства, «умное» телевидение, бытовая электроника с подключением к Интернету (например, IPTV), встраиваемые системы и т.п. При этом в перспективе нивелируются аппаратные отличия. Уже сейчас формируется то, что можно назвать своеобразным компьютерным континуумом Intel (рис. 1.10), включающий наряду с новыми аппаратными платформами и операционные системы, специализированные программы поддержки разработчиков, средства хранения, доступа и реализации разработанных компьютерным сообществом приложений.

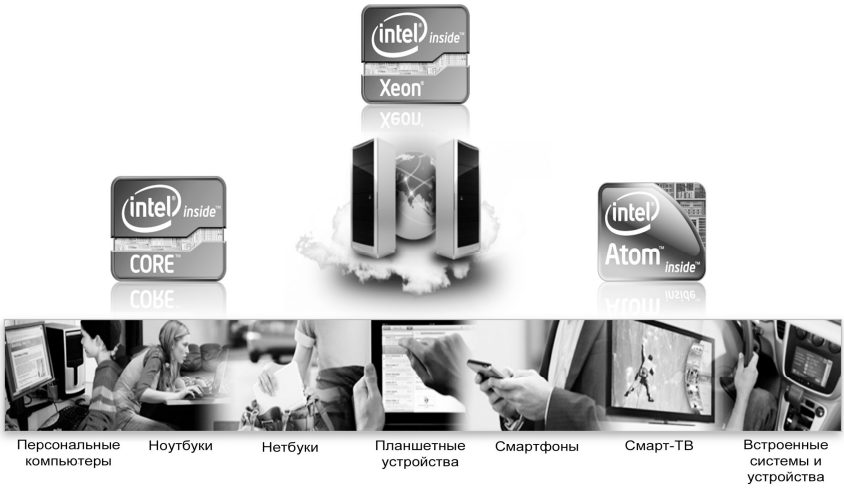


Рис. 1.10. Компьютерный континуум Intel.

1.5. Лабораторная работа № 1 «Установка ПО для разработки и тестирования приложений»

1.5.1. Цель лабораторной работы

Освоить основные приемы разработки приложений для ОС Android, изучить процесс установки и настройки среды разработки, знакомство с Android Virtual Device и пользовательским интерфейсом (UI), проведение сеанса отладки.

1.5.2. Инструкция по выполнению лабораторной работы

1.5.2.1.Подготовка

Перед тем, как приступать к созданию приложений на Android, необходимо выбрать и установить подходящую среду разработки и установить соответствующий инструментарий для разработки приложений под ОС Android - Android SDK.

В качестве среды разработки мы будем использовать Eclipse IDE. Все примеры будут приведены для этой среды разработки. Конечно, можно выбрать любую другую платформу разработки, например Intellij IDEA, но лучше всего подходит Eclipse, так как она превосходно «заточена» для разработки под ОС Android.

Убедитесь в том, что ваше ОС удовлетворяет системным требованиям для установки Eclipse IDE и Android SDK, описанным в http://developer.android.com/sdk/index.html.

Для начала разработки понадобится набор из Eclipse IDE, Android SDK и JDK (Java Development Kit). Весь набор программ можно скачать на официальном сайте Android для разработчиков по ссылке: http://developer.android.com/sdk/index.html. Также, программы доступны на официальных сайтах по ссылкам: http://www.oracle.com/technetwork/java/javase/downloads/index.html (JDK), http://www.eclipse.org/downloads/ (Eclipse IDE). Обратим

внимание, что при скачивании программ из разных источников необходимо проверить, совместима ли версия Eclipse с Android SDK.

Android SDK представляет собой средство разработки приложений под ОС Android, и распространяется в виде ZIP или EXE файлов. Android SDK состоит из пакетов, содержимое которых возможно подгружать с помощью Android SDK Manager. В Android SDK входят следующие пакеты:

- пакет с версиями платформ (SDK Platform) – содержит папки с версиями платформ, в каждую из которых входит файл android.jar – файл архива Java, содержащий все классы SDK Android, необходимые для создания приложений;
- пакет с образами систем (System Image) – образ системы используется в иммуляторе Android устройства при отладке и разработке приложений;
- пакет с исходным кодом платформы Android (sources);
- пакет документации (Documentation) – документация SDK предоставляется локально и через Интернет. В основном она выполнена в формате Javadocs, что позволяет легко ориентироваться во множестве пакетов SDK;
- каталоги инструментов (SDK Tools, SDK Platform-tools) – содержат все инструменты для отладки и тестирования Android-приложений;
- USB-driver – каталог, содержащий все необходимые драйвера для подключения к поддерживающим Android устройствам.

Начальный пакет Android представляет только основные инструменты. Остальные компоненты, при необходимости, можно скачать и установить (например, GoogleAPI).

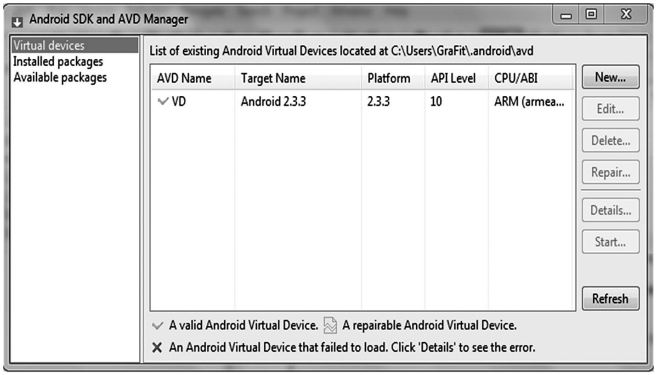
Android предлагает плагин ADT (Android Developer Tools) для Eclipse IDE, который обеспечивает разработчиков эффективными инструментами для создания Android-приложений. Он расширяет богатые возможности Eclipse новыми полезными утилитами. К примеру, после установки ADT на панелях toolbar и menubar появятся «контролы», позволяющие быстро создавать проект.

Для установки ADT, необходимо открыть панель расширения среды (InstallNewSoftware) в закладке «Help», а затем в появившем-

ся окне добавить новый путь https://dl-ssl.google. com/android/eclipse/.

Далее Eclipse выполнит поиск требуемых данных, и в выпадающем списке вы сможете выбрать ADT и установить его, нажав Install. Запустится процесс инсталляции плагина. Если в ходе установки возникнут ошибки, то, возможно, что указанный ресурс более по каким-то причинам недоступен, либо вы используете старую версию Eclipse, либо не установили JDK.

Запустите AVD Manager прямо из Eclipse (Window->Android SDK and AVD Manager). На экране появится окошко, содержащее слева список из трех основных компонентов: виртуальные устройства, установленные и доступные пакеты. В установленных пакетах находятся все компоненты, установленные по умолчанию или по необходимости. С помощью вкладки «доступные пакеты» можно автоматически скачивать и устанавливать необходимые решения. К примеру, мы можем установить Google API's любой версии, с целью использования в разрабатываемом приложении сервисов Google (карты, поиск).



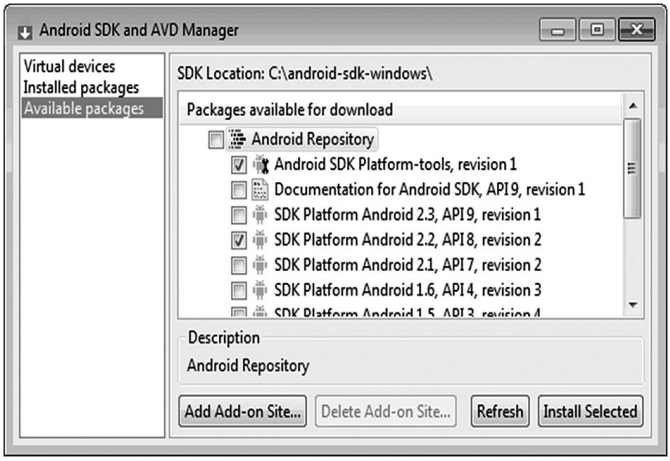


Рис. 1.11. Android API для элементов UI.

Прежде чем писать приложение с использованием Android API's необходимо создать виртуальное устройство, которое позволит тестировать написанное. Чтобы создать виртуальное устройство в главном окне Android SDK and AVD Manager, кликните на кнопку New. Появится диалоговое окно, которое позволяет настроить виртуальную машину. Заполняйте поля сверху вниз. Всплывающие мини-окошки помогут вам при заполнении. На этом установка Android полностью завершена, теперь можно приступать к написанию приложений.

Кнопки являются неотъемлемым компонентом любого интерактивного приложения. ОС Android обладает превосходными средствами создания и управления кнопками. Кнопка описана классом android.widget.Button, который является суперклассом для всевозможных типов кнопок (флажковая кнопка, радиокнопка, переключатель).

У кнопки определено несколько конструкторов:

→ Button(Context context);

→ Button(Context context, AttributeSetattr);

→ Button(Context context, AttributeSetattr, intdefStyle).

Context – это класс содержащий информацию о среде приложения и предоставляющий доступ к различным системным ресурсам и классам.

Поля текстового ввода. Android.widget.EditText– является небольшой облицовкой вокруг TextView, позволяющей создавать редактируемый текст.

EditText имеет несколько конструкторов:

→ EditText(Contextcontext);
→ EditText(Contextcontext, AttributeSetattr);
→ EditText(Context context, AttributeSetattr, intdefStyle).

Класс EditText содержит несколько методов. Рассмотрим некоторые из них:

→ public void selectAll(Spannable); метод фиксирует целиком в качестве текста указанный ресурс. Spannable — это интерфейс для организации текста в виде блоков (кусков текста).
→ public Editable getText(); возвращает текст, отображаемый TextView. Editable интерфейс для описания динамического текста, содержимое и разметка которого могут быть изменены.
→ public void setText(CharSequence, TextView.BufferType); устанавливает текст, отображаемый в TextView. Первый параметр определяет текст, второй – способ его хранения: динамический текст, нормальный текст(default), блочный текст.

Списки. Android.widget.ListView – класс, отображающий набор элементов в виде списка с возможностью прокрутки. Элементы, ассоциированные с этим представлением, описываются классом ListAdapter.

ListAdapter – наследует базовый класс Adapter и служит мостом между данными и ListView. Часто данные могут быть представлены курсорами, но необязательно. Удобство в том, что ListView может отображать любые данные, лишь бы они были завернуты в ListAdapter. ListAdapter имеет несколько подклассов (ArrayAdapter, BaseAdapter,..), которые предназначены для различных целей. Приведем пример использования ArrayAdapter.

ArrayAdapter – представляет данные в виде массива и добавляет удобный функционал (добавление, удаление, поиск) для работы с ними.

→ ArrayAdapter(Context, TextViewResourceId);

→ ArrayAdapter(Context, TextViewResourceId, T[] objects).

TextViewResourceId – ссылка на xml-файл представления, данных списка. Параметр T[] objects задает отображаемые данные.

LinearLayout. Линейный менеджер размещения используется для отображения элементов в виде линейных списков, горизонтальных или вертикальных. Причем можно использовать несколько Layouts одновременно, в этом случае они будут делить экран.

→ LinearLayout(Context context);

→ LinearLayout(Context context, AttributeSet);

→ LinearLayout(Context context, AttributeSet , intdefStyle).

Рассмотрим основные методы:

generateLayoutParams(AttributeSet attr) – задает параметры layout (ширина, длина, фон, притяжение);

getOrientation() – задает направление размещения.

RelativeLayout. Этот менеджер размещения используется в том случае, если необходимо размещать элементы относительно какого-то виджета. Позиция размещения задается в виде: элемент X расположить ниже элемента Y.

RelativeLayout очень мощное средство расположения элементов, не зависящее от размеров окна приложения и не привязанное к какой-то конкретной позиции. Таким образом, с помощью RelativeLayout мы получаем «резиновый», легко программируемый интерфейс. Конструкторы те же самые, что и у LinearLayout.

TableLayout. Этот менеджер размещения, располагает элементы в табличном виде по строкам и колонкам. TableLayout состоит из элементов разметки TableRow, определяющих строку в которой будет содержаться виджет. TableLayout не отображает границы ячеек, из которых состоит. Каждая строка TableLayout имеет ноль или более ячеек, а каждая ячейка может хранить объект для отображения (ViewObject). Число столбцов в таблице определяется самой длинной по количеству ячеек строкой. Таблица может содержать пустые ячейки. Ячейки могут охватывать несколько столбцов, также как и в HTML. Ширина колонки есть максимум по длинам ячеек, принадлежащих этому столбцу.

В конкретных случаях, можно настроить таблицу под собственные требования. В этом помогут стандартные методы, жестко задающие параметры таблицы.

TabLayout. Иногда нам приходиться описывать несколько различных отображений в одном окне обеспечивая между ними быстрое и комфортное переключение. В таком случае необходимо использовать вкладочный менеджер размещения (tablayout). Чтобы создать tablayout, необходимо выполнить следующее: создать TabHost либо в виде xml описания, либо непосредственно кодом в программе. Этот элемент является контейнером для вкладок (табов) оконного вида. TabHost должен иметь два дочерних элемента: TabWidget и FrameLayout.

TabWidget – элемент представляющий список вкладок FrameLayout, используемый для отображения содержимого каждой вкладки.

Существует два способа использования TabLayout:

1) Можно создать несколько представлений в одной Activity и переключаться между ними, кликая по табам.

2) Можно создать несколько Activity, описывающих различные представления, а в главном Activity обеспечить корректное переключение.

Android SDK обеспечивает большой набор инструментов, которые необходимы для отладки приложений. У вас должен быть JDWP-совместимый отладчик (Java Debug Wire Protocol (JDWP) – это протокол, используемый для обмена данными между отладчиком и виртуальной машиной Java, которую он и отлаживает).

Основные компоненты среды отладки Android:

ADB (AndroidDebugBridge) – выступает в качестве посредника между устройством и программой.

Dalvik Debug Monitor Service (DDMS) – это элемент плагина Android, используемый для анализа работы виртуальной машины. Последовательно отображает поток информации, которая описывает каждый шаг работы виртуальной машины.

Device or Android Virtual Device (AVD**).** Ваше приложение должно выполняться на устройстве или в AVD так чтобы оно могло быть отлажено. Демон adb устройства работает на устройстве или эмуля-

торе, и предоставляет средства для демона adb хоста для предоставления связи с устройством или эмулятором.

ADB – клиент-серверное приложение, состоящее из трех компонентов (Рис. 1.12).

Клиент запускается на машине разработчика. Клиент можно запустить из командной строки при помощи команд, посылаемых adb. Другие инструменты Android, вроде плагина ADT и DDMS, тоже создают adb-клиенты.

Сервер запускается на машине разработчика в виде фонового процесса. Сервер управляет соединениями между клиентами и adb-сервисом, запущенным на эмуляторе или устройстве.

Сервис – фоновый процесс, который запускается на каждом эмуляторе или устройстве.

Debug	`logcat [<option>] [<filter-specs>]`	Prints log data to the screen.	
	`bugreport`	Prints `dumpsys`, `dumpstate`, and `logcat` data to the screen, for the purposes of bug reporting.	

	`jdwp`	Prints a list of available JDWP processes on a given device.	You can use the `forward jdwp:<pid>` port-forwarding specification to connect to a specific JDWP process. For example:
			`adb forward tcp:8000 jdwp:472`
			`jdb -attach localhost:8000`

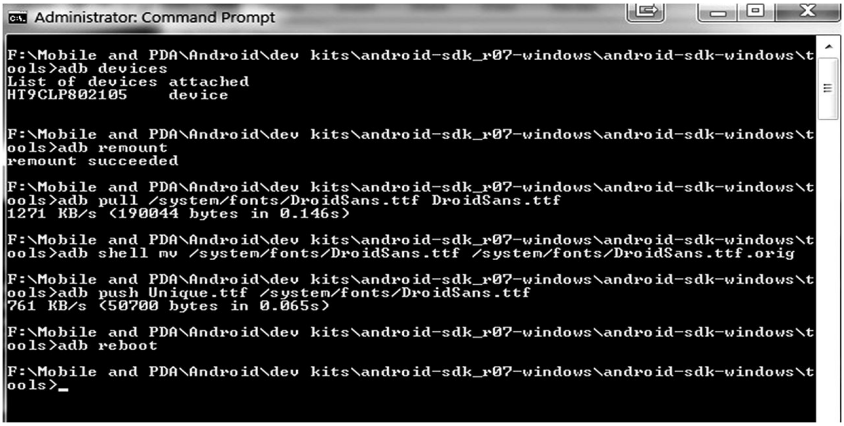


Рис. 1.12. Пример использования клиент-серверного приложения ADB.

LogCat – система ведения лога в Android обеспечивает механизм для сбора и просмотра системных отладочных сообщений. Логи из различных приложений и элементов системы Android собираются воедино, а затем их можно просматривать и фильтровать посредством команды logcat (Рис. 1.13).

Logcat можно использовать для просмотра и слежения за содержанием буферов системного лога. Формат использования:

```
[adb] logcat [<option>] ... [<filter-spec>] …
```

Фильтрация вывода лога: каждое сообщение лога в Android имеет *тэг* и *приоритет*.

Тэг – это строка, указывающая на компонент системы, от которого принято сообщение (например, View для системы view).

Приоритет имеет одно из нижеследующих значений (в порядке от меньшего к большему): V – Verbose (низший приоритет); D – Debug; I – Info; W – Warning; E – Error; F – Fatal; S – Silent (наивысший приоритет, при котором ничего не выводится).

Выражения фильтра позволяют указать системе нужные комбинации <тэг> и <приоритет>. Остальные сообщения система не выводит. Выражения фильтра имеют следующий формат: <тэг>:<приоритет>.

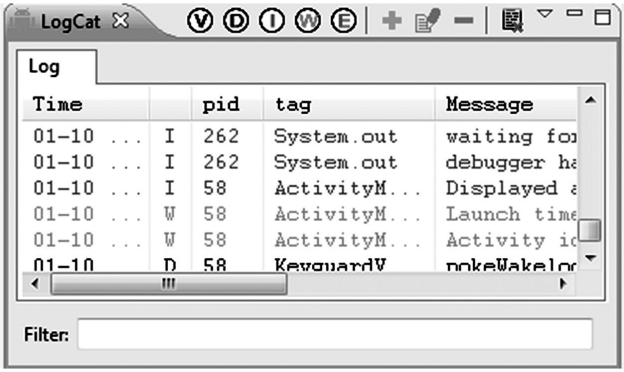


Рис. 1.13. Пример использования LogCat.

Отлаживать приложения в Eclipse гораздо проще, чем в консоли, благодаря использованию standpoints. Но проблема отладки через

Eclipse может возникнуть, если вы отлаживаете приложения, запущенные на эмуляторе, а не на устройстве. Эмулятор работает довольно медленно, а при отладке каждый переход работает более 3 секунд. Это очень долго. Так что, если вы отлаживаете большое приложение, и при отладке наткнулись на цикл, содержащий 100 итераций, то, чтобы его пройти, вам потребуется более length(circle)*3*100 секунд. В таких случаях лучше использовать отладку с фильтрами через консоль и выводить отладочную информацию в файл с дальнейшим выявлением ошибок.

При отладке в Eclipse пользуйтесь утилитой

LogCat (Window->Show view->Android->LogCat,

которая, по сути, является оболочкой команд отладки через консоль.

1.5.2.2.Описание приложения

Приложение, которое будет подробно рассмотрено ниже, написано в IDE Eclipse Indigo, поэтому используйте именно эту версию или более позднюю.

Пример имеет исключительно обучающий характер и нацелен на то, чтобы показать разработчику процесс инициализации, описания внутреннего устройства компонентов UI и дизайна интерактивных приложений.

Общая идея. Обычно, если приложение не имеет сложной структуры, то в одном состоянии приложения используется только один стиль размещения. Мы рассмотрим приложение, которое содержит в себе много различных layouts. Это достигается путем создания в качестве главного layout, TabLayout, у которого каждый tab имеет свой способ размещения. Конечно, такая архитектура используется редко, так как пользователю приятнее видеть плавное переключение между табами в противовес нашему варианту.

Структура.

TabLayout →

- Tab1: -> LinearLayout.
- Tab2: -> RelativeLayout.
- Tab3: -> GridLayout.

Архитектура Tab1.

Интерфейс для работы с датой в Android.

Компоненты:

- TextView – простое текстовое поле, в котором будет отображаться текущая или вручную установленная дата.
- Button – кнопка, которая инициирует появление окошка DitaPicker, в котором отображается текущая дата.

В окне стандартно есть контролы, позволяющие изменить дату.

Архитектура Tab2.

В этом табе будут представлены некоторые виджеты UI и в качестве бонуса будет продемонстрирован Spinner и процесс взаимодействия с ним.

Компоненты:

- TextView – текстовое поле, отображающее статическую информацию.
- EditText – текстовое поле с поддержкой редактирования.
- Button – кнопка OK.
- Button – кнопка Cancel.
- Spinner – выпадающий список с обработкой события выбора.

Архитектура Tab3.

В этом табе будут представлены графические объекты (картинки), размещенные в виде таблицы.

Компоненты:

- GridView – представление, отображающее элементы в двумерной сетке с поддержкой вертикального или горизонтального скрола. Элементы, ассоциированные с Grid представлением, должны быть представлены с помощью ListAdapter.

1.5.2.3.Модель «Дизайн-XML»

Two hands. Android SDK очень удобен, не только тем, что представляет понятный и прозрачный API, который позволяет

даже неквалифицированному разработчику быстро освоить основные библиотеки Android и приступить к написанию собственных приложений, но и тем, что структура программы может быть разделена на две независимые части: Java-код, описывающий поведение программы, и Xml-код, описывающий компоненты UI(кнопки, списки, окна, …).

Xml-подход. Такой способ был выбран по нескольким причинам:

- Разделение работы веб-дизайнера и программиста.
- Уменьшение массивности программы за счет использования фрагментов xml-файла в качестве виджетов UI.

1.5.2.4. Выполнение работы

Проделайте следующие шаги, чтобы посмотреть предварительно подготовленное приложение в действии.

Приложение может быть запущено с использованием виртуальной машины Davlik.

1) Импорт приложения.

Запустите Eclipse, выберите пункт меню «File», в списке выполните команду «Import». В появившемся окне во вкладке General, щелкните на «Existing project into workspace». Появиться диалоговое окно, в котором вы можете указать путь до проекта приложения в виде архива, либо в виде простой папки. Пример находится в каталоге lab01 файла labAtom32.rar. Затем нажимаете на кнопку «OK». В «Packageexplorer» у вас появится проект с исходным кодом.

2) Запуск приложения.

Нажмите на стрелку рядом с кнопкой «Run», в раскрывшемся списке выберите:

```
Run as->Android application
```

В результате действий проект перестроится и запустится приложение. После загрузка эмулятора можно убедиться в работоспособности программы.

Файл графического интерфейса. Рассмотрим подробнее файл описания графического интерфейса приложения. Очень часто при написании приложений возникает необходимость разместить на экране несколько представлений, совершенно различных и не связанных друг с другом по логике. В этом случае программист сразу же представляет себе интерфейс расположения вкладок браузера. Такой интерфейс поддерживает и операционная система Android.

В папке layouts проекта, откройте с помощью xml редактора файл «tab_layout.xml». Рассмотрим его содержимое.

Самый первый узел, «Tabhost», представляет собой контейнер для вкладок. Он содержит всего два типа элементов: «TabWidget» и «FrameLayout». Элемент «TabWidget» используется для описания дизайна панели вкладок, содержит список вкладок. Менеджер размещения «FrameLayout» необходим при описании содержимого одной вкладки. Для «TabHost» и «FrameLayout» заданы идентификаторы, так как в коде программы нам будет необходимо к ним обращаться. Менеджер размещения «LinearLayout» будет использоваться на первой вкладке. Он содержит текстовое поле для отображения даты и кнопку, позволяющую установить дату. Следующий менеджер размещения – «RelativeLayout» – содержит несколько текстовых полей: поле с поддержкой редактирования, несколько кнопок и выпадающий список. И, наконец, последний менеджер размещения – «LinearLayout» – содержит «Grid» представление, для отображения картинок в таблице. Количество колонок жестко не задано, зато можно увидеть, что ширина каждой колонки – 90 пикселей.

Код приложения. В файле «LabAndroidActivity.java» находится код, описывающий всю логику программы. Архитектура приложения основана на следующей идее: внутри одной activity обеспечить навигацию между represent-tab используя «TabHost».

Прежде, чем переходить к подробному разбору кода, остановимся на вопросе «Жизненный цикл приложения».

В отличие от приложений в большинстве других систем приложения Android не имеют единой точки входа (например, отсутствует функция `main()`). Вместо этого существуют четыре типа основных компонентов, из которых строятся андроид-приложения и которые система может запускать по мере необходимости. Это Activity, Service, Broadcast receiver и Content provider. Как упоминалось

раньше, Activity представляет собой визуальный интерфейс для одного действия, которое пользователь может совершить. Приложение обычно строится из нескольких Activity, каждый из которых выполняет свою определенную функцию. Обычно Activity не зависят друг от друга. Activity может находиться в одном из трех состояний:

- **Активно** (`active`) или **запущено** (`running`) – находится на переднем плане (на вершине стека activity текущей задачи) и имеет фокус для взаимодействия с пользователем.
- **Приостановлено** (`paused`) – потеряло фокус, но всё ещё видно пользователю. Сверху находится другое activity, которое или прозрачно или закрывает не весь экран. Приостановленное activity полностью <живое> (его состояние сохранено и оно привязано к оконному менеджеру), но может быть уничтожено системой в случае нехватки памяти.
- **Остановлено** (`stopped`) – полностью перекрыто другим activity. Оно больше не видно пользователю и будет уничтожено системой, когда понадобится память.

Жизненный цикл activity состоит из трёх вложенных циклов:

- Жизненный цикл activity начинается с вызова метода `onCreate()`, в котором производится первоначальная настройка глобального состояния, и завершается вызовом метода `onDestroy()`, в котором оно освобождает занятые ресурсы. Например, в onCreate() можно создать поток, загружающий данные из сети в фоновом режиме, и затем остановить его в `onDestroy()`.
- Видимая часть жизненного цикла происходит между вызовами `onStart()` и `onStop()`. В течение этого времени пользователь может видеть activity на экране, хотя оно может быть не на переднем плане и не взаимодействовать с пользователем. Между этими двумя методами вы можете выделять ресурсы, необходимые для отображения activity пользователю. Методы `onStart()` и `onStop()` могут вызываться столько раз, сколько activity становится видимым или скрытым для пользователя.

- На переднем плане activity находится между вызовами `onResume()` и `onPause()`. В течение этого времени activity находится поверх других и взаимодействует с пользователем. Activity может часто переходить в состояние паузы и выходить из него. Например, метод `onPause()` может быть вызван, когда устройство переходит в спящий режим или когда запускается другое activity, а метод `onResume()` – при получении результата от закрывающегося activity. Таким образом, код в этих двух методах должен быть довольно легким.

Рассмотрев основные детали построения приложения, перейдем к обзору кода. В главном методе onCreate(), зададим в качестве содержимого Activity созданный нами интерфейс с помощью вызова метода `setContentView` («Путь к файлу через класс ресурсов»). Очень часто при раздельном описании интерфейса и логики приложения, необходимо получить доступ к конкретному элементу интерфейса (кнопка), чтобы назначить ему определенную функцию. В таком случае, лучшим способом является использование метода `findViewById` (идентификатор). Параметр идентификатор – это имя компонента, определенное в файле интерфейса. Использование метода `setup()`, перед добавлением вкладок необходимо, если объект «TabHost» загружается через файл, как в нашем примере.

Далее используется TabSpec для создания вкладки, устанавливания индикатора и содержимого вкладки. В классе определена закрытая переменная mDateDisplay, в которой храниться время, и переменная mPickDate – кнопка, вызывающая диалог DatePicker. У кнопки установлен фон с авто-триггером, подключаемым из файла «Button.xml». Кнопке назначен слушатель на нажатие, внутри определения которого реализован метод `onClick(View)`, вызываемый всякий раз при возникновении события. В методе `onClick()` вызывается функция `ShowDialog()`, которая влечет функцию `onCreateDialog()`. Обратите внимание, что внутри этой функции инициализируется объект DatePickerDialog, причем вторым параметром конструктору передается объект слушателя для события «изменение даты». В переменные mYear, mMonth, mDay заносится текущая дата и вызывается метод `UpdateDisplay()`, который обновляет содержимое текстового блока.

Аналогично, создается вторая вкладка и заполняется содержимым. Единственное, что стоит здесь отметить, это класс Spinner. Объект класса Spinner – выпадающий вниз список, с возможностью выбора элементов. В файле strings.xml определен тег string-array, содержащий элементы item. По сути, там содержатся данные, которые будут отображены в Спиннере. Метод createFromResource() создает новый ArrayAdapter, который связывает каждый элемент из массива «planets_array» с начальным представлением Спиннера. Иначе говоря, как каждый элемент будет отображаться в тот момент, когда пользователь надумает раскрыть Спиннер. Метод setDropDownViewResource, устанавливает настройку «раскрытие-вниз» для Спиннера. И наконец, все элементы ArrayAdapter сопоставляются со Спиннером при помощи метода `setAdapter()`. Для Спиннера регистрируется слушатель на событие «Выбор элемента списка». Реализация объекта слушателя описана в отдельном классе с целью проведения аналогии читателем с предыдущим способом. Класс `MyOnItemSelectedListener` реализует интерфейс `OnItemSelectedListener`. В методе `onItemSelected()` пользователю показывается всплывающее окошко с информацией о выбранном элементе.

Последняя вкладка отображает элементы с помощью табличного менеджера компоновки, описанного в файле интерфейса. С помощью метода `setAdapter()`, назначаются данные для отображения. `ImageAdapter` - дочерний класс `BaseAdapter`. Массив `mThumbs` содержит ссылки на картинки через класс ресурсов. Аналогично, как и Спиннеру, `GridView` устанавливается слушатель на событие «Выбор элемента». Заканчивают метод `onCreate()` три строки кода, в которых контейнеру табов добавляются созданные вкладки, по очереди.

1.6. Выводы

Материал этой главы вместе с лабораторной работой № 1 поможет читателям выбрать для себя необходимое устройство на платформе Intel Atom, загрузить и проинсталлировать необходимое программное обеспечение и начать с ней работать.

1.7. Контрольные вопросы

1) Что такое информационная революция?

Варианты ответов:

1. Преобразование общественных отношений из-за кардинальных изменений в сфере обработки информации
2. Преобразование общественных отношений из-за кардинальных изменений в форме получения информации
3. Преобразование общественных отношений из-за накопления чрезмерного объема информации и сложности в ее хранении

2) Что такое кибернетика?

1. Форма существования и систематизации результатов познавательной деятельности человека.
2. Наука об общих закономерностях процессов управления и передачи информации в различных системах
3. Сфера человеческой деятельности, направленная на выработку и теоретическую систематизацию объективных знаний о действительности.

3) Когда началась кибернетическая революция?

1. 1990-е
2. 2000-е
3. 2010-е

4) Что считается ключевым ресурсом в настоящее время?

1. Информация
2. Капитал
3. Материалы

5) Что такое адаптивность?

1. Способность реализовать алгоритм при существенных ограничениях
2. Способность действовать в любой среде
3. Способность быстрого ответа на непредсказуемые изменения

6) Какого типа законы управления называются обратными связями по наблюдениям?

1. Когда управляющее воздействие формируется на основе

изначально известных данных
2. Когда процесс принятия решения – это выработка управляющего воздействия на основе зарегистрированных данных
3. Когда управляющее воздействие формируется на основании связей между зарегистрированными данными

7) Что такое онтология?
1. Совокупность накопленных знаний
2. Алгоритм распределения информации
3. Функция управления

8) Как влияют помехи на сбор данных в любом регистрирующем устройстве?
1. Они не влияют, так как всегда можно свести влияние ошибки к минимуму
2. Существенно не влияют, так как можно предположить, что помехи отсутствуют
3. О помехах необходимо делать реалистичные предположения

9) В каком случае информация может быть значимой?
1. Когда ее достаточно много
2. После анализа, сравнения и интеграции с другой, уже существующей в воспринимающей системе информацией
3. Когда она формируется на основе уже существующей информации

10) Что такое SoC?
1. System of Chaos
2. System of Control
3. System on Chip

11) Какой из перечисленных процессоров потребляет меньше всего энергии?
1. Intel Celeron Core i5 660.
2. Intel Celeron Dual-Core E1500.
3. Intel Atom E620.
4. Intel Atom Z500.

12) Что такое Eclipse?
    4. Java-машина
    5. Операционная система
    6. Среда разработки

13) Для чего предназначен Android SDK?
    1. Разработка прикладных приложений для Android
    2. Разработка драйверов для Android
    3. Разработка программно-аппаратных комплексов для Android

14) Что нельзя сделать с помощью плагина ADT?
    1. Создать Android-проект в Eclipse
    2. Запустить Eclipse в Dalvik
    3. Запустить сеанс отладки Android-приложения в Eclipse

15) Какие версии Android API поддерживаются SDK?
    1. Любые по выбору программиста
    2. Только самая свежая версия
    3. Только те версии, которые включены в установочный файл SDK

16) Как называется базовый класс кнопок в Android API?
    1. PushButton
    2. BaseButton
    3. Button

17) Для чего предназначен класс ListAdapter?
    1. Корректное отображение списка внутри layout
    2. Связывает данные списка и их представление на экране
    3. Содержит список адаптеров Android

18) Как размещает виджеты LinearLayout?
    1. Только по горизонтали
    2. Только по вертикали
    3. В зависимости от возможностей экрана устройства
    4. и по горизонтали, и по вертикали

19) На каких устройствах возможна отладка Android-приложений через ADB?
    1. Только на виртуальных
    2. Только на реальных

3. и на реальных, и на виртуальных

20) Что такое logcat?
   1. Протокол для отладки Java-приложений
   2. Система ведения лога Android
   3. Демон для отправки информации о сбое

21) Какая фильтрация сообщений возможна в logcat?
   1. По тэгу и приоритету
   2. По дате
   3. По PID

Список литературы

1. Граничин О.Н., Кияев В.И. Информационные технологии в управлении. – М.: Изд-во «Бином», 2008, 336 с.
2. Rzevski, G. Modelling large complex systems using multi-agent technology // In Proc. of 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD2012), August 8-10, Kyoto, Japan, 2012. PP. 434-437.
3. Винер Н. Кибернетика, или управление и связь в животном и машине. – М.: Наука. 1983.
4. Фельдбаум А.А. О проблемах дуального управления // В кн.: Методы оптимизации автоматических систем. – М.: Наука. 1972.
5. Граничин О.Н., Поляк Б.Т. Рандомизированные алгоритмы оценивания и оптимизации при почти произвольных помехах. – М.: Наука. 2003.
6. Абчук В.А., Трапицин С.Ю., Тимченко В.В. Менеджмент. Учебник. — СПб.: ООО «Книжный Дом», 2006.
7. Винер Н. Кибернетика и общество. — М.: ИЛ, 1958.
8. Готт В.С., Урсул А.Д. Определенность и неопределенность как категории научного познания. М.: Знание, 1971.
9. Шеннон К., Уивер Л. Математическая теория связи. М: Физматгиз, 1956.
10. Урсул А.Д. Природа информации. — М.: Политиздат, 1968.

11. Эшби У.Р. Введение в кибернетику. — М.: ИЛ. 1959.
12. Жуков Н.И. Информация. — Минск. Наука и техника, 1966.
13. Соколов Б.В., Юсупов Р.М. Неокибернетика – возможности и перспективы развития // Докл. на общем пленарном заседании 5-й науч. конф. `Управление и информационные технологии (УИТ-2008), Россия, СПб., 14–16 октября, 2008 г. СПб., 2008. С. 1-15.
14. Андриевский Б.Р., Матвеев А.С, Фрадков А.Л. Управление и оценивание при информационных ограничениях: к единой теории управления, вычислений и связи // Автоматика и телемеханика. 2010. № 4. С. 34--99.
15. Граничин О.Н. Обратные связи, усреднение и рандомизация в управлении и извлечении знаний // Стохастическая оптимизация в информатике. 2012. Том 8. Вып. 2. С. 3-48.
16. Граничин О.Н. Кибернетика и перспективы развития суперкомпьютеров // Суперкомпьютеры. 2012.

2. Мультиагентные технологии

Мультиагентные технологии. Понятие «Агента».

2.1. Введение

Под мультиагентными технологиями сейчас часто понимают как технологии разработки и использования мультиагентных систем (МАС) [1], так и мультиагентное управление (МАУ).

Задачи управления и распределенного взаимодействия в сетях динамических систем привлекают в последнее десятилетие внимание все большего числа исследователей. Во многом это объясняется широким применением мультиагентных систем в разных областях, включая автоматическую подстройку параметров нейронных сетей распознавания, управление формациями, роение, распределенные сенсорные сети, управление перегрузкой в сетях связи, взаимодействие групп БПЛА, относительное выравнивание групп спутников, управление движением групп мобильных роботов, синхронизации в энергосистемах и др.

На практике все чаще используются распределенные системы, выполняющие определенные действия параллельно, для которых актуальна задача разделения пакета заданий между несколькими вычислительными потоками (устройствами). Подобные задачи возникают не только в вычислительных сетях, но также и в производственных сетях, сетях обслуживания, транспортных, логистических сетях и др. Оказывается, что при естественных ограничениях на связи, децентрализованные стратегии способны эффективно решать такого типа задачи.

2.2. Мультиагентный подход

В основе мультиагентного подхода лежит понятие мобильного программного агента, который реализован и функционирует как самостоятельная специализированная компьютерная программа или элемент искусственного интеллекта.

Изначально, до появления соответствующих информационных технологий, "агент" был человеком, которому делегировалась часть полномочий – как в выполнении конкретных функций, так и в принятии решений. В первых (не компьютерных) мультиагентных системах агенты представляли сотрудников компаний, от имени и по поручению которых они взаимодействовали между собой при выполнении определенной задачи – например, представители покупателя и продавца в торговой сети или в других видах бизнеса). Такие системы наследовали многие черты «бюрократической» организации, включая централизацию управления, статичную структуру и узкоспециализированную агентную функциональность. В частности, базовый агент (резидент) получал задачу, декомпозировал её и распределял подзадачи между другими агентами, после чего получал результат и принимал решение – при этом, как правило, большинство агентов занимались исключительно сбором и поставкой информации.

На смену таким системам, копирующим централизованную иерархию, быстро пришли распределенные системы, в которых знания и ресурсы распределялись между достаточно «самостоятельными» агентами, но сохранялся общий орган командного управления, принимающий решения в критических или конфликтных ситуациях. Дальнейшим шагом в этом направлении стала парадигма полностью децентрализованных систем, в которых управление происходит только за счет локальных взаимодействий между агентами. При этом узкая функциональная ориентация агента на решение какой-то одной отдельной части "общей" задачи постепенно стала уступать *место универсальной целостности* (автономности). Примерами таких децентрализованных организаций отчасти могут служить колонии насекомых, например, пчел или муравьев.

Суть мультиагентных технологий заключается в принципиально новом методе решения задач. В отличие от классического способа, когда проводится поиск некоторого четко определенного (детерминированного) алгоритма, позволяющего найти наилучшее решение проблемы, в мультиагентных технологиях решение получается автоматически в результате взаимодействия множества самостоятельных целенаправленных программных модулей — так называемых агентов.

Зачастую классические методы решения задач либо неприменимы к реальной жизни (не трудно представить себе, что значит попытаться решить задачу управления предприятием в непредсказуемой динамичной обстановке современного бизнеса, даже с помощью высшей математики), либо они требуют огромных объемов расчетов (для которых не хватит мощности всех современных компьютеров), либо они вовсе отсутствуют.

Значит ли это, что ситуация, когда точный алгоритм решения отсутствует, безнадежна?

Нет, отвечают мультиагентные технологии. В конце концов, лю¬дям в своей жизни постоянно приходится в условиях дефицита вре¬мени и средств решать задачи, не имеющие точного формального решения — и они решаются часто не самым худшим образом.

На рис. 2.1 показаны в сравнении две схемы (из [2]) построения программного обеспечения: традиционная и на базе мультиагентной системы. В МАС каждой сущности ставится в соответствие программный агент, который представляет ее интересы.

Дело в том, что человеку присущ интеллект — это его отличает от компьютера, действующего строго по заложенной в него программе. То, что позволяет ему ориентироваться в сложной обстановке, иметь дело с нечетко поставленными задачами, адаптироваться к меняющимся условиям. Неопределенность присутствует, когда существует набор альтернатив, и невозможно предсказать, какой из вариантов окажется лучшим по прошествии достаточно длительного времени.

При составлении расписаний движения грузовиков, это та ситуация, когда, к примеру, существует выбор между несколькими грузовиками, перевозящими грузы, несколькими дорогами, которые

могут быть использованы для достижения разных точек назначения, и многими водителями, которые могут управлять грузовиками. Каждый из ресурсов (грузовик, дорога и водитель) имеют различные свойства.

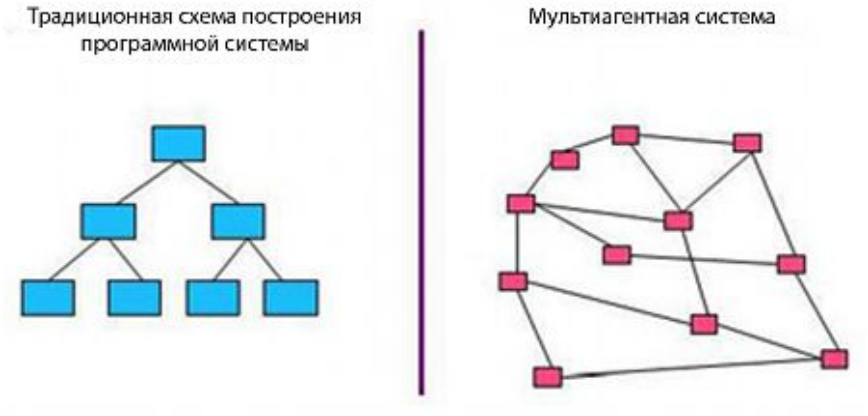


Рис. 2.1: Схемы построения программного обеспечения [2].

Неопределенность возрастает в ситуациях, когда возможны непредсказуемые события, такие как изменения в условиях поставок или спроса, аварии или сбои ресурса, задержки, отмены заказов, и тому подобное [3].

А есть ли интеллект, скажем, у колонии муравьев?

С одной стороны, каждый отдельно взятый муравей, очевидно, им не обладает. С другой — колония в целом проявляет удивительные образцы поведения, которое во многом может считаться интеллектуальным. Такие ситуации называются проявлением эмерджентного интеллекта, или неожиданных свойств, которыми обладает система, но не обладает ни один входящий в нее отдельный элемент. Возникающий при этом эффект "интеллектуального резонанса" часто так и называют "Интеллект роя". Действительно, интеллект и физическая сила одной пчелы не так велики, но рой пчел, согласованно действующий, может победить медведя и даже человека. Агенты очень похожи на членов команды, которые могут соревноваться друг с другом или сотрудничать в процессе принятия решения. Ключевая особенность эмерджентного интеллекта — динамика и непредсказуемость процесса принятия решений. На прак-

тике это означает, что решение достигается за счет сотен и тысяч взаимодействий, которые почти невозможно отследить. Но это и не требуется, поскольку агентам дают цели, которые они должны достигать, но не предопределяют сценарии исполнения задач по достижению этих целей. Эти сценарии формируются и исполняются агентами самостоятельно. На каждом шаге агенты рассматривают входы системы и реагируют на непредсказуемые события (задержки, сбои, изменения). Реакция может быть самостоятельной, или осуществляться во взаимодействии с оператором. Таким образом, эмерджентный интеллект — это не есть какой-либо один новый и специально сконструированный уникальный "блок", добавленный к системе. Напротив, это нечто (результат самоорганизации), что возникает как бы "из воздуха" (за счет множества скрытых или явных условий, сложившихся в ситуации), спонтанно и в заранее не предвиденный момент времени, и так же неожиданно исчезает, но в процессе своего существования определяющим образом руководит работой всей системы. Тут мы имеем дело с возникновением порядка из хаоса, с одним из тех явлений, которые изучали и описывали такие выдающиеся ученые, как Александр Богданов (теория организации), Илья Пригожий (самоорганизация в физических системах), Марвин Минский (психология и теория мышления), Артур Кестлер (биология) [2].

2.3. Мультиагентные системы

В начале XXI в. группа ведущих мировых ученых [4], проработав несколько лет, составила список приоритетных задач кибернетики на ближайшие 50 лет. Среди них:

- динамически реконфигурируемое интеллектуальное управление,
- асинхронная теория управления,
- управление через Интернет,
- перепрограммирование системы управления бактериями,
- создание футбольной команды роботов, которая выиграет у победителя кубка мира среди людей.

МАС кардинально отличаются от традиционных "жестко" организованных систем, и, в перспективе, способны помочь в решении этих задач.

Начало построения моделей и применения искусственных мультиагентных систем (МАС) на практике было положено в 1960-х годах. В качестве основы были взяты достижения таких областей деятельности человека, как системы искусственного интеллекта (Artificial Intelligence), параллельные вычисления (Parallel Computing), распределенное решение задач (Distributed Problem Solving). Многоагентные системы имеют реальную возможность интегрировать в себе самые передовые достижения перечисленных областей, демонстрируя принципиально новые качества [5]. Сейчас МАС — одно из наиболее динамично развивающихся и перспективных направлений в области искусственного интеллекта [5].

Открытый характер современного информационного общества и глобальной рыночной экономики приводит к ускорению научно-технического прогресса и обострению конкуренции на рынках. Это заставляет предприятия искать новые методы и средства организации и управления, направленные на более качественное и эффективное удовлетворение индивидуальных запросов потребителей. Большинство современных систем характеризуются отсутствием средств своевременной идентификации новых потребностей и возможностей в среде, позволяющих предприятию оперативно принимать эффективные решения по реконфигурации производственных, кадровых, финансовых и других ресурсов. Типичными примерами событий, вызывающих необходимость заново идентифицировать потребности и возможности, являются: появление нового выгодного заказа, для исполнения которого недостаточно собственных ресурсов предприятия, выход из строя части имеющихся ресурсов, а также изменение критериев принятия решений. Чем выше неопределенность, чем более распределенный характер имеют процессы принятия решения и чем чаще случаются незапланированные события, тем ниже эффективность существующих систем, не способных самостоятельно принимать решения и автоматически перестраиваться под изменения в среде [6]. Кроме того, необходимость модификации схемы принятия решений в традиционных системах оказывается сложной и трудоемкой задачей, которая требует высо-

кой квалификации исполнителей. Это делает разработку и эксплуатацию таких систем крайне дорогостоящими. Соответственно, еще одной актуальной проблемой современности становится рост объемов информации и степени сложности описания систем.

Для решения подобных проблем применяются мультиагентные технологии, в основе которых лежит понятие "агента", которое в последнее время было адаптировано ко многим областям как прикладного и системного программирования, так и к исследованиям в областях искусственного интеллекта и распределенных интеллектуальных систем. Причем в каждом конкретном случае понятию придается несколько разное значение.

2.4. Агенты

Как отмечено в [7], первоначально идея создания интеллектуального посредника (*агента*) "возникла в связи с желанием упростить стиль общения конечного пользователя с компьютерными программами, поскольку доминирующий, в основном, и ныне стиль взаимодействия пользователя с компьютером предполагает, что пользователь запускает задачу явным образом и управляет ее решением. Но это совершенно не подходит для неискушенного пользователя". Иначе говоря, сначала идея интеллектуального посредника возникла как попытка интеллектуализации пользовательского интерфейса.

Развитие методов искусственного интеллекта позволило сделать новый шаг к изменению стиля взаимодействия пользователя с компьютером. Возникла идея создания так называемых "автономных агентов", которые породили уже новый стиль взаимодействия пользователя с программой. Вместо взаимодействия, инициируемого пользователем путем команд и прямых манипуляций, пользователь вовлекается в совместный процесс решения [8]. При этом, как пользователь, так и компьютерный посредник, оба принимают участие в запуске задачи, управлении событиями и решении задачи. Для такого стиля используется метафора *персональный ассистент* [7], который сотрудничает с пользователем в той же рабочей среде.

Словари дают следующее толкование слова *агент*: "некто или нечто, прикладывающее усилия для достижения эффекта". Такое

самое общее определение указывает на первый признак агента — *агенты совершают действия*. Часто утверждается, что агенты не просто совершают действия, но они действуют *автономно и рационально*. Под автономностью обычно понимают, что агент действует без прямого вмешательства человека или другой управляющей сущности. Под рациональностью понимают стремление агента оптимизировать значение некоторой оценочной функции. Мера рациональности неявно указывает на то, что агент имеет цели (желания англ. desires), которых агент "хочет" достичь, и представления о внешнем мире (убеждения, англ. beliefs), на которые агент опирается при выборе действия (реализации намерений, англ. intentions — множество избранных, совместимых и достижимых желаний).

Еще одним важным свойством агента является то, что он помещен во внешнюю среду, с которой он способен взаимодействовать. Обычно, среда не контролируется агентом, он лишь способен влиять на нее. Разделение намерений и желаний необходимо, так как агент может иметь несовместимые желания или желания могут быть недостижимы. Поскольку агент ограничен в ресурсах и не может достичь всех желаний одновременно, естественно выбирать наиболее значимые цели — намерения. Итак, агент — разумная сущность, помещенная во внешнюю среду, способная взаимодействовать с ней, совершая автономные рациональные действия для достижения целей, т. е. Интеллектуальный агент — это агент, обладающая следующими свойствами:

- *реактивность* (англ. reactivity) — агент ощущает внешнюю среду и реагирует на изменения в ней, совершая действия, направленные на достижение целей;
- *проактивность* (англ. pro-activeness) — агент показывает управляемое целями поведение, проявляя инициативу, совершая действия направленные на достижение целей;
- *социальность* (англ. social ability) — агент взаимодействует с другими сущностями внешней среды (другими агентами, людьми и т. д.) для достижения целей.

При разработке системы каждое из первых двух свойств достигается достаточно легко. Наибольшую сложность представляет совмещение в системе обоих свойств в нужных пропорциях. Будет

не слишком эффективно, если агент жестко следует сценарию достижения цели, не реагируя на изменения во внешней среде и не обладая способностью заметить необходимости корректировки плана. Но также не эффективно будет и поведение, ограниченное лишь реакцией на поступающие из вне стимулы, без какого-либо планирования целенаправленных действий. На самом деле описанная проблема настолько сложна, что даже далеко не все люди способны эффективно ее решать. Очень часто можно увидеть человека, который кидается на каждую подвернувшуюся возможность, но никогда не доводит ничего до конца, т. к. не концентрируется на этой возможности достаточное время, чтобы полноценно ее реализовать. Но также часто встречаются люди, которые, однажды поставив цель и сформировав план, будут пытаться принципиально ему следовать, не замечая изменений в ситуации, требующих пересмотра целей или планов.

Достичь свойства социальности тоже нелегко. *Социальность* — это не просто обмен данными. Помимо коммуникации, социальное поведение должно включать кооперацию с другими сущностями, заключающуюся в разделении целей между отдельными сущностями, совместном планировании и координации действий, направленных на достижение общих целей. Социальное поведение, как минимум, предполагает наличие у агента представлений о целях других сущностей и том, как они планируют этих целей достичь.

Сложность формулирования содержательных практически значимых задач и невозможность априорного точного задания всех условий функционирования выдвигают адаптивные постановки проблем, отдельно выделяя такую особенность агентов, как *адаптивность* — способность автоматически приспосабливаться к неопределенным и изменяющимся условиям в динамической среде.

Таким образом, предшественниками программных агентов можно считать *сложные адаптивные системы*, которые умеют подстраиваться под ситуацию или обстоятельства и принципиальным образом менять свое поведение или характеристики, чтобы обеспечить решение стоящих перед ними задач. Однако в случаях, когда агент функционирует в сложной, постоянно изменяющейся среде, взаимодействуя при этом с другими агентами, такая муль-

тиагентная система значительно сложнее просто адаптивной системы, так как она быстрее обучается и может действовать эффективнее за счет перераспределения функций или задач между агентами.

Сложные системы часто рассматривают как среду действия агентов. С понятием сложных систем связаны следующие фундаментальные идеи, которые непосредственно влияют на функционирование MAS.

- в сложных системах существуют автономные объекты, которые взаимодействуют друг с другом при выполнении своих определенных задач;
- агенты должны иметь возможность реагировать на изменяющиеся условия среды, в которой они функционируют и, возможно, изменять свое поведение на основе полученной информации;
- сложные системы характеризуются возникающими структурами – логически связанными схемами, которые формируется в результате взаимодействия между агентами;
- сложные системы с возникающими структурами часто существуют на грани порядка и хаоса;
- при создании сложных систем на базе агентов имеет смысл рассматривать биологические аналогии, такие как: паразитизм, симбиоз, репродукцию, генетику, митоз и естественный отбор (например, компания British Telecom при формировании сети направления звонков использует модель деятельности колонии муравьев).

Концепция агентов, разработанная в рамках мультиагентных технологий и мультиагентных систем (MAS), предполагает наличие активного поведения агентов, т.е. способности компьютерной программы самостоятельно реагировать на внешние события и выбирать соответствующие действия. Сегодня агентные технологии предлагают различные типы агентов, модели их поведения и свойства, семейство архитектур и библиотеки компонентов, ориентированные на современные требования.

В настоящее время не существует устоявшегося определения агента. Ниже перечислены некоторые из них:

"Агент – это аппаратная или программная сущность, способная действовать в интересах достижения целей, поставленных пользователем".

"Под агентом можно понимать самостоятельную программную систему, состоящую из программ-объектов, имеющую возможность принимать воздействие из внешнего мира, определять свою реакцию на это воздействие и в соответствии с этим формировать ответное действие. Такие агенты способны действовать, «рассуждать» и обмениваться данными друг с другом в сети для формирования индивидуальных или коллективных решений".

По определению Кристиана Доннегара (директор по технологии компании Living Systems, занимающейся созданием систем совместной коммерции на основе технологии агентов): "агенты – программные объекты, которые выполняют определенные упреждающие и корректирующие действия в соответствии с заданиями, делегированными человеком".

Алан Кэй, который начал первым развивать теорию агентов, определил агент как "программу, которая после получения задания способна поставить себя на место пользователя и действовать по адаптивному сценарию. Если же агент попадает в тупик, он может задать пользователю вопрос, чтобы определить, каким образом ему необходимо действовать дальше".

Простая компьютерная программа отличается от агента тем, что "не утруждает" себя целевым поведением и анализом достигнутых результатов. Напротив, агент, представляющий интересы пользователя, "заинтересован" в том, чтобы задание было выполнено. В случае неудачи или какого-то сбоя он должен повторить попытку позднее или иметь про запас альтернативный вариант решения проблемы. Агенты в процессе отработки заданий всегда формирует список выполненных действий, результаты тестирования и верификации и отсылают его в управляющую систему.

Отметим, однако, что вопрос по определению того, что такое *агент* не закрыт до сих пор, и обсуждение этого вопроса периодически выносится на конференции самого высокого уровня. На рис. 2.2 показаны области знания и технологии, с помощью которых

формируются механизмы искусственного интеллекта и применения мультиагентных систем.

Рис. 2.2. Области знания и технологии, используемые интеллектуальными агентами.

На основании изложенного выше можно скомпилировать следующее определение: агент – это самостоятельная программная система [12]:

- имеющая возможность принимать воздействие из внешнего мира;
- определяющая свою реакцию на это воздействие и формирующая ответное действие;
- изменяющая свое поведение с течением времени в зависимости от накопленной информации и извлеченных из нее знаний,
- обладающая мотивацией и способная после делегирования полномочий пользователем поставить себя на его место и принять решение, соответствующее ситуации».

Интеллектуальный агент должен обладать следующими свойствами:

- автономность – способность функционировать без вмешательства со стороны своего владельца и осуществлять контроль внутреннего состояния и своих действий;
- адаптивность – агент обладает способностью обучаться;

- коллаборативность – агент может взаимодействовать с другими агентами несколькими способами, играя разные роли;
- способность к рассуждениям – агенты могут обладать частичными знаниями или механизмами вывода, а также специализироваться на конкретной предметной области;
- коммуникативность – агенты могут общаться с другими агентами;
- мобильность – способность передачи кода агента с одного сервера на другой;
- социальное поведение – возможность взаимодействия и коммуникации с другими агентами;
- реактивность – адекватное восприятие среды и соответствующие реакции на ее изменения;
- активность – способность генерировать цели и действовать рациональным образом для их достижения;
- наличие базовых знаний – знания агента о себе, окружающей среде, включая других агентов, которые не меняются в рамках жизненного цикла агента;
- наличие убеждений – переменная часть базовых знаний, которые могут меняться во времени;
- наличие цели – совокупность состояний, на достижение которых направлено текущее поведение агента;
- наличие желаний – состояния и/или ситуации, достижение которых для агента важно;
- наличие обязательств – задачи, которые берет на себя агент по просьбе и/или поручению других агентов;
- наличие намерений – то, что агент должен делать в силу своих обязательств и/или желаний.

Иногда в этот же перечень добавляются и такие человеческие свойства, как рациональность, правдивость, благожелательность.

2.7. Лабораторная работа № 2 «Разработка приложения для БПЛА»

Пример получения и записи данных от видеоустройства.

2.7.1. Цель лабораторной работы

Демонстрация процесса разработки практического приложения, позволяющего получать фотографии со встроенной фотокамеры и записывать их в заданную нами папку на внешнюю память мобильного устройства.

В настоящее время миниатюризация электронных компонент и технологичность их изготовления позволяет применять небольшие и доступные беспилотные летательные аппараты (БПЛА) для выполнения задач исследования территории. Несмотря на небольшие размеры «игрушечные» самолеты сейчас способны развивать скорость свыше четырехсот км/ч при установки на них небольшого реактивного двигателя. При этом технические характеристики даже сравнительно малых по размеру моделей позволяют устанавливать на борт микрокомпьютер, фото- и видеокамеры, современные средства связи. В частности, для многих задач можно в качестве бортового микрокомпьютера использовать смартфон Intel с ОС Android, построенный на основе процессора Intel Atom X86.

Далее детально рассмотрим простейшую задачу для БПЛА – видеонаблюдение, которое заключается в получении и передаче серии фотоснимков с камеры, установленной на БПЛА, в ЦОД через GSM/GPRS канал.

2.7.2. Инструкция по выполнению лабораторной работы

Задача видеонаблюдения естественным образом распадается на составные части (рис. 2.7). Во-первых, нам нужно уметь подключаться к Интернету с помощью GSM-модема. Во-вторых, мы должны научиться получать кадр от камеры и сохранять его в формате

JPG. В-третьих, нам нужен механизм для приема данных на ЦОД. Эти подзадачи независимы друг от друга.

Одним из стандартных датчиков получения данных из внешней среды на различных мобильных устройствах – фотокамера. Сейчас такие камеры имеют миниатюрный размер, высокое разрешение фото и видео съёмки. В основном такие камеры были доступны только как встроенная опция в мобильных устройствах. В последнее, время все чаще встречаются цифровые миниатюрные фотокамеры, как самостоятельное устройство или внешнее дополнение. ОС Android обладает всеми необходимыми библиотеками для работы с фотокамерами, а также, для работы с памятью куда можно записать накопленные данные. В качестве памяти может выступать карта памяти, например, microSD, или часть встроенной флеш-памяти устройства.

В этой лабораторной работы мы будем использовать встроенную фотокамеру и внутреннюю память мобильного устройства, посмотрим, как подключать необходимые нам устройства в приложение и каким образом использовать нужные нам библиотеки в коде. Для этого потребуются необходимые базовые знания языка Java, знакомство с Android SDK и необходимыми методами и классами

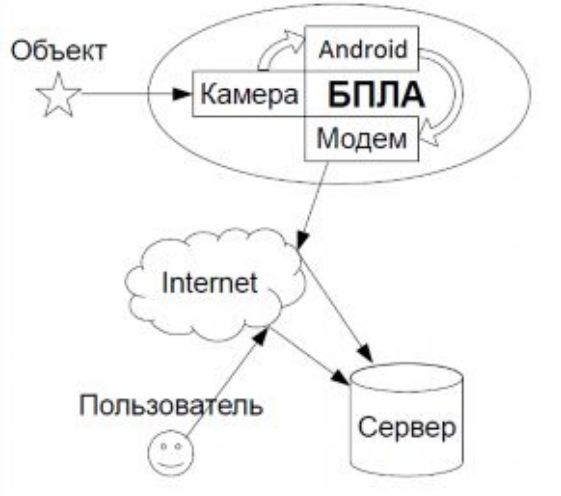


Рис. 2.7.

.

В качестве одной из составляющих приложения будут использованы материалы лабораторной работы «Захват видеоизображения» из первой части курса. Также необходимо мобильное устройство (планшет, коммуникатор, нетбук и т.п.), работающий под управлением ОС Android. Нами использовался планшет компании Intel с ОС Android 4.0.

Задача состоит из трех частей – получение кадра от камеры, сжатие кадра в формат jpeg, запись полученного кадра в памяти мобильного устройства.

Для начала работы необходимо запустить Eclipse SDK, создать новое рабочее пространство (например, android_camera), и создать новый проект Android Application Project. Для этого необходимо зайти во вкладку File и перейти в New -> Android Application Project. Далее необходимо задать название приложения, версию Android и запустить проект. Мы будем работать с уже созданным нами проектом ImageUploader. Для этого необходимо войти File -> Import -> и выбрать папку с проектом.

Далее опишем, какие необходимые элементы программного кода нашего примера относятся непосредственно к работе с фотокамерой и памятью.

Для начала в проекте необходимо прописать то оборудование, которое будет задействовано в приложении и его дополнительные функции. Для этого необходимо открыть файл AndroidMonifest.xml и внести следующие строки:

```
1. <uses-sdk
2. android:minSdkVersion="8"
3. android:targetSdkVersion="14" />
4. <uses-permission
   android:name="android.permission.CAMERA" />
5. <uses-permission an-
   droid:name="android.permission.WRITE_EXTERNAL_STORA
   GE" />
6. <uses-feature an-
   droid:name="android.hardware.camera" />
7. <uses-feature an-
   droid:name="android.hardware.camera.autofocus" />
```

В первой, второй и третьей строках описываются версии минимальной SDK и той, для которой предназначено приложение. Далее необходимо установить «разрешения» на использования того или иного устройства – permission. Строки 4 и 5 являются запросами на «разрешения» для использования фотокамеры и внешнего носителя. В строках 6 и 7 прописаны функции камеры. Если вы планируете, что приложение будет работать с соответствующим оборудованием и его определенными функциями, то функции указываются без дополнительных условий. Если же вы хотите чтобы ваше приложение также запускалось на устройствах, которые не имеет указанных устройств и функций, то в соответствующей строке необходимо прописать:

```
.............................android:required="false" />
```

после указания функции. В данном примере мы хотим воспользоваться камерой и функцией автофокус.

Для того чтобы начать использовать камеру необходимо получить доступ к ней. Для этого необходимо воспользоваться Camera.open( ) . В нашем примере часть кода работы с камерой такой

```
1. public boolean openCam() {
2.          try {
3.          camera = Camera.open();
4.          Camera.Parameters parameters = cam-
   era.getParameters();
5.    parame-
   ters.setFlashMode(Camera.Parameters.FLASH_MODE_OFF)
   ;
6.          if (focusBox.isChecked())
7.    parame-
   ters.setFocusMode(Camera.Parameters.FOCUS_MODE_AUTO
   );
8.          else
```

```
9.    parame-
  ters.setFocusMode(Camera.Parameters.FOCUS_MODE_INFI
  NITY);
10.   parameters.setJpegQuality(95);
11.         List<Camera.Size> psizes = parame-
  ters.getSupportedPictureSizes();
12.         int w=0;
13.         int h=0;
14.         for (int i=0; i < psizes.size(); i++) {
15.               if (w < psizes.get(i).width && h <
  psizes.get(i).height) {
16.               w = psizes.get(i).width;
17.               h = psizes.get(i).height;
18.                         }
19.                     }
20.         log ("Picture size:
  "+String.format("%dx%d", w, h));
21.         parameters.setPictureSize(w,h);

22.         camera.setParameters(parameters);

23.         cam-
  era.setPreviewDisplay(fakeView.getHolder());
24.   } catch (Exception e) {
25.         e.printStackTrace();
26.         log("Camera open fail:"+e.toString());
27.         closeCam();
28.         return false;
29.               }
30.               return true;
31.       }
```

В начале, мы указывает на инициализацию и начала работы с камерой. В строках с 4 по 23 мы указываем параметры работы камеры и параметры получаемых нами данных в зависимости от того, какие данные нам необходимы. В нашем примере это: работа без вспышки, с автофокусом, данные в формате jpeg. Затем мы перебираем размеры изображения, которые может предоставить камеры, и выбираем максималь-

ный. В конце мы устанавливаем камере фиктивный Surface для предварительного отображения кадра. Для решения задачи он не нужен, но камера не сможет без него работать. При неправильной работе камеры или её отсутствии программа сообщает нам об этом “Camera open fail” и выводит подробности ошибки.

Далее в примере описывается еще две функции камеры – остановка работы камеры closeCam( ) и takePhoto( ). Эти классы вызываются по средством нажатия заданных ранее кнопок. Сам код такой:

```
1. public void closeCam() {
2.          if (camera != null) {
3.               camera.stopPreview();
4.               camera.release();
5.               camera = null;
6.               log("Release camera");
7.          }
8.     }
9.
10.         private void takePhoto(){
11.              if (previewStarted) {
12.                   log("Skip photo, wait previ-
  ous...");
13.                   return;
14.              }
15.              if (camera == null) {
16.                   log("Camera not initial-
  ized...");
17.                   return;
18.              }
19.              previewStarted = true;
20.              camera.startPreview();
21.              camera.takePicture(null, null, jpeg-
  Callback);
22.         }
```

где с 1 по 8 строку – команда остановки работы камеры, а с 9 по 22 – команда сделать фото в формате jpeg. . Обратите вни-

мание, что для получения кадра от камеры непосредственно перед этим включается режим предпросмотра. Потом, в обработчике кадра, флаг предпросмотра будет сброшен.

Далее необходимо указать, что делать с данными полученными с камеры. В нашем пример, мы конвертируем данные в формат jpeg с разными параметрами “small” и “big”, даем им имена и записываем их в две разных папки. Мы делаем два размера для получения фотографий предварительного просмотра, а также для возможной их отправки через Internet.

```
1. PictureCallback jpegCallback = new PictureCall-
   back() {
2.          public void onPictureTaken(byte[] data,
   Camera camera) {
3.          FileOutputStream outStream = null;
4.          String filename = String.format(imageDir +
   File.separator + "big" + File.separator + "%d.jpg",
5.                System.currentTimeMillis()/1000);
6.          String filename2 = String.format(imageDir
   + File.separator + "small" + File.separator +
   "%d.jpg",
7.                System.currentTimeMillis()/1000);
8.          String gps = "No";
9.          try {
10.         outStream = newFileOutputStream(filename);
11.         outStream.write(data);
12.         outStream.close();
13.         Bitmap origImage = BitmapFac-
   tory.decodeByteArray(data, 0,data.length);
14.         int origHeight = origImage.getHeight();
15.         int origWidth = origImage.getWidth();
16.         double scale = (orig-
   Height>origWidth?origHeight:origWidth)/300;
17.         int newHeight = (int)(origHeight/scale);
18.         int newWidth = (int)(origWidth/scale);
19.         Bitmap smallImage = Bit-
   map.createScaledBitmap(origImage, newWidth, new-
   Height, false);
20.         origImage.recycle();
```

```
21.         int quality = 75;
22.         outStream = new FileOutput-
  Stream(filename2);
23.         BufferedOutputStream bos = new Buffere-
  dOutputStream(outStream);
24.         smallImage.compress(CompressFormat.JPEG,
  quality, bos);
25.         smallImage.recycle();
26.         bos.close();
27.               };
```

В строках с 1 по 8 описывается место, где необходимо брать исходные данные изображения, и каким образом формировать название файла. В строках с 9 по 27 указывается, что необходимо сделать с данными и куда их записать, а именно создается копия изображения меньшего размера с пониженным качеством для более быстрой передаче по сети. В примере, исходные данные берутся с камеры, название формируется исходя из номера фотографии, и записывается на карту памяти в формате .jpg.

2.7.3. Задания для самостоятельной работы

1. Изменить папку для сохранения и размер получаемого кадра.
2. Добавить использование вспышки при съемке.
3. Добавить функцию определения лица на кадре.

2.8. Выводы

В этой главе были представлены основные положения новых мультиагентных технологий и сформулировано понятие интеллектуального агента. Далее будут рассмотрены вопросы взаимодействия агентов, платформы для разработки мультиагентных систем и проблемы мультиагентного управления.

В лабораторной работе мы познакомились с примером процесса разработки практического приложения под ОС Android, был рассмотрен пример разработки прототипа практического приложения

по захвату видеокадров, сохранению их и записи в формате .jpg на внешний носитель.

2.9. Контрольные вопросы

1) В чем основные отличие децентрализованной стратегии управления от централизованной?

Варианты ответов:

1. Отсутствие единого центра принятия решений
2. Отсутствие жестко ориентированных связей
3. Отсутствие помех и задержек

2) Какими основными свойствами обладает интеллектуальный агент?

Варианты ответов:

4. Реактивность, автономность, адаптивность.
5. Реактивность, проактивность, социальность
6. Адаптивность, социальность.

3) В чем заключается мультиагентный подход решения задач?

Варианты ответов:

1. Поиск некоторого детерминированного алгоритма, позволяющего найти наилучшее решение проблемы
2. Решение получается автоматически в результате взаимодействия множества самостоятельных целенаправленных программных модулей
3. Поиск оптимальной стратегии для управления группой автономных агентов

4) С какими непреодолимыми сложностями сталкиваются классические методы решения задач в современных реальных условиях?

Варианты ответов:

1. Требуется огромный объем ресурсов
2. Требуется постоянное обновление информации о текущем состоянии среды
3. Требуется найти адекватную математическую модель

5) Какие ситуации называются проявлением эмерджентного интеллекта?

Варианты ответов:

1. Входящие в систему элементы не могут действовать отдельно друг от друга
2. Проявление неожиданных свойств, которыми обладает система, но не обладает ни один входящий в нее отдельный элемент
3. Когда выход из системы одного элемента не влияет на поведение остальных

6) Что понимается под термином "Интеллект роя"?

Варианты ответов:

1. Интеллектуальный резонанс, возникший в результате проявления эмерджентного интеллекта
2. Суммарные знания, накопленные всеми элементами системы
3. Элемент, который обладает наибольшими ресурсами и знаниями по сравнению с остальными

7) Когда было положено начало применения и построения мультиагентных систем?

Варианты ответов:

1. 1960-е
2. 1980-е
3. 2000-е

8) Какое свойство мультиагентных систем более всех помогает решать задачи при динамических внешних изменениях среды?

Варианты ответов:

1. Способность выделения дополнительных ресурсов для решения новых задач
2. Способность одновременно получать информацию со всех элементов системы
3. Способность самостоятельно принимать решения и автоматически перестраиваться под изменения в среде

9) Как именно создание автономных агентов помогло облегчить общение пользователей с программой?

Варианты ответов:

1. Появились интеллектуальные подсказки для пользователей
2. Большинство работы по запуску задачи, управлению событиями и решению задач стало происходить автоматически
3. Вместо создания команд и прямых манипуляций, пользователь вовлекается в совместный процесс решения

10) Как соотносятся желания и намерения агентов?

Варианты ответов:

1. Намерения – это наиболее значимые желания
2. Желания – это наиболее значимые цели агентов, а намерения – это те желания, для выполнения которых у агента есть ресурсы
3. Между ними нет разницы

11) Верно ли следующее определение агента: агент — разумная сущность, помещенная во внешнюю среду, способная взаимодействовать с ней, совершая автономные рациональные действия для достижения целей?

Варианты ответов:

1. Да
2. Нет, агент не совершает автономных действий
3. Нет, агент может взаимодействовать только с другими агентами, а не со всей средой

12) В чем заключается свойство агента - реактивность?

Варианты ответов:

1. Агент может собирать информацию о состоянии внешней среды
2. Агент ощущает внешнюю среду и реагирует на изменения в ней
3. Агент может реагировать на близость к другим агентам

13) В чем заключается свойство агента - проактивность?

Варианты ответов:

1. Агент показывает управляемое целями поведение, проявляя

инициативу
2. Агент выполняет действия, направленные на выполнение цели, независимо от состояний внешней среды
3. Агент взаимодействует с другими агентами для достижения общих целей

14) В чем заключается свойство агента - социальность?
Варианты ответов:
1. Агент передает другим сущностям внешней среды информацию о статусе решения задачи
2. Агент использует информацию, полученную от других агентов, для решения собственных задач
3. Агент взаимодействует с другими сущностями внешней среды для достижения целей

Список литературы

1. *Shoham Y., Leyton-Brown K.* Multiagent Systems: Algorithmic, Game-Theoretic and Logical Foundations — London: Cambridge University Press. 2009. 513 p.

2. Официальный сайт НПК "Генезис знаний" http://www.kg.ru/.

3. *Амелина Н., Лада А., Майоров И., Скобелев П., Царев А.* Исследование моделей организации грузовых перевозок с применением мультиагентной системы адаптивного планирования грузовиков в реальном времени // Проблемы управления. 2011. № 6.

4. *Murray R.M., Astrom K.J., Boyd S.P., Brockett R.W., Stein G.* Future directions in control in an information-rich world // IEEE Control Systems. 2003. Vol. 23. No. 2. P. 20-33.

5. *Городецкий В.И., Грушинский М.С., Хабалов А.В.* Многоагентные системы (обзор) // Новости искусственного интеллекта. 1998. №2. С. 64-116.

6. *Скобелев П. О.* Открытые мультиагентные системы для оперативной обработки информации в процессах принятия решений // Автометрия. 2002. № 6. С. 45-61.

7. *Maes P.* Agent that reduce work and information overload // Communication of the ACM. July 1994. Vol. 37. No. 7. P. 30-40.

8. *Cohen P.R., Morgan J.L., Pollack M.* Intentions in Communication - Bradford Books. MIT Press. 1990. 508 p.

9. Инструкция по разработке приложений Android. Media and Camera. http://developer.android.com/guide/topics/media/camera.

10. Проект JPEG library: http://www.ijg.org/

11. *Кияев В.И., Герасимов Р.В.* Интеллектуальный CRM на базе мультиагентного подхода// В сб. «Стохастическая оптимизация в информатике». Изд-во С.Петербург. ун-та, т. 8, вып. 1, 2012, с. 50-94.

3. Разработка мультиагентных систем

Взаимодействие агентов. Платформы для разработки мультиагентных систем. Примеры реализаций.

3.1. Мультиагентные системы

Система, в которой несколько агентов могут общаться, передавать друг другу некоторую информацию, взаимодействовать между собой и решать поставленную задачу называется мультиагентной (МАС). В МАС задачи (или подзадачи) распределены между агентами, каждый из которых рассматривается как член группы или организации. Распределение задач предполагает назначение ролей каждому из членов группы, определение меры его «ответственности» и требований к «опыту» [2].

Многоагентные системы зародились на пересечении теории систем и распределенного искусственного интеллекта. С одной стороны, речь идет об открытых, активных, развивающихся системах, в которых главное внимание уделяется процессам взаимодействия агентов как причинам возникновения системы с новыми качествами. С другой стороны, достаточно часто МАС строятся как объединение отдельных интеллектуальных систем, основанных на знаниях [1]. МАС обычно состоит из следующих основных компонент:

- множество организационных единиц, в котором выделяются: подмножество агентов, манипулирующих подмножеством объектов;
- множество задач;
- среда, т. е. некоторое пространство, в котором существуют агенты и объекты;
- множество отношений между агентами;
- множество действий агентов (например, операций над объектами).

Основой формой организации взаимодействия между агентами, характеризующаяся объединением их усилий для достижения совместной цели при одновременном разделении между ними функций, ролей и обязанностей является кооперация. В общем случае это понятие можно определить формулой: кооперация = сотрудничество + координация действий + разрешение конфликтов. Под координацией обычно понимается управление зависимостями между действиями. Коммуникация между искусственными агентами зависит от выбранного протокола, который представляет собой множество правил, определяющих, как синтезировать значимые и правильные сообщения. Фундаментальными особенностями группы, составленной из агентов, сотрудничающих для достижения общей цели, являются социальная структура и распределение ролей между агентами.

Основой архитектуры агента является контекст, или серверная среда, в котором он исполняется. Каждый агент имеет постоянный идентификатор – имя. В серверной среде может исполняться не только исходный агент, но и его копия. Агенты способны самостоятельно создавать свои копии, рассылая их по различным серверам для исполнения работы. По прибытии агента на следующий сервер его код и данные переносятся в новый контекст и стираются на предыдущем местонахождении. В новом контексте агент может делать все, что там не запрещено. По окончании работы в контексте агент может переслать себя в другой контекст или по исходящему адресу отправителя. Агенты способны также выключаться («умирать») сами или по команде сервера, который переносит их после этого из контекста в место, предназначенное для хранения.

На рис. 3.1 показана укрупненная структура типичного агента. Входами являются внутренние параметры агента и данные о состоянии среды. Выходы – параметры, воздействующие на среду и информирующие пользователя (или программу, выполняющую роль менеджера в системе) о состоянии среды и принятых решениях. Решатель – процедура принятия решений. Решатель может быть достаточно простым алгоритмом или элементом системы искусственного интеллекта.

Рис. 3.1. Укрупнённая структура агента.

Рис. 3.2 - Архитектура ядра мультиагентной системы.

В архитектуре МАС основную часть составляет предметно-независимое ядро, в составе которого выделяются следующие базовые компоненты (рис. 3.2):

- служба прямого доступа обеспечивает непосредственный доступ к атрибутам агентов;
- служба сообщений отвечает за передачу сообщений между самим агентами, а также между агентами и дополнительными системами ядра;
- библиотека классов агентов (часть базы знаний) содержит информацию о классификации агентов в данной МАС.

- сообщество агентов – серверное «место», где размещаются агенты; этот блок, кроме жизнедеятельности агентов, обеспечивает еще функции по загрузке/записи агентов и их свойств и за оптимизацию работы агентов с ресурсами.
- онтология – предметная база знаний, содержащая конкретные знания об объектах и среде функционирования, представляемые в виде соответствующей семантической сети.

Общая методология восходящего эволюционного проектирования МАС может быть представлена цепочкой: <среда – функции МАС – роли агентов – отношения между агентами – базовые структуры МАС – модификации>, и включает следующие этапы:

- формулирование назначения (цели разработки) МАС;
- определение основных и вспомогательных функций агентов в МАС;
- уточнение состава агентов и распределение функций между агентами, выбор архитектуры агентов;
- выделение базовых взаимосвязей (отношений) между агентами в МАС ;
- определение возможных действий (операций) агентов;
- анализ реальных текущих или предполагаемых изменений внешней среды.

При проектировании организацию агентов можно рассматривать как набор ролей, находящихся между собой в определенном отношении, и взаимодействующих друг с другом. Таким образом, методология восходящего проектирования МАС требует предварительного задания исходных функций (ролей агентов), определения круга их обязательств по отношению друг к другу, формирования исходных и развивающихся структур на основе выделенных функций и исследования адекватности этих структур характеру решаемых задач в выделенных проблемных областях

Главная идея нисходящего проектирования состоит в определении общих социальных характеристик МАС по некоторому набору критериев, построении базовых типов их организаций с последующим определением требований к архитектуре агентов. Когда речь идет о «выращивании» искусственных социальных систем и сообществ, на первый план выдвигается нисходящий подход к организационному проектированию.

В рассмотренных ниже примерах, безусловно, наиболее подходящим является проектирование на основе восходящего подхода.

3.2. Современные международные стандарты создания агентов и платформы МАС

Существует несколько международных подходов к созданию мультиагентных систем, наиболее известные их них – это OMG MASIF, созданный Object Management Group, в основе которого лежит понятие мобильный агент; спецификации FIPA (Foundations for Intelligent Physical Agents), основанные на предположении об интеллектуальности агента, а также стандарты, разработанные исследовательским подразделением Пентагона – Агентством Передовых Оборонных Научных Исследований (Defense Advanced Research Projects Agency – DARPA), в частности Control of Agent Based Systems.

Относительно мобильности и интеллектуальности агентов, большинство специалистов сходятся на том, что мобильность – центральная характеристика агента, интеллектуальность – желаемая, но не всегда строго требуемая. Различия подходов к определению агента в стандарте FIPA и OMG представлены на рис. 3.2.

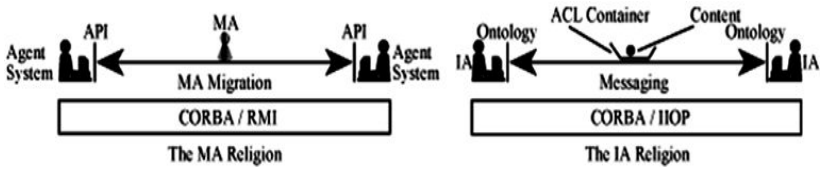


Рис. 3.3. Сравнение мобильных и интеллектуальных агентов.

Деятельность FIPA заключается в совместном исследовании и разработке членами организации международных согласованных спецификаций, которые позволят максимизировать взаимодействие между агентными приложениями, услугами и оборудованием. Членами FIPA являются такие высокотехнологичные компании как Alcatel, Boeing, British Telecom, Deutsche Telekom, France Telecom, Fujitsu, Hitatchi, HP, IBM, Fujitsu, Hewlett Packard, IBM, Intel,

Lucent, NEC, NHK, NTT, Nortel, Siemens, SUN, Telia, Toshiba, различные университеты, государственные организации.

Спецификации FIPA ориентируются на обеспечение возможности взаимодействия интеллектуальных агентов через стандартизированную коммуникацию агентов и языки контента. Наряду с общими основами коммуникации FIPA специализируется также на протоколах онтологии и переговоров для поддержки взаимодействия в конкретных прикладных сферах (транспортная поддержка, производство, мультимедиа, поддержка сетевого взаимодействия).

Стандарт OMG MASIF нацелен на создание условий для миграции мобильных агентов между мультиагентными системами посредством стандартизированных интерфейсов CORBA IDL.

Организация DARPA инициировала работу по распределению знаний (Knowledge Sharing Effort), в результате которой языки программирования агентов были разделены на синтакс (syntax), семантику (semantics) и прагматику (pragmatics):

- KIF – Knowledge Interchange Format (syntax);
- Ontolingua – a language for defining sharable ontologies (semantics);
- KQML (Knowledge Query and Manipulation Language) – a high-level interaction language (pragmatics).

Важным элементом при создании мультиагентных систем является язык коммуникации агентов – Agent Communication Language, который определяет типы сообщений, которыми могут обмениваться агенты. В рамках парадигмы коммуникации между агентами, кооперация между ними достигается за счет ACL, языка контента и онтологии, которые определяют набор базовых концепций, используемых в сообщениях кооперации. Онтология здесь выступает синонимом понятия API (Application Programming Interface), т.е. она определяет конкретный интерфейс интеллектуальных агентов.

На техническом уровне коммуникация между агентами происходит за счет передачи сообщений используя какой-либо транспортный протокол нижнего уровня (SMTP,TCP/IP, HTTP, IIOP). Альтернативами к использованию ACL является ряд других языков, таких как языки БД (SQL), Distributed object systems (CORBA и др.),

Service languages (e-speak от Hewlett Packard, BizTalk от Microsoft и др.) и Web languages (XML, RDF, DAML).

Еще одной альтернативой ACL является CORBA ORB, разработанный уже упоминавшийся Object Management Group. Вся функциональность, предоставляемая CORBA доступна и на языке JAVA, путем комбинации Java RMI, Java RMI servers, Jini, Java event servers и других.

В настоящее время языки коммуникации агентов продолжают эволюционировать. Поскольку совместимость – определяющая характеристика агентов, при разработке MAS – очень важна именно стандартизированная коммуникативность. Основными объектами для стандартизации являются: архитектура агента, языки взаимодействия агентов, протоколы взаимодействия агентов, знания агентов, языки программирования агентов.

Как отмечают эксперты в области разработки агентов, для последующей эволюции технологий создания агентов необходимы следующие действия:

- развитие семантики языков коммуникации агентов (ACL) (общих языков контента и онтологии; языков для описания действий агентов, намерений и стремлений);
- развитие онтологии агентов (разделяемые онтологии для свойств агентов и их поведения);
- улучшение использования метаданных (абстрактное и совмещаемое со многими языками контента);
- декларативные и ясные протоколы (языки для определения протоколов высокого уровня, базирующиеся на более примитивных);
- практический обмен знаниями между агентами (социальные механизмы для обмена информацией и знаниями, рассмотрение обмена знаниями как мобильный код);
- развитие схем и методов для контроля за системами агентов (искусственные рынки, естественный отбор и т. п.).

Агентные платформы представляют собой один из способов построения распределенных систем и позволяют описать и предоставить доступ всех приложений, работающих на агентной платформе к необходимым им сервисам. Кроме того, в функции агентной

платформы входит распределение агентов, аудит их функционирования и управление.

На данный момент известно несколько агентных платформ, ориентированных на использование спецификации FIPA-2000 (табл. 3.1) (Bellifemine F., 1999; Willmott S. N., 2000; Burg B., 2001).

Таблица 3.1.

Компания	Агентная платформа	Адрес в Интернете (URL)
BTexact Technologies (Великобритания)	ZEUS	http://www.labs.bt.com/projects/agents/zeus/
Comtec (Япония)	Comtec Agent Platform	http://fipa.comtec.co.jp/glointe.htm
CSELT (Италия)	JADE	http://jade.tilab.com/
Fujitsu Labs (США)	AAP	http://www.sourceforge.net/
Nortel Networks (Великобритания)	FIPA-OS	www.nortelnetworks.com/fipa-os

Агентная платформа в стандартах FIPA представляет собой следующую конструкцию (рис. 3.4):

Рис. 3.4. FIPA-модель агентной платформы.

Система управления агентами (СУА) представляет собой также агента, который осуществляет контроль доступа и использования агентной платформы. В каждой агентной платформе присутствует одна СУА, которая предоставляет сервис жизненного цикла программных агентов и их реестр с идентификаторами, а также содержит состояния каждого программного агента. Маршрутизатором каталога является программный агент, который обеспечивает направление запросов в другие агентные платформы. Система транспортировки сообщений, или канал коммуникации агентов, является программным компонентом для управления потоками сообщений, приходящих на агентную платформу.

3.3. Платформы для разработки МАС

Перед тем, как приступать к созданию приложений на Android, необходимо выбрать подходящий инструментарий разработки и установить соответствующий Android SDK.

Наиболее популярные средства разработки МАС следующие:

- JADE (Java Agent Development Framework) — широко используемая программная среда для создания мультиагентных систем и приложений, поддерживающая FIPA-стандарты для интеллектуальных агентов. Включает в себя среду выполнения агентов (агенты регистрируются и работают под управлением среды), библиотеку классов, которые используются для разработки агентных систем, набор графических утилит для администрирования и наблюдения за жизнедеятельностью активных агентов. Программная среда JADE подключается к любом проекту на языке Java. Агенты JADE могут быть совершенно разными — от простых, только реагирующих, до сложных — ментальных.
- JACK Intelligent Agents — Java платформа для создания мультиагентных систем. Так же как и JADE, расширяет Java своими классами. JACK одна из немногих платформ, где используются модель логики агентов, основанная убеждениях-желаниях-намерениях (Belief–desire–intention software model – BDI), и встроенные формально-логические средства планирования работы агентов.
- MadKIT — модульная и масштабируемая мультиагентная платформа, написанная на Java. Поддерживает агентов на разных языках: Java, Python, Jess, Scheme, BeanSchell. Красиво визуализирует и позволяет управлять этими агентами.
- AgentBuilder — большой коммерческий продукт, выпускаемый так же и в Academic Edition. Агенты достаточно интеллектуальны, и общаются на языке KQML (Knowledge Query and Manipulation Language) и обладают ментальной моделью. Платформа является Java-ориентированной.
- Cougaar (Cognitive Agent Architecture) — также Java-ориентированная платформа для построения распределенных мультиагентных систем. Включает не только исполняющую систему

(run-time engine), но и некоторые средства для визуализации, управления данными и др.

- NetLogo — кроссплатформенное программируемое окружение для программирования мультиагентных систем.
- VisualBots — бесплатный мультагентный симулятор в Microsoft Excel с Visual Basic синтаксисом.
- MASON — Java библиотека для моделирования мультиагентных систем.
- REPAST — набор инструментов для создания систем, основанных на агентах.
- CogniTAO — C++ платформа разработки автономных мультиагентных систем, ориентированная на реальных роботов и виртуальных существ (CGF).

3.4. Области применения

На сегодняшний день мультиагентные системы используются для разработки широкого спектра информационных и промышленных систем. В промышленности МАС наиболее распространены применительно к решению задач автоматизации управления сложными системами, для сбора и обработки информации, в играх. Мультиагентные технологии применимы в управлении мобильными ресурсами, а также в таких сферах, как проектирование объектов, промышленное производство, финансовое планирование и анализ рисков, распознавание образов, извлечение знаний из данных, понимание текста и решение других сложных проблем.

Так, например, IBM использует агентов для производства полупроводниковых микросхем, датская судостроительная компания – для заварки отверстий в кораблях, а в Японии система на базе агентов выполняет функции интерфейса оператора сверхскоростного поезда

МАС могут применяться как для конструирования и моделирования гибких производственных систем, так и для управления реальными системами производства (логистика), продажи продукции различного назначения (е-коммерции), интеграции и управления знаниями и научной работы. Большое значение в мультиагентном

подходе имеет социальный аспект решения современных задач как его концептуальная основа.. Такие системы должны постоянно «жить» на сервере предприятия и непрерывно участвовать в решении задач, а не быть запускаемыми от случая к случаю, а для этого – обеспечивать пользователю возможность введения новых данных и компонентов. Наконец, такие системы должны накапливать информацию, извлекать из нее новые знания и в зависимости от этого изменять свое поведение с течением времени.

В настоящее время интеллектуальные агенты применяются в следующих областях бизнеса:

- управление распределенными или сетевыми предприятиями;
- сложная и многофункциональная логистика;
- виртуальные организации и Интернет-порталы по продаже продуктов и услуг;
- управление учебным процессом в системах дистанционного обучения;
- компании с развитыми дистрибьюторскими и транспортными сетями (например, в Procter&Gamble);
- управление каналами распределения;
- моделирование предпочтений пользователей.

Для крупных компаний преимущества мультиагентного подхода очевидны. Среди них можно отметить: сокращение сроков решения проблем, уменьшение объема передаваемых данных за счет передачи другим агентам высокоуровневых частичных решений; сокращение сроков согласования условий и формирования заказов.

Для распределенных компаний преимущества в первую очередь заключаются в возможности оптимального обеспечения продукцией, облегчении контроля удаленных подразделений и структур и взаимодействия с ними.

Для компаний с широким и быстро меняющимся ассортиментом – возможность гибко реагировать на изменения в предпочтениях клиентов и просчитывать периоды изменения. Для компаний оказывающих услуги – накопление опыта взаимодействия и решения проблем не только «в головах» сотрудников, но и в МАС.

Среди примеров компьютерных программ-агентов, существующих в настоящее время и широко используемых в Интернете можно выделить следующие:

Copernic Agent (http://www.copernic.com/) – одновременно отправляет запросы нескольким популярным поисковым системам, выбирает наиболее рейтинговые ссылки, сопоставляет их между собой, удаляет дубли и, сортируя отобранное по рейтингу в соответствии со своим алгоритмом ранжирования, выводит их пользователю.

MySimon (http://www.mysimon.com/) – осуществляет интеллектуальный поиск, сравнивая цены миллионов товаров в более чем двух тысячах онлайновых магазинов.

MP3-Wolf (http://www.trellian.com/) – сканирует Интернет в поисках нужных пользователю музыкальных файлов. В процессе работы он использует различные поисковые системы, а также сайты, найденные им ранее и содержащиеся в его базе.

WebSite-Watcher (http://www.aignes.com/) – предназначена для слежения за изменениями на сайтах. Поддерживает работу RSS-ленты. Имеет гибкие настройки по предотвращению ложных срабатываний, когда отдельные изменения на страницах носят случайный или технический характер, например изменение числа просмотров.

Помимо этого, агенты могут быть уполномоченными представителями пользователя при общении с другими пользователями или их агентами, при решении порученных им задач.

Для решения задач автоматизации управления ресурсами предприятий в реальном времени в последнее время разрабатывается большое количество интеллектуальных программных систем нового поколения, построенных на основе мультиагентных технологий, которые позволяют автоматизировать полный цикл управления мобильными ресурсами в реальном времени, включая:

- оперативную реакцию на важные события;
- динамическое планирование и адаптивное перепланирование заказов/ресурсов;

- взаимодействие с клиентами, менеджерами и исполнителями для согласования принимаемых решений через Интернет или сотовый телефон;
- мониторинг исполнения построенных планов и бизнес-процессов заказчика;
- перепланирование расписаний в случае рассогласования между планом и фактом.

В таких системах агенты способны взаимодействовать друг с другом путем переговоров и демонстрировать коллективный интеллект, возникающий в системе в форме спонтанных цепочек согласованных изменений планов агентов.

Таким образом, в разработке программных систем используются фундаментальные принципы самоорганизации и эволюции, присущие живым системам, например, колонии муравьев или рою пчел, отличающихся способностью решать сложные задачи в реальном времени, открытостью к изменениям, высокой эффективностью, надежностью и живучестью.

Обычно такие системы легко интегрируются с существующими коммуникаторами, учетно-контрольными системами предприятия, электронными картами, средствами GPS навигации, RFID-чипами и т.п.

Применение мультиагентных инструментальных средств позволяет решить ряд сложных задач производственной и транспортной логистики [3]. В [4] описаны промышленные реализации нескольких интеллектуальных систем, основанных на мультиагентных технологиях: для компаний занимающихся междугородними транспортными перевозками (организовано согласованное планирование парка грузовиков для менеджеров центра и филиалов), для компании корпоративного такси (планирование и распределение текущих заказов в режиме реально времени), для аэропорта (система управления наземными сервисами аэропорта на основе РФИД технологий). Для решения этих задач использовалась инструментальная платформа, которая состоит из основных модулей, представленных на Рис. 3.1.

Детектор образов распознает типовые ситуации, возникающие в ходе поступления заявок и вырабатывает прогноз заявок и реко-

мендации по планированию с учетом предыстории. **Адаптивный планировщик** обрабатывает поток входящих событий (поступлений заявок, ввода новых ресурсов, выхода из строя ресурсов и т. п.).

Рис. 3.1: Схема взаимодействия модулей инструментальной системы [4].

Конструктор сцены позволяет редактировать начальную конфигурацию сети и определить все параметры ресурсов компании. Конструктор сцены основывается на общей базе знаний *(онтологии)*, описывающей деятельность компании, в которой присутствуют базовые понятия и отношения между ними, и которая при развитии бизнеса может расширяться с использованием редактора онтологии.

Редактор онтологии позволяет ввести и изменить общую онтологию компании, описывающую модель знаний предметной области, которая затем применяется в редакторе сети для описания конфигурации бизнеса. Онтология содержит базовые знания и отношения между ними, представляемые в форме семантической сети.

Моделирующая система — программный модуль, позволяющий осуществлять моделирование ситуации по принципу "Что если?".

Эволюционный дизайн — модуль, вырабатывающий предложения по улучшению конфигурации сети в части увеличения или

уменьшения определенного числа ресурсов, изменению географии ресурсов и т. д. [4].

Приведём несколько примеров построения МАС для решения конкретных управленческих задач.

МАС для распределения заказов такси. Для одной из крупнейших в мире компаний корпоративного такси *Addison Lee* (Лондон) была разработана система, которая позволила распределять и планировать примерно 13 тысяч заказов в день при наличии нескольких тысяч собственных машин (из них до 800 постоянно на линии), оснащенных средствами GPS-навигации. При появлении нового заказа система автоматически находит наилучшую машину, получая сведения о координатах ближайших машин на электронной карте Лондона, и предварительно бронирует заказ. Если эта машина была уже занята, то начинается цепочка переговоров, направленная на разрешении возникшего конфликта и достижение компромисса, что позволит перебросить старый заказ на другую машину, если это выгодно для всех. Но и после этого работа системы с новым заказом не останавливается.

В среднем на подачу машины требуется около 15 минут, при этом примерно половину этого времени система продолжает непрерывно искать возможности для улучшения перевозки с учетом поступающих заказов и появляющихся новых ресурсов и не принимает окончательного решения до момента, когда необходимо отправлять машину с учетом времени пути проезда до заказа. Когда уже пора отправлять автомобиль на заказ, система принимает окончательное решение, посылает водителю сообщение о параметрах заказа и ждет подтверждения о приеме заказа [4].

Внедрение такой системы дало возможность роста компании за счет повышения управляемости. Существенно увеличилась эффективность автопарка (10-15%) благодаря оптимальному распределению заказов за счет минимизации "холостого" пробега и времени простоя автомобилей. Также сократилось количество опозданий и время обслуживания заявок.

МАС для управления группами интеллектуальных роботов. Хорошим примером интеллектуальных агентов служат роботы. Роботы могут иметь широкий ассортимент искусственных органов чувств (сенсорные датчики) и искусственных эффекторов (манипуляторы, педипуляторы). Их мобильность достигается благодаря колесным, гусеничным, шагающим и прочим системам перемещения. Активность и автономность роботов тесно связаны с наличием средств целеполагания и планирования действий, систем поддержки решения задач, а интеллектуализация, помимо обладания системой обработки знаний, предполагает развитые средства коммуникации различных уровней, вплоть до средств естественноязыкового общения [1].

Неотъемлемым атрибутом интеллектуальных роботов является наличие специальной подсистемы планирования, составляющей программу действий робота в реальных условиях окружающей среды, которые определяются рецепторами робота. Для планирования деятельности робот должен иметь знания о свойствах окружающей среды и путях достижения целей в этой среде.

Рассмотрим два подхода к управлению группой интеллектуальных роботов.

Первый основан на рыночных отношениях в группе роботов. Его суть заключается в следующем. Сначала каждый робот генерирует список целей. Цели, соответствующие известным областям среды, исключаются из списков, а оставшиеся размещаются в порядке следования. Каждый робот, по очереди, пытается продать каждую из целевых задач всем остальным роботам группы, с которыми возможна связь, выставляя их на аукцион. Каждый из остальных роботов предлагает цену, которая определяется предположительными расходами и доходами. Робот-аукционист, предлагающий целевую задачу, ждет, пока все роботы не предложат цену. Причем на это отводится определенное время. Если какие-либо роботы предлагают цену, большую чем минимум, предложенный роботом-аукционистом, то целевая задача передается тому их них, который предлагает наибольшую цену.

После того, как все аукционы проведены, роботы начинают движение к первой своей цели. По ее достижении каждый робот генерирует новые цели и начинает движение к следующей цели, содер-

жащейся в задании, предлагая все оставшиеся в списке целевые задачи остальным роботам группы через аукцион. Однако в этом подходе используются очень громоздкие механизмы аукционов и регулирования цен, что существенно ограничивает его применение.

Другой подход к организации мультиагентного взаимодействия в группах интеллектуальных роботов основан на принципах коллективного управления [5], характерных для коллективов людей.

Коллективом называют группу роботов, решающих общую целевую задачу и взаимодействующих между собой для решения этой задачи наилучшим для группы образом. Коллективное взаимодействие — это взаимодействие, охватывающее большое число элементов некоторой системы и проявляющееся в их согласованных действиях.

Метод коллективного управления заключается в том, что каждый робот группы, во-первых, самостоятельно управляет процессом своего функционирования, т. е. определяет свои действия, а во-вторых, согласовывает эти действия с действиями других роботов группы, для того чтобы наиболее эффективно, т. е. с минимальными затратами и максимальной выгодой для группы, решить целевую задачу.

Основные принципы коллективного управления [5]:

- каждый член коллектива группы самостоятельно формирует свое управление (определяет свои действия) в текущей ситуации;
- формирование управлений (выбор действий) каждым членом коллектива осуществляется только на основе информации о коллективной цели, стоящей перед группой, ситуации в среде в предыдущей отрезок и в текущий момент времени, своем текущем состоянии и текущих действиях других членов коллектива;
- в качестве оптимального управления (действия) каждого члена коллектива в текущей ситуации понимается такое управление (действие), которое вносит максимально возможный вклад в достижение общей (коллективной) цели или, иными словами, дает максимально возможное приращение целевого функционала при переходе системы "коллектив-среда" из текущего состояния в конечное;

- оптимальное управление реализуется членами коллектива в течение ближайшего отрезка времени в будущем, а затем определяется новое управление;
- допускается принятие компромиссных решений, удовлетворяющих всех членов коллектива, то есть каждый член коллектива может отказываться от действий, приносящих ему максимальную выгоду, если эти действия приносят малые выгоды или даже ущерб коллективу в целом.

В отличие от группового управления, которое может быть как централизованным, так и децентрализованным, коллективное управление группой роботов всегда децентрализованное по своей сути. Поэтому описанный метод коллективного управления роботами наиболее эффективен при реализации в распределенных мультиагентных системах. Основное преимущество — относительно низкая вычислительная сложность алгоритмов, что позволяет быстро принимать если не оптимальные, то близкие к ним решения в условиях динамически изменяющейся ситуации. Этот подход применяется не только для решения задач коллективного управления интеллектуальными роботами, но и для решения проблемы организации отказоустойчивого вычислительного процесса в распределенных многопроцессорных информационно-управляющих системах сложных динамических объектов [5].

МАС для управления безопасностью автомобиля. Примером инновационного продукта, обеспечивающего комплексную безопасность автомобиля на дороге может служить информационно-диагностическая подсистема "Автоматизированная помощь на дороге" (АПД) для on-line обслуживания водителей [13].

Подсистема предназначена для постоянного текущего мониторинга устройств и механизмов автомобиля клиента и базируется на современных достижениях в области сбора, передачи, и автоматизированного анализа сигналов, обработки сигналов в режиме on-line с использованием современных датчиков и контроллеров, а также контроля всей системы на базе мультиагентного подхода. Пользователем может стать любой человек, который пожелает установить систему АПД при покупке нового автомобиля распространенных марок Ford, Opel, Chevrolet, KIA, BMW и др.

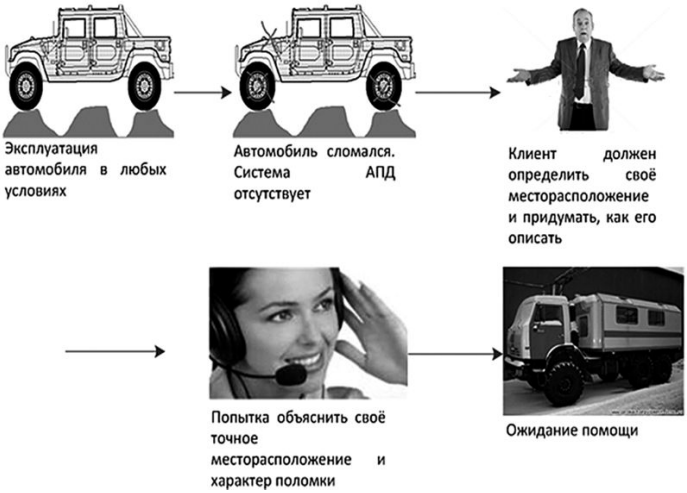


Рис. 3.2. Эксплуатация автомобиля, не оборудованного системой АПД.

Главным назначением АПД является повышение безопасности клиента. Во время движения автомобиля могут возникать нештатные ситуации, связанные с состоянием агрегатов автомобиля, дорожной обстановкой, состоянием покрытия, погодой и т.д. При этом водитель может не подозревать о надвигающейся аварийной ситуации, связанной с некорректной работой двигателя, тормозной системы, рулевого управления. Система АПД обязана минимизировать последствия критических поломок, так как она призвана при первых симптомах неисправности, отправить сведения о них в дилерский центр, оператор которого свяжется с клиентом и сообщит о возможных последствиях и путях их предотвращения. Рассмотрим вариант, когда автомобиль не оборудован системой АПД (рис. 3.2.) Клиент эксплуатирует автомобиль определённое количество времени, проводит регламентированные работы (ТО) в официальном автосервисе. В случае инцидента на дороге (отказ двигателя, съезд с полотна и переворот вследствие прокола колеса, наезд на препятствие, столкновение с другим транспортным средством) водитель обязан вызвать сотрудников ГИББД, страхового агента и т.д. Но если водитель серьёзно пострадал, то он не может определить своё

состояние, местоположение, вызвать себе скорую помощь и соответствующие службы. И если ДТП произошло на просёлочной дороге, то шансы быстро получить необходимую помощь резко уменьшаются.

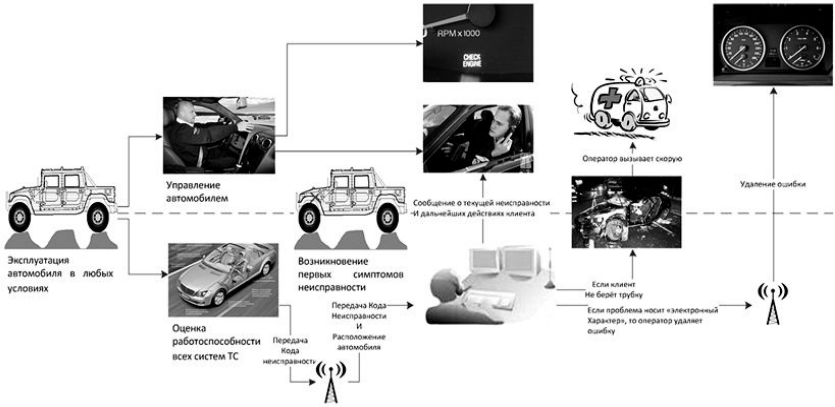


Рис. 3.3. Эксплуатация автомобиля, оборудованного системой АПД.

Если же автомобиль оснащен системой оказания комплексной услуги АПД (рис. 3.3), то при возникновении малейшей неисправности (например, нештатная вибрация кузова, рулевого управления, утечка масла или бензина, критический износ тормозных колодок и пр.) система отображает возникшую коллизию на панели приборов. Бортовой компьютер отсылает код неисправности оператору в дилерский центр или станцию технического обслуживания. Оператор немедленно связывается с дежурным мастером-консультантом, а тот, в свою очередь, удаленно тестирует неисправность, связывается с клиентом и даёт рекомендации по поводу дальнейшего эксплуатирования автомобиля. Если клиент попадает в ДТП, то компьютер автоматически отсылает информацию о том, что сработали ремни и подушки безопасности. Оператор сразу же пытается связаться с клиентом по сотовому телефону. Если клиент не снимает трубку, то оператор вызывает скорую помощь на место, которое было определено с помощью GPS-передатчика на автомобиле.

Таким образом, система АПД позволяет повысить безопасность благодаря своевременному выявлению проблем, а также вызову

помощи тогда, когда клиент в ней нуждается. Помимо всего прочего, система АПД может иметь ряд других полезных функций. Так например, если клиент оставил ключи от автомобиля дома, но очень спешит, то он может позвонить оператору, назвать свой уникальный ID и пароль. Оператор открывает и заводит автомобиль. Также оператор может закрыть автомобиль.

Система АПД может выполнять также функцию охранной сигнализации: если злоумышленники угнали автомобиль, клиент связывается с оператором, а тот, в свою очередь, связывается с правоохранительными органами и сообщает им расположение автомобиля.

Архитектура системы АПД может состоять состоит из трёх базовых модулей (рис. 3.4):

1. модуль диагностики автомобиля;
2. программный модуль коммуникатора;
3. АРМ оператора у дилера.

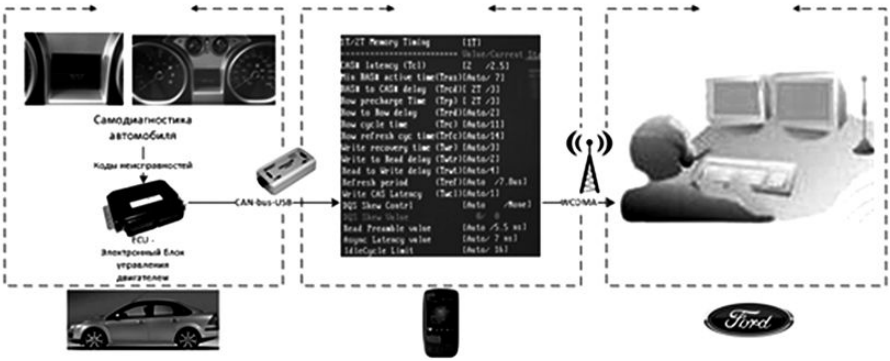


Рис. 3.4. Архитектура системы АПД.

Модуль диагностики автомобиля включает в себя систему самодиагностики и блоки ECU (Electronic Control Unit). «Самодиагностика (иногда называемая бортовой диагностикой) – это система, которая постоянно держит под наблюдением сигналы различных датчиков и исполнительных механизмов *системы управления двигателем* (СУД). Эти сигналы сравниваются с их контрольными значениями, которые хранятся в памяти бортового компьютера.

Набор таких контрольных значений может быть разным в разных автомобилях и их моделях. Он может в себя включать верхние и нижние допустимые границы контролируемых параметров, допустимое число ошибочных сигналов в единицу времени, неправдоподобные сигналы, сигналы, выходящие за допустимые пределы и др. При выходе сигнала за пределы контрольных значений блок электронного управления квалифицирует это состояние как неисправность, формирует и помещает в память соответствующий код.

Ранние конструкции систем диагностики были способны формировать и хранить лишь небольшое число кодов. Современные системы в состоянии генерировать и хранить 100 и более кодов и способны ещё увеличить это количество по мере того, как ПО бортовых компьютеров, основанное на технологии интеллектуальных агентов "научится" выделять новые нештатные, сбойные и аварийные ситуации.

ECU – основной элемент электронной системы управления, реализующий алгоритм управления путём преобразования электрических сигналов датчиков в управляющие воздействия на исполнительные устройства. Электронный блок управления (ЭБУ) двигателем получает сигналы от разных датчиков и обрабатывает их по определённой схеме. После обработки данных принимает решение о том, как управлять работой различных систем двигателя, меняя подачу топлива, выставляя зажигание, а также диагностирует все системы управления автомобилем и уведомляет о неисправностях.

Связь внешнего устройства (в данном случае коммуникатора) с электронным блоком управления осуществляется благодаря CAN-шине. CAN (Contoller Area Network) является сетевым интерфейсом передачи данных на скорости до 1 Мбит/сек. Автомобиль связывается с коммуникатором посредством CAN-шины и переходника CAN-bus-USB. На смартфоне водителя устанавливается ПО, которое способно выводить информацию о работе систем автомобиля на дисплей, а также отправлять их в Интернет. В случае, если ошибок никаких не возникает, то сведения о работе автомобиля остаются в телефоне. В случае, если возникает какая-либо критическая неисправность либо срабатывают датчики ремней и подушки безопасности), то информация о местоположении автомобиля, а также ин-

формация с кодом неисправности отсылается на АРМ оператора соответствующего Call-центра.

В соответствии с такой архитектурой, алгоритм сбора, анализа, и выработки рекомендаций для пользователя будет основан на мультиагентном подходе. Суть мультиагентного подхода в этом случае заключается в том, что гораздо эффективнее внедрить несколько интеллектуальных агентов, которые будут отвечать за свой небольшой сегмент, чем использовать одно электронное устройство с заданной жестко программой функционирования, отвечающее за всю диагностику (в настоящее время вся информация с датчиков поступает в "головной центр" сбора информации ECU).

Каждый агент содержит базу знаний, которая распространяется только на один элемент автомобиля. Например, первый агент отвечает за работу включения и выключения атвомобиля, другой агент отслеживает работу тормозной системы, третий агент контролирует работу двигателя, четвёртый агент следит за покрытием дороги, и т.д.

Рассмотрим два алгоритма работы системы на базе мультиагентного подхода и опишем типичную схему принятия решения.

- На улице зима, человек садится в автомобиль, прогревает его (режим контролирует 1-й агент) и выезжает на шоссе. Во время движения из-за снега и наледи постоянно меняется рельеф дорожного покрытия, и человек не всегда успевает реагировать на эти изменения. Поэтому агент, который отвечает за анализ состояния дорожного покрытия (4-й агент), постоянно общается с агентами, которые отвечают за работу двигателя (3-й агент) и тормозной системы (2-й агент). Если автомобиль выезжает на лёд, то четвертый агент сообщает об этом первому и второому. Второй агент приводит тормозную систему в состояние повышенный готовности (усиливает давление в тормозной системе), а первый агент ограничивает обороты и плавность работы двигателя. Тем самым уменьшается риск заноса или, другими словами, повышается уровень безопасности.
- Автомобиль двигается по дороге. Соответствующие интеллектуальные агенты следят за состоянием элементов активной безопасности (рис. 3.5). Первый агент отвечает за состояние

ремней безопасности, второй агент следит за состоянием подушек безопасности, третий отслеживает состояние тормозной системы, четвёртый передаёт данные на коммуникатор, пятый отправляет информацию на АРМ оператора дилерского центра, шестой агент отвечает за информацию поступающих с датчиков парковки (патронников), седьмой агент следит за работой климатической установки. Автомобиль подъезжает к пешеходному переходу, но водитель задумался, и не успевает своевременно плавно остановиться и резко нажимает на педаль тормоза. Агенты анализирует изменившуюся информацию (автомобиль ехал с постоянной скоростью, но резко остановился), общаются между собой: шестой агент сообщает, что контакта не было, второй сообщает, что подушки безопасности не раскрылись, первый агент говорит о том, что преднатяжители ремней безопасности сработали.

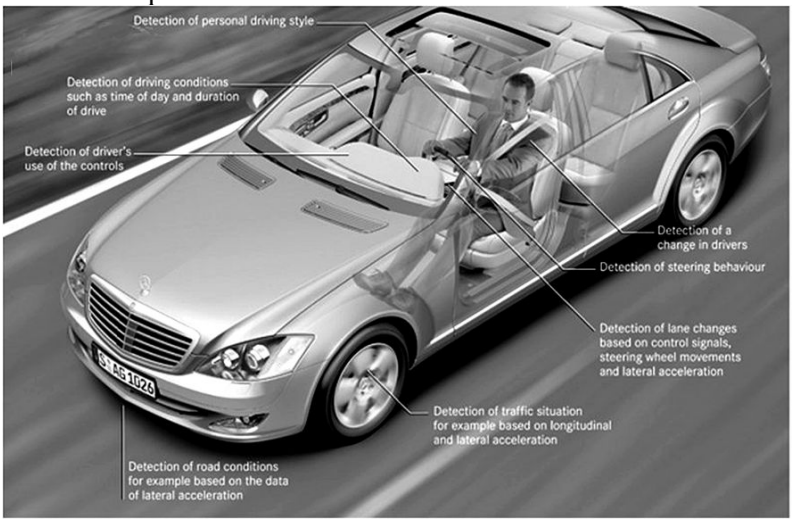


Рис. 3.5 Схема анализа информации, полученной агентами от датчиков.

Исходя из этих данных, они решают не отправлять данные на АРМ оператора дилерского центра, но занести информацию о характере торможения в память (для того, чтобы при возникновении неполадки в будущем, можно было легко вычислить причину её возникновения). Автомобиль продолжает движение и совершает

наезд на впереди стоящий автомобиль. И тут уже начинается рассмотрение как минимум двух ситуаций:

a. Агенты сообщают, что патронники зафиксировали критическое приближение (удар), преднатяжители ремней сработали, но температура в салоне не изменилась (лобовое стекло цело) и подушки безопасности не открылись, т.е. удар произошёл на небольшой скорости. Как вывод, агенты сообщают эту информацию на АРМ оператора дилерского центра, и он связывается с клиентом.

b. Агенты сообщают о срабатывании всех систем активной безопасности: преднатяжители сработали, зафиксировано критическое сближение (удар), подушки безопасности раскрылись, температура в салоне изменилась – т.е. произошёл один из худших вариантов развития для человека. Агенты автомобиля сообщают информацию о случившемся агенту, который отвечает за коммуникатор, далее этот агент передаёт информацию о происшествии и местоположении автомобиля на АРМ оператора дилерского центра. Оператор быстро обрабатывает всю информацию, пытается связать с клиентом и в случае необходимости вызывает скорую помощь на место происшествия.

Таким образом, видно, что агенты общаются между собой и при анализе поступающей информации принимают решение о реализации необходимых процедур.

Устройство управления самодиагностикой состоит из нескольких выделенных блоков: для двигателя – ECU (Engine Control Unit) или ECM (Engine Control Module); для антиблокировочной системы тормозов – ABS (Antiblocking System), для подушек безопасности – ABSRS (Air Bag Supplemental Restraint System), для автоматической коробки передач – A/T (Electronic Automatic Transaxles) и др.

При получении сигнала о возникновении неполадки система диагностики обязана ответить унифицировано:

- во-первых, классифицировать неисправность по номеру (коду ошибки) и запомнить этот код в долговременной памяти;
- во-вторых, предпринять действия по минимизации ущерба, предусмотренные на этот случай управляющей программой.

После этого сохранённые в памяти коды ошибок считываются специальным прибором (сканером), или вручную, при помощи оп-

ределённой процедуры, которая вводит ECU в режим индикации кодов самодиагностики.

МАС для управления группой БПЛА. Еще одним перспективным направлением применения МАС являются системы беспилотных летательных аппаратов. В современном мире "беспилотные летательные аппараты" (англ. Unmanned Aerial Vehicles, далее БПЛА) приобретают все большую популярность в качестве легких и недорогих инструментов для исследований территорий, разведки и воздушных съемок. В основной массе современная беспилотная техника не обладает автономностью (обычно дистанционно управляется оператором со специального пульта). Есть успешные разработки одиночных БПЛА, которые под управлением автопилота способны в автономном режиме облететь территорию по заданному маршруту, собирая ту или иную информацию или выполняя другие задания. Анализ большинства задач решаемых с использованием одиночных аппаратов показывает, что они могут более эффективно решаться группой, так как у группы эффективно взаимодействующих легких и маленьких самолетов появляются дополнительные полезные свойства [6-8]. В следующей главе будет рассмотрен один из возможных вариантов реализации МАС для управления группой БПЛА.

3.5. Лабораторная работа № 3 «Знакомство с JADE»

3.5.1. Цель лабораторной работы

Познакомится с мультиагентной платформой JADE, установить платформу на рабочий компьютер, освоить протокол передачи данных между агентами на платформе, запуск главного контейнера платформы и нового агент, на примере агентов «PingAgent» - запрос-ответ.

3.5.2. Инструкция по выполнению лабораторной работы

3.5.2.1. Подготовка

Перед началом работы необходимо запустить среду разработки Eclipse IDE, загрузить и установить JDK и JADE. Запустить платформу можно двумя способами: из командной строки или в среде разработки. Как и в предыдущих лабораторных работах мы будем использовать среду разработки Eclipse IDE. Для запуска из командной строке необходимо обратится к инструкции на официальном сайте платформы JADE по ссылке: http://jade.tilab.com/doc/index.html.

Для начала знакомства с JADE необходимо установить JDK и запустить Eclipse IDE (см. лаб. работу 1). В Eclipse IDE создать рабочую папку и открыть новый Java проект. Далее, открыть новый файл java, в который будем записывать программный код агента.

Перед тем как создавать агента нам потребуется установить платформу JADE. Скачать архив с платформой можно с официального сайта JADE по ссылке: http://jade.tilab.com/. Но необходимо будет зарегистрироваться и получить подтверждение регистрации по электронной почте. Скачайте последнюю версию платформы. Распакуйте архив и откройте его. Архив содержит 3 папки:

- JADE-bin-4.X – папка с исполняемыми файлами;
- JADE-src-4.X – папка с исходными файлами;
- JADE-example-4.X – папка с примерами.

Создайте отдельную папку, например, с названием Jade, и скопируйте содержимое всех трех папок из архива в новую папку (для упрощения адреса лучше всего новую папку создать в корне диска).

В результате мы получаем готовую платформу. Проверьте расположение таких файлов как:

- build.xml – находится в корне папки Jade (созданной нами);
- jade.jar – находится в папке lib/;
- папки исходных файлов examples/ и jade/ - находятся в src/.

Перед запуском платформы и дальнейшей работы с ней хотелось бы обозначить следующие моменты в архитектуре системы. Основными элементами мультиагентной системы являются *Агенты*.

Мультиагентная платформа состоит из контейнеров, в которых «живут» Агенты. Контейнер может не содержать ни одного агента или содержать сколько угодно агентов. Каждый контейнер обладает своим сетевым адресом и именем. Есть особый контейнер – главный контейнер – без него платформа не работает. Главный контейнер имеет несколько особенностей. Перечислим их:

- он должен быть создан первым;
- он должен включать в себя двух специальных агентов:

o AMS (agent management system) – агент, который управляет остальными агентами; он способен создавать и останавливать агентов;

o DF (directory facilitation) – по сути представляет из себя желтые страницы, в котором записываются адреса, имена и возможности агентов в системе.

3.5.2.2.Выполнение работы

Для запуска платформы необходимо запустить Eclipse IDE, создать новую рабочую среду и в ней открыть новый Java проект (рис. 3.6).

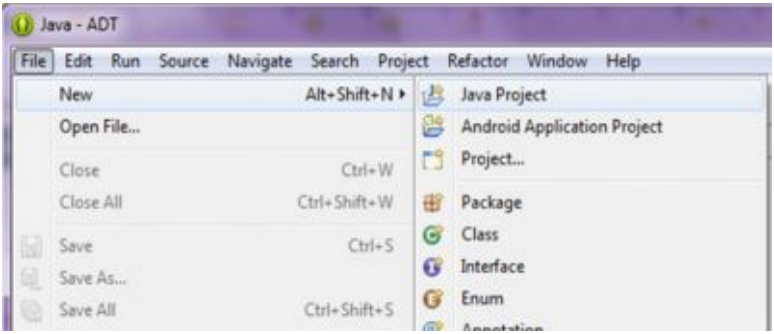


Рис. 3.6. Новый Java проект в Eclipse IDE.

Далее переходим в окно создания проекта, задаем имя и нажимаем *Next*. В следующем окне необходимо подключить к нашему проекту библиотеки Jade (рис. 3.7)

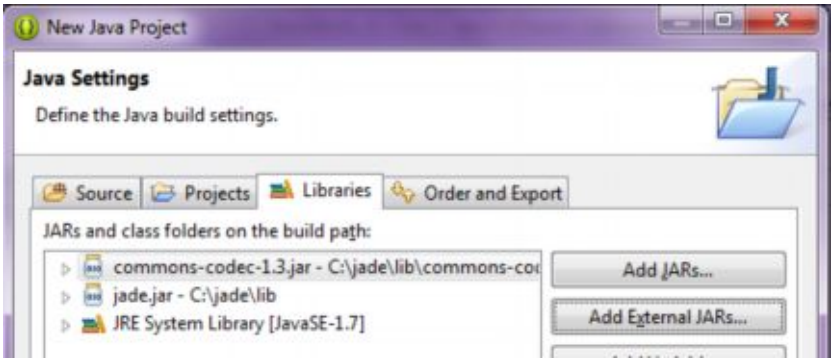


Рис. 3.7. Библиотеки Jade в проекте.

Для подключения библиотек необходимо нажать *Add External JARs* и указать путь к библиотекам. Они находятся в папке \lib, куда были скопированы все файлы архивов платформы Jade. После этого, нажимаем *Finish*.

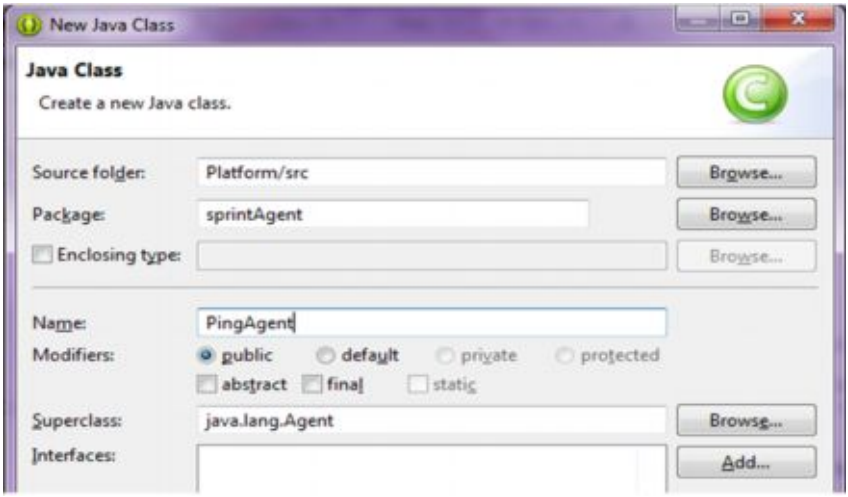


Рис. 3.8. Новый класс.

Теперь к проекту подключены библиотеки JADE. Создаем нового Агента из примера «PingAgent». Для этого необходимо создать новый класс в папке \src (рис. 3.8).
Теперь вводим информацию, создаем новый класс в папку **Project/src** в пакете **sprintintel** с именем **PingAgent**. Также на этом этапе необходимо записать в поле «Superclass» следующую информацию: **java.lang.Agent** и нажать *Finish*.

Сейчас у нас есть новый Агент и подключены необходимые библиотеки. Необходимо задать программный код Агента. Для этого открываем PingAgent.java из папки *examples\PingAgent*, которая находится в директории, в которую распакованы архивы Jade. Копируем все содержимое примера в нашего агента. Посмотрим на код нашего агента, в нем есть ошибки, которые необходимо исправить (рис. 3.9).

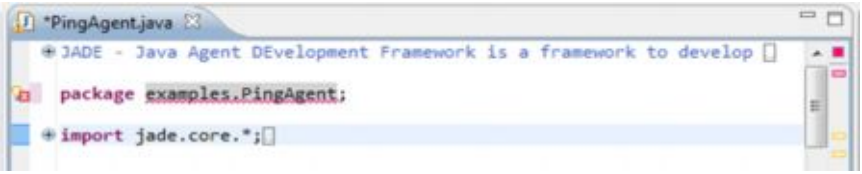


Рис. 3.9. Ошибка в коде Агента.

Эта ошибка вполне понятна. Необходимо исправить имя пакета, который мы используем. В нашем случае – это **sprintintel**. Далее необходимо обязательно сохранить изменения в файле Агента.

Для того чтобы запустить Агента и мультиагентную платформу переходим меню Run Configuration. В нем добавляем новую конфигурацию запуска Java-приложения. На рис. 3.10. и 3.11 показаны параметры главного класса (MainClass) – jade.Boot и видно название настоящего проекта. В поле *Program Argument* вкладки *Arguments* вводим значении «–gui». Таким образом, мы определили, что нам необходимо запустить платформу и главный контейнер, который уже содержит дополнительный класс нашего агента PingAgent.

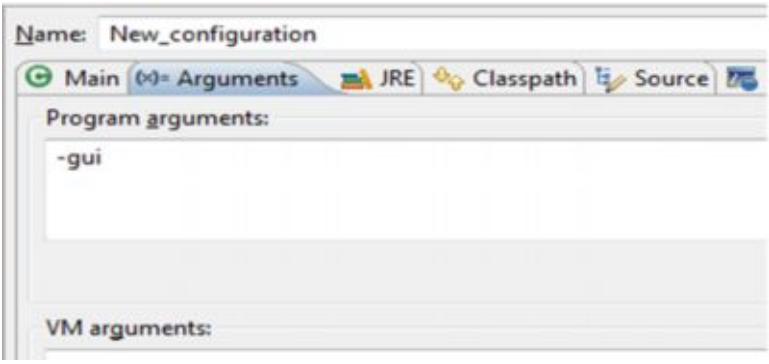


Рис. 3.10. Конфигурация запуска, вкладка Arguments.

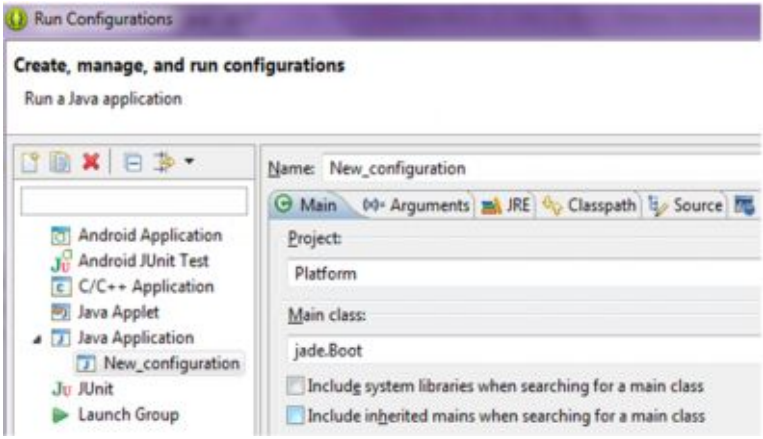


Рис. 3.11. Конфигурация запуска. Вкладка Main.

Нажимаем на кнопку Run и запускаем нашу платформу. Платформа запущена, и мы видим пользовательский интерфейс, в котором запущена платформа и два базовых агента AMS и DF. Чтобы добавить в главный контейнер нашего Агента «Ping», необходимо войти во вкладку *Action* и добавить класс нашего Агента (рис. 3.13).

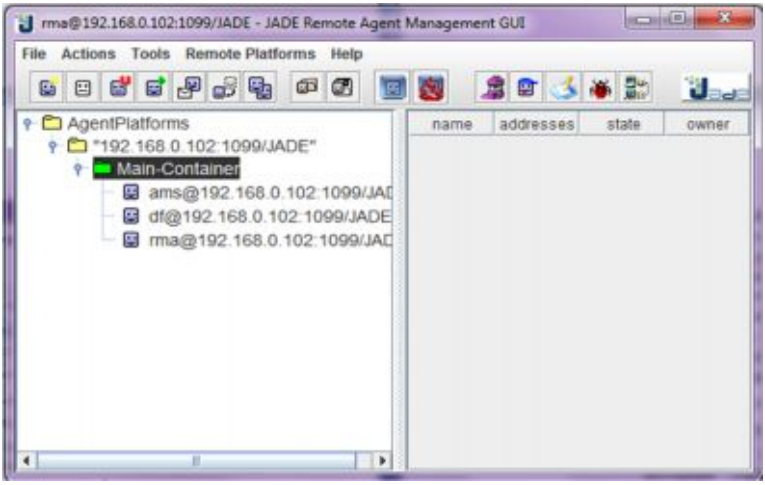


Рис. 3.12. Пользовательский интерфейс платформы Jade.

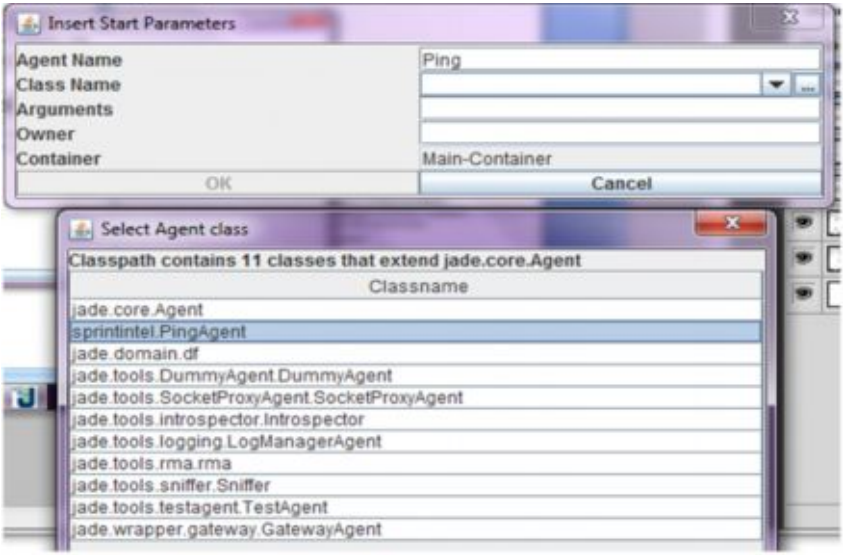


Рис. 3.13. Class Агента Ping Agent в пакете sprintintel.

Наш агент теперь находится в главном контейнере. В чем же суть Агента в примере PingAgent? Основная задача этого агента – это

отправлять ответ в виде слова «Ping» при запросе в виде слова «Pong». В коде эта часть выглядит так:

```
public void action() {
  ACLMessage  msg = myAgent.receive();
  if(msg != null){
    ACLMessage reply = msg.createReply();
    if(msg.getPerformative()== ACLMessage.REQUEST){
      String content = msg.getContent();
      if ((content != null) &&
           (content.indexOf("ping") != -1)){
        myLogger.log(Logger.INFO, "Agent
            "+getLocalName()+" - Received PING Request
               from "+msg.getSender().getLocalName());
        reply.setPerformative(ACLMessage.INFORM);
        reply.setContent("pong");
      }
    }
  }
}
```

При запуске агента в главном контейнере он обращается в «желтые страницы» – агент DF, который записывает у себя адрес этого агента. Все запросы осуществляются через DF агента. Таким образом, в систему могут входить и выходить новые агенты и быть «на связи».

Теперь проверим работу нашего Агента и отправим ему запрос с помощью DummyAgent – готовый агент с пользовательским интерфейсом, посредством которого возможно отправлять сообщения на другие агенты и получать от них ответы. Для запуска DummyAgent необходимо нажать кнопку StartDummyAgent в ряду кнопок быстрого доступа пользовательского интерфейса платформы. В самом Агенте необходимо составить запрос в правильной форме, как это показано на рис. 3.14.

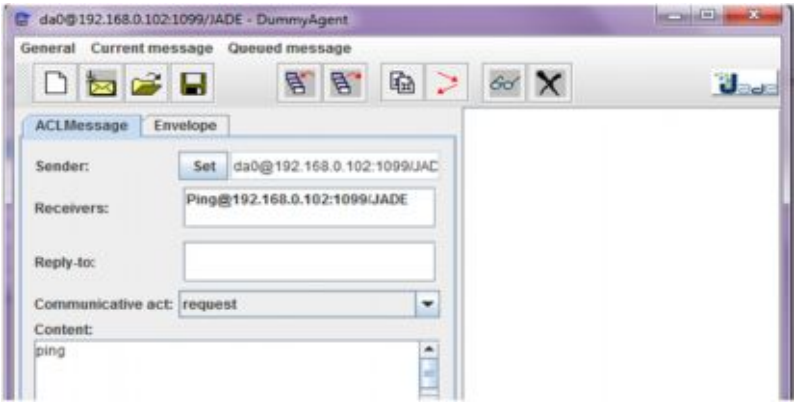


Рис. 3.14. DummyAgent, запрос на Ping агента.

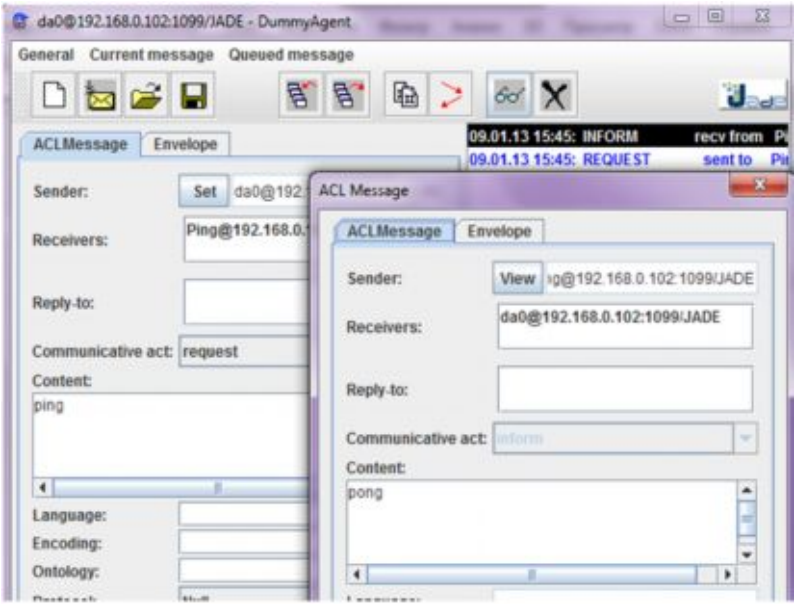


Рис. 3.15. Ответ на запрос DummyAgent.

После правильного заполнения запроса необходимо нажать на кнопку: отправить сообщение. DummyAgent позволяет отправляет запрос с одного точного адреса агента на точный адрес другого агента даже если они находится в различных контейнерах. Но эти агенты должны быть прописаны в одной мультиагентной системе

(т.е. известны DF агентам). Справа, нажав на значок очки, в белом поле пользовательского интерфейса DummyAgent, можно увидеть, пришел ли ответ и какого он содержания (рис. 3.15.).

Для того, чтобы проще было отслеживать отправку сообщений, можно запустить еще одного агента Sniffer, в котором отражаются стрелочками отправленные сообщения между агентами (Рис. 3.16).

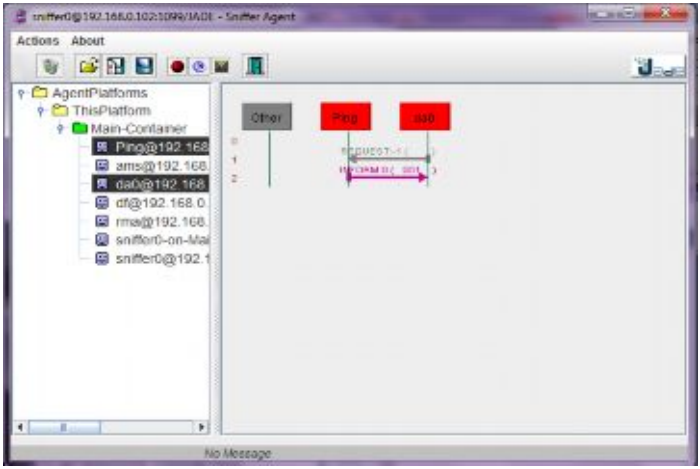


Рис. 3.16. Агент Sniffer.

3.5.3. Задания для самостоятельной работы

1. Создать еще одного агента Ping1 и отправить запрос двум агентам одновременно
2. Изменить текст сообщения запроса и ответа.
3. Создать Container1 (не главный) и создать в нем агента.
4. Запустить агентов в примере папки src/examples/bookTrading.

3.6. Выводы

В этой главе были описаны основные принципы разработки мультиагентных систем и разобран пример работы с платформой JADE.

3.7. Контрольные вопросы

1) Верно ли, что МАС обычно состоит из следующих основных частей: множество организационных единиц, среда, множество отношений между агентами, множество задач?

Варианты ответов:

1. Нет, отсутствует множество действий агентов
2. Нет, отсутствует множество намерений
3. Да, верно

2) Что такое JADE?

Варианты ответов:

1. Java Autonomous Development Environment
2. Java Agent Development Environment
3. Java Agent Development Framework

3) Какие стандарты для интеллектуальных агентов поддерживает JADE?

Варианты ответов:

1. AgentService
2. FIPA
3. NetLogo

4) К проектам на каком языке подключается JADE?

Варианты ответов:

1. Java
2. C
3. Delphi

5) Какого типа агенты поддерживаются в JADE?

Варианты ответов:

1. Простые, только регистрирующие
2. Простые, а также агенты с одним дополнительным свойством
3. От простых, только реагирующих, до сложных — ментальных

6) Что такое BDI?

Варианты ответов:

1. Стандарт для интеллектуальных агентов

2. Модель логики агентов, основанная на убеждениях-желаниях-намерениях
3. Платформа для разработки МАС, основанная на убеждениях-желаниях-намерениях

7) Для чего используется язык KQML?

Варианты ответов:

1. Для общения агентов
2. Для описания агентов в среде MadKIT
3. Для описания агентов в среде AgentBuilder

8) Что такое MASON?

Варианты ответов:

1. Платформа для разработки мультиагентных систем
2. Java библиотека для моделирования мультиагентных систем
3. Язык для описания агентов

9) Для чего чаще всего используются МАС в промышленности?

Варианты ответов:

1. Разработка новых подходов к добыванию сырья
2. Управление жестко организованными системами
3. Автоматизация управления ресурсами

10) Для чего может эффективно использоваться МАС в компании, занимающейся перевозками пассажиров (такси)?

Варианты ответов:

1. Для составления маршрута движения автомобиля
2. Для приема заказов на перевозку пассажиров
3. Для планирования и распределения заказов в режиме реального времени

11) Какой блок в схеме взаимодействия модулей инструментальной системы отвечает за построение прогноза заявок?

Варианты ответов:

1. Адаптивный планировщик
2. Конструктор сцены
3. Детектор образов

12) Какой блок в схеме взаимодействия модулей инструментальной системы обрабатывает поток входящих событий?

Варианты ответов:

1. Адаптивный планировщик
2. Редактор онтологии
3. Конструктор сцены

13) Что делает моделирующая система в схеме взаимодействия модулей инструментальной системы?

Варианты ответов:

1. Строит план распределения заказов
2. Осуществляет моделирование ситуации по принципу "Что если?"
3. Строит модель системы

14) Есть ли в схеме взаимодействия модулей инструментальной системы отдельный модуль, отвечающий за улучшение конфигурации сети?

Варианты ответов:

1. Да, эволюционный дизайн
2. Нет
3. Нет, за это отвечает конструктор сцены

Список литературы

1. *Тарасов В.Б.* Агенты, многоагентные системы, виртуальные сообщества: стратегическое направление в информатике и искусственном интеллекте // Новости искусственного интеллекта. 1998. № 2. С. 5-63.

2. *Shoham Y., Leyton-Brown K.* Multiagent Systems: Algorithmic, Game-Theoretic and Logical Foundations — London: Cambridge University Press. 2009. 513 p.

3. *Граничина Н.О.* Мультиагентная система для распределения заказов // Управление большими системами. Спец. выпуск 30.1 "Сетевые модели в управлении". 2010. С. 549-566.

4. *Виттих В.А., Скобелев П.О.* Метод сопряженных взаимодействий для управления распределением ресурсов в реальном

масштабе времени // Автометрия. 2009. Т. 45. № 2. С. 84-86.
5. *Каляев И.А., Гайдук А.Р., Капустян С.Г.* Распределенные системы планирования действий коллективов роботов. — М.: Янус-К. 2002. 292 с.
6. *Амелин К.С, Антал Е.И., Васильев В.И., Граничина И.О.* Адаптивное управление автономной группой беспилотных летательных аппаратов // Стохастическая оптимизация в информатике. Вып. 5. 2009. С. 157-166.
7. *Antal C, Granichin O., Levi S.* Adaptive autonomous soaring of multiple UAVs using SPSA // Proc. of IEEE Conf. on Decision and Control (CDC-49). 2010. Atlanta. USA. P. 3656-3661.
8. *Амелин К. С, Граничин О.И.* Мультиагентное сетевое управление группой легких БПЛА // Нейрокомпьютеры: разработка, применение. 2011. № 6. С. 64-72.
9. Сайт JADE: http://jade.tilab.com/doc/index.html.
10. *Амелин К.С.* Технология программирования легкого БПЛА для мобильной группы // Стохастическая оптимизация в информатике, выпуск 7, 2011, С.93-115.
11. *Амелина Н.О.* Достижение консенсуса автономной группой беспилотных самолетов // Стохастическая оптимизация в информатике, выпуск 6(1), 2010. С. 127–132.
12. Статьи по разработки адаптивного управления для группы автономных мобильных роботов https://sites.google.com/site smartflyllc/file-cabinet.
13. *Кияев В.И., Герасимов Р.В.* Интеллектуальный CRM на базе мультиагентного подхода// В сб. «Стохастическая оптимизация в информатике». Изд-во С.Петербург. ун-та, т. 8, вып. 1, 2012, с. 50-94.

4. Проект «Мультиагентная система для БПЛА»

Описание трехуровневой системы управления БПЛА для реализации мультиагентного взаимодействия.

4.1. Введение

В настоящее время миниатюризация электронных компонент и технологичность их изготовления позволяет применять небольшие и доступныен беспилотные летательные аппараты (БПЛА) для выполнения задач исследования территории. Несмотря на небольшие размеры «игрушечные» самолеты сейчас способны развивать скорость свыше четырехсот км/ч, при установки на них небольшого реактивного двигателя. В нашем проекте «Мультиагентная система для беспилотного летательного аппарата (БПЛА)», представленном в сентябре 2012 года на «Петербургской осенней выставке ЭКСПО-2012», используется легкий планер PAPRIKA и небольшой квадрокоптер (вертолет), который представляет из себя четырехмоторную раму в виде креста (рис. 4.1).

Технические характеристики планера:

- размах крыльев 2 м, вес 2 кг
- полезная нагрузка 500 г
- скорость 40-60 км/ч
- дальность 200 км

и квадрокоптера:

- длинна лучей 450 мм
- полезная нагрузка до 1,2 кг
- скорость до 20 км/ч
- дальность до 1 км

Рис. 4.1. Внешний вид планера и квадрокоптера (на рисунке слева – один из разработчиков проекта К.С.Амелин, на заднем плане – научный руководитель проекта, профессор О.Н.Граничин)

Комплект бортового оборудования для управления и связи состоит из:

- исполнительных механизмов, которые в зависимости от источника способны работать, как с цифровым, так и с аналоговым управляющим сигналом;
- комплекта мотоустновки, в который входит: аккумулятор (LiPo – хорошая токоотдача, но потеря емкости при отречательных температурах или LiFePO4 – низкая токоотдача, но работа при низких температурах), контроллер оборотов и безколлекторный электрический двигатель;
- автопилота Ardupilot (Arducopter) построенного на основе процессора ATmega 2560 (16 MHz) и дополнительного процессора ATMega328 для обработки управляющего сигнала;
- комплекта приемника радиосигнала;

- для навигации используется стандартный набор инерциальной системы (гироскопы, магнитометры и акселерометры) и современный ГЛОНАСС/GPS приемник;
- передача телеметрии построена на модемах XBee 2.4 GHz;
- различное дополнительное оборудование;
- смартфон Intel с ОС Android построен на основе процессора Intel Atom X86.

За счёт небольшого веса БПЛА взлёт осуществляется с рук человека (в стандарте предполагается взлёт с катапульты). Посадка – либо за счёт встроенного парашюта, либо за счёт «перехвата» на ручное управление. Наличие «на борту» полнофункционального микрокомпьютера позволяет использовать БПЛА не просто в режиме «удаленной видеокамеры», которая передает видеопотоки в зоне прямой радиосвязи, а как интеллектуального мобильного агента с функциями автопилота. Такой агент способен существенно переработать исходную информацию, подготовить ее для обработки и самостоятельно автономно передать ее суперкомпьютеру по разным каналам связи – в том числе и через Интернет. Реальное ускорение в обработке данных позволит перевести в практическую плоскость и организацию автоматической обратной связи от суперкомпьютера к БПЛА с указанием возможных корректировок планов полета и наблюдений. Бортовые микрокомпьютеры со связью позволяют организовать и автономное взаимодействие внутри группы БПЛА для возможности перераспределения или уточнения заданий, обмена данными и т. п.

Например, при использовании группы БПЛА для геологоразведки на определенной территории работа системы организуется следующим образом:

1. Выбирается тип задачи (в выбранном примере – геологоразведка).
2. В зависимости от площади исследуемой территории и количества БПЛА в группе, их характеристик, территория разделяется на участки, и формируются отдельные задачи для каждого члена группы.
3. В микрокомпьютер каждого БПЛА группы записывается глобальная задача (параметры исследуемой территории и т. п.) и отдельная задача этого самолета-агента.

4. Каждый агент приступает к выполнению поставленной ему задачи.
5. Когда в зону Wi-Fi видимости одного БПЛА из группы попадает другой, при «общении» происходит передача между агентами накопленной информации и при необходимости взаимное уточнение отдельных задач. Таким образом, по ходу выполнения частной задачи, все агенты накапливают информацию о ходе решении общей задачи группы, а также локально принимают решения о корректировке своих частных задач для более эффективного выполнения общей. Например, группа БПЛА летит по определённому маршруту, все БПЛА летят на разной высоте, при обмене информацией между членами группы выясняется наилучшая высота полёта по энергозатратам, по ходу выполнения задания вся группа оперативно перестраивается на эту высоту.
6. Базовые наземные станции, обеспечивая связь с центром обработки данных (ЦОД), принимают/передают информацию от БПЛА, находящихся в их зоне видимости или поддерживающих связь через Интернет. Так как в процессе общения между БПЛА информация о выполнении общей задачи накапливается во всех микрокомпьютерах группы, то данные даже от тех самолетов, которые редко выходят на связь, все равно попадает в ЦОД.
7. Полученная в ЦОДе информация обрабатывается и визуализируется для заказчика (выдается карта с нанесенными исследуемыми характеристиками).
8. Наличие обратной связи с мобильными агентами (БПЛА) позволяет оперативно формировать из ЦОДа инструкции по корректировке их заданий.

В мире ИТ уже во многих областях побеждает схема, в которой нет четкого выделения трех изолированных стадий: подготовка данных, счет, анализ результатов. Этому способствовало всеобщее проникновение персональных компьютеров, смартфонов, мобильных телефонов, которые кардинально изменили представления о месте ЭВМ в обычной жизни. Компьютеры уже интегрированы во многие сферы человеческой деятельности, разработчики встраивают миниатюрные вычислительные

устройства и специализированные программы во внутрь многих процессов. В массовом сознании о роли ЭВМ в практической жизни уже произошел сдвиг от супервычислителя к мобильности, понимаемой в очень широком смысле.

В действительности мобильность и высокопроизводительные вычисления, по нашему мнению, будут всё более тесно связаны. Посмотрим на пример такой традиционной области для ИТ, как «кодирование/декодирование информации». Полвека развития со времен пионерских работ Котельникова-Найквиста-Шеннона успели убедить несколько поколений в незыблемости и правильности постулатов о минимальном числе отсчетов необходимых для точного восстановления сигнала (функции). Но развитие технологий в 90-е годы прошлого века позволило массово перейти от обработки одномерных сигналов к изображениям, а сейчас на повестке дня стоит решение задач трехмерного телевидения. Однако реальные задачи обработки таких сигналов для традиционных средств кодирования/декодирования уже становятся «не по плечу» даже суперкомпьютерам!

Что же происходит сейчас у нас на глазах с развитием и изменением «парадигмы кодирования/декодирования»? Вместо классической появляется новая парадигма «Compressive Sensing», для которой еще даже не придумано русское название (статья О. Граничина и Д. Павленко в журнале «Автоматика и телемеханика» [1]). В чем ее суть? Пользователям в реальном времени обычно не нужна вся информация о процессах в каждый момент времени, а важно только изменение принципиальных компонент. Например, при передаче ведеопотока с трехмерной сцены достаточно однократно передать общий вид и детали, выделить характерных персонажей, а потом только отслеживать их изменения. Оказывается, имея интеллектуальные сенсоры (использование принципа мобильности сбора информации), целесообразно до момента кодировки и передачи данных вместо определения всех значений трехмерной сцены (обычно это огромное число) сначала произвести несколько «сверток» всей сцены с некоторыми рандомизированными матрицами, а потом закодировать и передать для обработки суперкомпьютеру только эти несколько измерений. А это означает, что уже сейчас можно

говорить о том, что мобильность и использование интеллектуальных агентов при сборе и предварительной обработке данных может существенно сократить время подготовки данных для суперкомпьютерных вычислений. Если глубже взглянуть на новую парадигму кодирования/декодирования, то становится ясным, что при ней существенно меняется не только процесс подготовки данных, но и сам процесс обработки. Таким образом мы говорим уже взаимном дополнении и развитии таких важнейших технологий, как мобильность и супервычисления..

Исходя из вышесказанного перспективную модель вычисления можно представить следующим образом: мобильные устройства, которые обладают различными датчиками и сенсорами (например, видео- или фото- камерой, gps, датчиками температуры и давления и т.п.) и способны «общаться» друг с другом, являются сборщиками информации, они передают информацию на суперкомпьютер где и происходят конечные вычисления. Мобильные устройства, собирая информацию, одновременно могут фильтровать её, т.е. отмечать особенности и разделять данные различные по тематике (например, отделить файлы с координатами и файлы с фотографиями).

Эта модель реализуется на примере проекта «Мультиагентная система для группы лёгких беспилотных летательных аппаратов (БПЛА)» [2]. Трудно переоценить актуальность такой задачи для России с её огромной территорией, сельскохозяйственными угодьями, лесами, множеством газо- и нефтепроводов, ЛЭП и «пожарами». При создании такой системы как раз и проявляются все обсуждавшиеся выше проблемы. Многие из задач – разведка полезных ископаемых, поиск людей, выяснение причин техногенных катастроф, плановое патрулирование территорий и т.п. требуют для получения практических результатов применения супервычислений с огромным количеством исходных данных, которые должны поступать на суперкомпьютер в режиме реального времени. Если используются традиционные БПЛА с оператором и приемом/передачей данных по радиоканалу, то реальные автоматизированные системы «захлебываются» от потоков данных, которые надо еще сохранять и передавать компьютеру для обработки. Об обратном потоке управления БПЛА обычно даже

речь и не идет, так как результаты не удается получить в реальном времени.

Основная идея проекта заключается в создании сети БПЛА. Такая сеть состоит из группы БПЛА-агентов, которые способны «общаться» между собой и обмениваться данными с базовой станцией или с сетью базовых станций. Базовая станция или сеть таких станций «собирают» и фильтруют информацию и отправляют конечный результат суперкомпьютеру.

В этой главе будет рассмотрена терхуровневая система управления БПЛА, на основе которой строится мультиагентная система для управления легкими БПЛА.

4.2. Мультиагентный подход и трёхуровневая система управления БПЛА

Комплекс БПЛА, как правило, состоит из одного БПЛА и базовой станции под управлением человека. Человек с базовой станцией определяет задачу для БПЛА, последний, в свою очередь выполняет её и передаёт информацию на базовую станцию, при этом человек способен отслеживать ход выполнения задания. Иногда используют несколько комплексов, при этом каждый БПЛА привязан к своей базовой станции. Обмен информацией между комплексами происходит только на земле, между базовыми станциями [6]. Такую работу комплекса можно представить, как двухуровневую систему управления. Верхний уровень управления осуществляется базовой станцией. Здесь определяется глобальная задача для БПЛА, задаётся высота полёта, скорость полёта, маршрут, тачки сбора информации и т.д. Нижний уровень управления осуществляется автопилотом БПЛА, который действует по записанной на него программе с базовой станцией. Для выполнения задачи автопилот управляет исполнительными механизмам, собирает информацию с датчиков и отправляет данные на базовую станцию.

Описанная выше систем управления комплексом и принцип применения нескольких комплексов БПЛА для группового

выполнения задачи характеризуются отсутствием автономной постановки новых задач, позволяющей группе оперативно принимать эффективные решения по изменению сценария выполнения поставленной задачи. Типичными примерами событий, вызывающих необходимость в постановке новых задач, являются: появление нового выгодной информации, для более эффективного выполнения задачи; выход из строя части имеющихся ресурсов; а также изменение критериев принятия решений. Чем выше неопределенность, чем более распределенный характер имеют процессы принятия решения и чем чаще случаются незапланированные события, тем ниже эффективность существующих систем, не способных самостоятельно принимать решения и автоматически перестраиваться под изменения в среде. Кроме того, любая модификация схем принятия решений в традиционных системах представляет собой весьма сложный и трудоемкий процесс и требует высокой квалификации исполнителей, что делает разработку и эксплуатацию рассматриваемых систем крайне дорогостоящими.

Для решения подобных проблем применяются мультиагентные технологии. В основе этих технологий лежит понятие “агента”, программного объекта, способного воспринимать ситуацию, принимать решения и взаимодействовать с себе подобными. Характерными особенностями интеллектуальных агентов являются:

- коллегиальность, т.е. способность к коллективному целенаправленному поведению в интересах решения общей задачи;
- автономность, т. е. способность самостоятельно решать локальные задачи;
- активность, т. е. способность к активным действиям ради достижения общих и локальных целей;
- информационная и двигательная мобильность, т. е. способность активно перемещаться и целенаправленно искать и находить информацию, энергию и объекты, необходимые для кооперативного решения общей задачи;
- адаптивность, т. е. способность автоматически приспосабливаться к неопределчнным условиям в динамической среде.

Эти возможности кардинально отличают мультиагентные системы (МАС) от существующих “жестко” организованных систем управления группы автономных БПЛА [7].

Мы применяем мультиагентный подход для управления группой БПЛА. Примером подобного взаимодействия может служить система описанная в статье [8]. Авторы статьи описывают взаимодействие главного компьютера (самолет управляемый человеком) и группы БПЛА, которые выполняют задачи поставленные первым. Наш же подход основан на “общении” моделей друг с другом, при этом образуется новый источник информации для корректировки полета автопилотом. Желательно выполнение двух основных требований

Желательно выполнение двух основных требований для разработки мультиагентной системы для группы БПЛА. Во-первых, на каждой модели долёжен быть небольшой, но мощный микрокомпьютер для работы, автономного изменения задания для автопилота, и общениея в мультиагентной системе. Во-вторых, необходимо организовать устойчивую связь между агентами группы.

Отметим полезные свойства, которыми обладает группа взаимодействующих моделей по сравнению с использованием одного БПЛА:

- коллективный облет вырабатывает большую картину мира;
- взаимодействие между собой помогает оптимизировать маршрут полета, основываясь на уже имеющихся данных с другого БПЛА;
- большее множество стратегий;
- возможность создания объемной картины мира;
- более эффективное решение задач (экология, метеорология, оптимизация полёта группы);
- больше гарантий выполнения задачи;
- выигрыш во времени (главное условие для задач поиска потерявшихся людей);
- возможность одновременного обследования территории;
- возможность корректировки плана и выбора оптимального маршрута, основываясь на имеющихся данных соседнего UAV;

- возможность постановке разных задач для разных участников группы UAV.

Для применения мультиагентного подхода к группе БПЛА в двухуровневую схему управления комплексом БПЛА мы добавляем промежуточный средний уровень, который осуществляется за счёт дополнительного микрокомпьютера установленного в БПЛА [2].

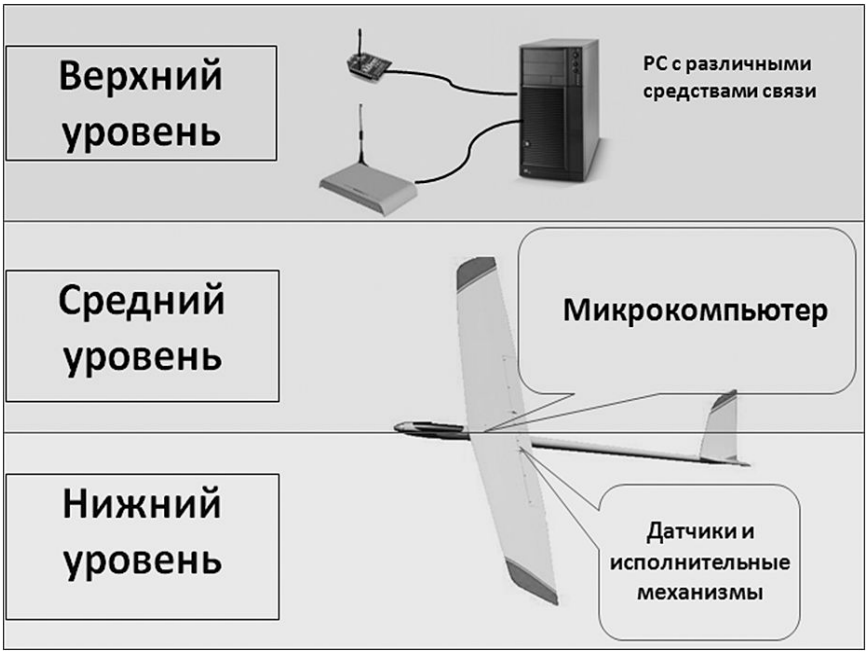


Рис. 4.2. Три уровня управления БПЛА-агента.

Таким образом, мы получаем новую трёхуровневую систему управления БПЛА-агента (рис 4.2), в которой, на верхнем уровне происходит формирование глобального задания уже для группы БПЛА, начальные условия, корректировка или изменения глобальной задачи, а также приём и обработка полученной от группы информации. На верхнем уровне находится персональный компьютер, к которому подсоединены различные модули для связи (модемы, радиоприемники и др.). Средний уровень управление осуществляется за счёт внедрения в БПЛА микрокомпьютера,

который способен обмениваться информацией с другими членами группы и базовой станцией. На основе этой информации микрокомпьютер может изменить свою начальную задачу для ускорения достижения цели глобальной задачи, если это потребуется. Таким образом, мы получаем группу БПЛА-агентов, которая способна адаптироваться к изменениям происходящим во внешней среде. На нижнем уровне находятся управление исполнительными механизмами, датчиками и дополнительным оборудованием.

4.3. БПЛА-агент для автономной группы

Выше была описана система управления группой БПЛА. Основное отличие группы от привычного комплекса БПЛА – это автономное общение БПЛА-агентов между собой. Для того, чтобы реализовать мультиагентную систему для управления группой БПЛА, необходимо пересмотреть аппаратное оснащение одиночного БПЛА.

Рассмотрим приведенную в предыдущей главе трехуровневую систему с точки зрения аппаратной реализации [9].

На верхнем уровне управления нами применяется компьютер – базовая станция (ноутбук, нетбук или стационарный ПК), оснащённый различными средствами связи (радиомодемом, Wi-Fi модемом, Интернет модемом) (рис. 4.3). Основными задачами базовой станции являются:

- определение глобальной миссии группы БПЛА-агентов (параметры территории исследования, задание способов исследования, высота полёта и т.д.);
- разбиение глобальной задачи на части и формирование начальной индивидуальной задачи для каждого БПЛА-агента группы в зависимости от количества БПЛА-агентов и поставленной перед группой задачей;
- обмен информацией с БПЛА-агентами, находящимися в зоне связи;
- сбор и обработка информации от БПЛА-агентов группы;

- формирование новой глобальной миссии дял группы в зависимости от поступающей на базовую станцию новой информации.

Рис. 4.3. Верхний уровень управления БПЛА-агента.

На среднем уровне управления БПЛА-агента находится бортовой микрокомпьютер – головное устройство системы управления БПЛА. Его основная цель – выполнить поставленную ему задачу с наименьшими затратами по времени и ресурсам [7]. Для этого он выполняет пять основных функций:

- генерация обновлений к программе полета для автопилота;
- обработка данных навигационного оборудования и телеметрии;
- работа с дополнительным оборудованием;
- общение с микрокомпьютерами других БПЛА (если работа ведется в группе);
- отправка данных на базовую станцию и получение от нее новых заданий.

С базовой станции микрокомпьютер получает общие задачи (начальное состояние, конечные точки, задача для группы и т.п.). При групповом полете общая задача разбивается на частные для каждого БПЛА. В процессе выполнения задачи микрокомпьютер осуществляет общение с другими членами группы, находящимися в радиусе действия радиосвязи. Взаимодействие в группе позволяет группе БПЛА эффективнее выполнить общую задачу, а также избежать столкновений.

На основе полученных данных от базовой станции, с навигационного оборудования и при общении с другими БПЛА, микрокомпьютер может сгенерировать новую программу для

автопилота, если старая не поддерживает нужных требований для выполнения общей задачи.

При пролете БПЛА в зоне связи с базовой станцией микрокомпьютер отправляет новые накопленные им данные и получает новые задания. При этом накопление данных происходит не только за счет собственных сенсоров и датчиков, но и при связи с другими БПЛА. Как говорилось выше, в микрокомпьютер закладывается отдельная для него задача и общая задача группы. В процессе общения микрокомпьютер накапливает информацию о выполнении общей задачи. Данные о выполнении своей частной задачи он получает сам, используя бортовые датчики.

Связь с базовой станцией осуществляется за счет отдельного радиоканала или через GPRS по GSM модему. GSM модем легко интегрируется с микрокомпьютером, но для передачи данных пакеты необходимо сжимать.

Связь между микрокомпьютерами разных БПЛА осуществляется за счет встроенного радиоприемника с частотой 2,4 GHz и протоколом общения 802.11 n (Wi-Fi), в котором применяется технология, связывающая два ближайших канала в один. Таким образом, микрокомпьютеры в БПЛА смогут одновременно принимать и отправлять информацию друг другу. Связь с базовой станцией осуществляется за счет отдельного радиоканала или через GPRS по GSM модему [9].

За счёт небольшого веса БПЛА взлёт осуществляется с рук человека или с катапульты. Посадка – либо за счёт встроенного парашюта, либо за счёт «перехвата» на ручное управление.

Нижний уровень управление осуществяется автопилотом БПЛА-агента (рис. 4.4). Автопилот – устройство с микроконтроллером с системой реального времени. Основная задача автопилота состоит в управлении исполнительными механизмами (сервоприводами, мотоустановкой, дополнительным оборудование), основываясь на записанной ему программе полёта и информации с датчиков (инерциальной системы, инфракрасных датчиков, датчиков давления и скорости и т. п.).

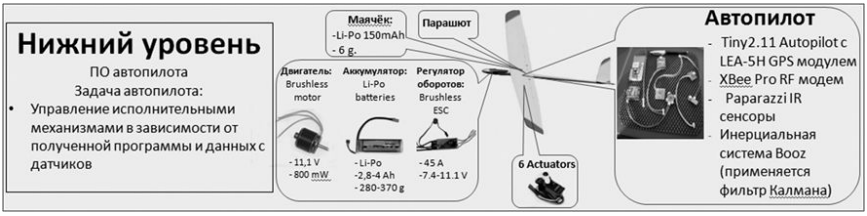


Рис. 4.4. Нижний уровень управления БПЛА-агентом.

Таким образом, аппаратная реализация трехуровневой системы, которая нужна для создание мультиагентной сети управления группой БПЛА, возможна.

4.4. Алгоритмы групповой работы сети БПЛА

Во введении были перечислены преимущества, которыми обладает группа взаимодействующих БПЛА-агентов по сравнению с использованием одного БПЛА. В разделе 3 описана аппаратная возможность построения группы БПЛА-агентов. В этом разделе мы рассмотрим два основных типа алгоритмов управления полётом БПЛА: алгоритм мониторинга местности на примере исследование экологической ситуации и алгоритм оптимизации полёта БПЛА на примере использования термических потоков для увеличения дальности полёта [6].

При использовании группы БПЛА для мониторинга местности (рассмотрим задачу исследования экологической обстановки в акватории залива, например, разлив нефти) работа системы организуется следующим образом:

- Выбирается тип задачи (в выбранном примере – поиск нефтяных пятен и источника их образования).
- В зависимости от площади исследуемой территории, количества БПЛА в группе и их характеристек территория разделяется на участки, и формируются отдельные задачи для каждого члена группы (для выбранного примера задача будет стоять таким

образом: поиск аномалии интенсивности цвета поверхности акватории).

- В микрокомпьютер каждого БПЛА группы записывается глобальная задача (параметры исследуемой территории и т. п.) и отдельная задача этого БПЛА-агента.
- Каждый агент приступает к выполнению поставленной ему задачи.

Когда в зону Wi-Fi видимости одного БПЛА из группы попадает другой при «общении» происходит передача между агентами накопленной информации и, при необходимости, взаимное уточнение отдельных задач. Таким образом, по ходу выполнения частной задачи, все агенты накапливают информацию о ходе решении общей задачи группы, а также локально принимают решения о корректировке своих частных задач для более эффективного выполнения общей. (Например, для выбранного нами примера, группа БПЛА исследует территорию на предмет нефтяных пятен, все БПЛА исследуют разные квадранты. При обмене информацией между членами группы выясняется наличие пятен в одном из квадрантов. По ходу выполнения задания вся группа перестраивает задачу, которая сводится к поиску источника разлива нефти.)

Базовые наземные станции, обеспечивая связь с центром обработки данных (ЦОД), принимают/передают информацию от БПЛА, находящихся в их зоне видимости или поддерживающих связь через Интернет. Т.к. в процессе общения между БПЛА информация о выполнении общей задачи накапливается во всех микрокомпьютерах группы, то данные даже от тех самолетов, которые редко выходят на связь, все равно попадает в ЦОД.

Полученная в ЦОДе информация обрабатывается и визуализируется для заказчика (выдается карта с нанесенными исследуемыми характеристиками).

Наличие обратной связи с мобильными агентами (БПЛА) позволяет оперативно формировать из ЦОДа инструкции по корректировке их заданий.

Такой алгоритм действия группы подходит к любым задачам визуального мониторинга территории. Основные отличия будут

состоять в типе искомого сигнала и в соответствующем дополнительном оборудовании для поиска этого сигнала.

4.5. Программирование бортовых микрокомпьютеров БПЛА

Выше были показаны преимущества группового взаимодействия и принцип построения системы управления БПЛА, основанный на трехуровной архитектуре. Такая архитектура системы управления БПЛА харрактерисуется разделением конкретных «обязанностей» между вычислительными блоками. Рассмотри логику и принципы разработки программ для трехуровневой архитектуры.

При работе трехуровневая система формирует управляющие воздействия на основе входных данных. Для корректного выполнения задания при создании управляющего воздействия в программе автопилота по принципу разных контуров управления формируются три последовательных набора данных, обеспечивающих:

- поддержание заданных параметров движения (управляющие воздействия на исполнительные механизмы, генерируемые автопилотом на основании данных, полученных от микрокомпьютера и систем навигации);
- корректировку курсу (данные для управления передаются автопилоту и формируются на основе данных навигационной системы и поставленной задачи для мобильного объекта);
- выполнение маршрута движения (формируются на базовой станции в зависимости от поступающей информации от мобильных объектов и поставленной задачи оператором).

Все три контура управления в конечном итоге формируют управляющий сигнал на исполнительные механизмы.

Схематично три контура показаны на рис. 4.5.

Первый контур - создание управляющего сигнала на основе поступающей информации из инерциальной и спутниковой систем навигации. Автопилот отслеживает отклонение мобильного объекта от заданных параметров движения на основе поступающей

информации с инерциальных датчиков (таких, как инфракрасные датчики, магнитометры и гироскопы) и корректирует углы крена, скорость, высоту и др. для поддержания заданных характеристик движения мобильного объекта.

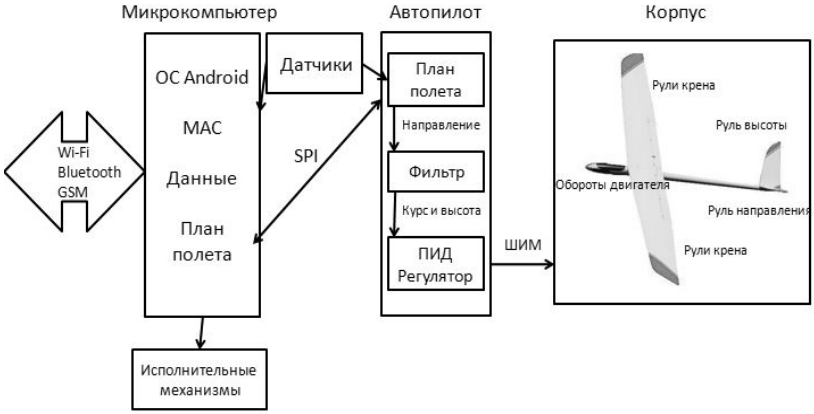


Рис. 4.5. Три контура управления.

Автопилот формирует управляющие воздействия на сервомеханизмы с применением пропорционально-интегрально-дифференциального (ПИД) регулятора. Это наиболее часто используемый вид управления с обратной связью. Если $u(t)$ - управляющее воздействие, $y(t)$ - измерения, $r(t)$ - желаемый результат измерения, $e(t) = r(t) - y(t)$ – «невязка», то ПИД регулятор имеет следующий вид:

$$u(t) = K_P\, e(t) - K_I \int e(t)dt + K_D \dot{e}(t),$$

где K_P, K_i и K_D - коэффициенты усиления пропорциональной, интегральной и дифференциальной составляющей регулятора. В устойчивых системах стабилизация движения часто может быть обеспечена только с помощью пропорциональной составляющей. Интегральный член компенсирует статистическую ошибку, а

дифференциальный служит для компенсации задержек в системе. Основой для составление управляющего сигнала являются данные, получаемые с различных датчиков и источников получения дополнительной информации (например, информация с базовой станции, полученная через модуль связи, и т. п.).

Второй контур необходим для формирования поправки на отклонение от заданного направления движения и корректировки курса. Одна из существенных причин отклонений легких мобильных объектов от заданного курса - изменения направления ветра или неровности поверхности земли. Также на отклонения влияют случайные и систематические ошибки в получаемых в процессе полета данных: ошибки в навигационных системах, дискретность в получении данных и т.п. Для борьбы с этой проблемой применяют фильтры, с помощью которых производится предсказание смещения в следующий момент времени и внесение на основе этих данных поправок в курс БПЛА.

Для любого изменения в конструкции БПЛА или в программе его полета необходимо создавать новый код автопилота. Для того, чтобы каждый раз не заходить в программный код, в автопилоте предусмотренно изменение плана полета в процессе его выполнения. Как правило это доступный для коорректировок текстовый фаил, в котором хранятся данные о координатах.

Обмен информацией между микрокомпьютером и автопилотом можно осуществлять посредством последовательного периферийного интерфейса (SPI, Serial Peripheral Interface).

В третьем контуре управления на основе поставленной глобальной задачи для группы мобильных объектов и частной задачи для члена группы выбираются точки маршрута. Первоначально точки маршрута формируются оператором на базовой станции на основе поставленной задачи. Глобальная задача записывается в микрокомпьютер, а начальные частные задачи для членов группы - в их автопилоты. В процессе полета изменения глобальной задачи посылаются микрокомпьютерам мобильных объектов группы, которые эти изменения обрабатывают и формируют новые частные задачи для автопилотов. Также, исходя из принятой информации от других мобильных объектов, в рамках

глобальной задачи для более быстрого ее завершения микрокомпьютер может поменять частную задачу для автопилота. В описанном выше примере трехуровневой архитектуры обмен информацией между мобильными объектами и между мобильным объектом и базовой станцией происходит посредством различных модулей связи: радио-, Wi-Fi-, Интернет-модема. Связь посредством GSM модема с выходом во всемирную систему компьютерных сетей Интернет в основном предусмотрена для передачи необходимой информации на сервер, доступ к которому можно осуществить с любого компьютера, подключенного к «всемирной паутине».

Конкретизируем алгоритм (протокол) взаимодействия мобильных объектов в группе без единого центра управления, но с единым центром сбора данных мониторинга некоторой территории с помощью различного оборудования для таких задач, как:

- визуальный поиск объекта на местности;
- поиск источника радиосигнала;
- поиск утечки газопродуктов;
- поиск источника магнитного излучения;
- поиск источника радиационного излучения и т.п.

Каждой комплекту данных (фотографии, набору данных по замеру химического состава и т.п.) приписывается свой индекс, который соответствует времени фиксации и координатам точки, в которой произведен сбор данных.

Если установилась связь с другим мобильным объектом, то происходит следующее:

- отправка набора индексов имеющейся информации;
- прием набора индексов;
- сравнение набора принятых индексов с имеющимся набором индексов;
- выделение новых индексов из принятого набора;
- отправка запроса на получение новых данных с новыми индексами;
- принятие запроса на отправку индексов;
- формирование пакета данных на основе запрошенных индексов;
- отправка данных;
- принятие новых данных, запись в память.

На основе новых данных микрокомпьютером может быть принято решение о смене программы для автопилота для корректировки района поиска.

Применение группы мобильных объектов также повышает гарантию выполнения задачи мониторинга местности. Например, возможна ситуация, при которой один из мобильных агентов выходит из строя. Это может быть резкое падение одного из мобильных объектов или выход из строя средств связи этого члена группы. Для завершения выполнения глобальной миссии для группы возникает задача изменить маршруты других объектов для покрытия всего исследуемого участка. При этом алгоритм работы мобильного объекта такой:

- завершить свою задачу;
- сравнить имеющиеся данные со всей картой необходимых данных в глобальной задаче;
- выделить область данных соседних участков;
- определить незаполненные данные в соседних участках;
- если есть незаполненные данные, то сформировать новый маршрут для автопилота;
- если все данные заполнены, отправить сообщение о завершении задачи.

При выполнении такого алгоритма мобильный объект после завершения своей задачи определяет, нужно ли помочь соседям. Если да, то расширяет поле своей работы.

При отправке данных на базовую станцию, необходимость хранить эти данные в микрокомпьютере отпадает, и в базе данных ставится пометка о времени отправки этой информации на сервер. Далее при необходимости освобождения места она стирается из памяти микрокомпьютера. При мониторинге территории данные с близкими координатами отмечаются как повторные, но с разным временем получения, и сравниваются на сервере с целью поиска изменений.

Обмен информацией мобильного объекта с базовой станцией происходит в тот момент, когда первый входит зону видимости базовой станции. Такая связь осуществляется за счет стабильного радиоканала с дальностью действия 1-2 км. На большем удалении

мобильный объект может передавать информацию через подключение к глобальной сети Интернет через связь GSM. Передача информации по такому каналу может быть прервана из-за сбоев в связи при использовании его в воздухе и на значительном удалении от вышек GSM. В связи с этим алгоритм отправки информации такой:

- создать пакет с полученными данными в установленном интервале времени (например, фото- изображение сделанные за первую минуту сбора данных);
- сжать пакет;
- если нет пакетов в очереди отправки, то отправить сжатый пакет;
- если есть очередь отправки, то поставить пакет в очередь отправки;
- после отправки пакет и получения подтвержения доставки поставить отметку в индексах – отправлено на сервер. В противном случае пакет считается неотправленным.

В ЦОДе данные обрабатываются для извлечени необходимой информации. Для обработки фотографий применяются программные приложения с алгоритмами распознавания образов, выделяются фотографии одинаковой местности и на них определяются изменения. На выходе потребитель получает фотографии с отмеченной наиболее важной информацией. Одним из результатов обработки данных может стать формирование новой глобальной миссии для группы мобильных объектов.

4.6. Лабораторная работа № 4 «Пример мультиагентного приложения с JADE-агентом на платформе Android»

4.6.1. Цель лабораторной работы

Создать несколько рабочих контейнеров в мультиагентной платформы JADE, определить взаимосвязь между

несколькими контейнерами, перенести один из контейнеров на мобильное устройство, запустить приложение в Android emulator, запуск DummyAgent на мобильном устройстве с ОС Android.

4.6.2. Инструкция по выполнению лабораторной работы

4.6.2.1. Подготовка

Перед началом работы необходимо запустить среду разработки Eclipse IDE. Запустить платформу и агента PingAgent (см. лаб. 3.) Необходимо скачать платформу JADE-LEAP Android (Lightweight Extensible Agent Platform) runtime environment. Также необходимо будет настроить Android виртуальное устройство с поддержкой x86 архитектуры для отладки приложения под наш планшет Intel.

Для того чтобы получить JADE-LEAP необходимо перейти на официальный сайт платформы JADE - http://jade.tilab.com/. Далее во вкладке «Add-ons» скачать полный архив Leap и JADE-Android в виде архивов .zip. Распакуйте оба архива в папку, где находятся полный пакет файлов платформы JADE (см. лаб. 3). В нашем примере мы получаем в папке \jade дополнительные папки *\add-ons* и *\leap*. В папка add-ons обязательно должны находится подпапки:

- doc – содержит документацию;
- lid – содержит архив .jar, в котором находятся собранные классы;
- src – ресурсный файл, которого нет в Jade Leap;
- demo – содержит проект DummyAgent.

4.6.2.2. Выполнение работы

Для запуска JADE-LEAP в Windows необходимо прописать параметр JAVA_HOME. Если он у вас еще не прописан сделать это можно перейдя во вкладку *Компьютер -> Свойства системы* далее перейти в *Дополнительные параметры системы* в левой части появившегося окошка. Нужная нам директория *Системные переменные* находится в опции *Переменные среды*. Создаем новую Системную переменную с названием JAVA_HOME и прописываем путь в

JRE, в нашем примере путь к необходимым нам файлам следующий C:\Program Files\Java\jre7.

В нашем примере мы хотим запустить агента на планшете Intel с процессором Intel Atom. Для тестирование приложения удобно использовать Android Virtual Device. В дальнейшем, на этом виртуальном устройстве можно будет запустить еще одного агента в системе.

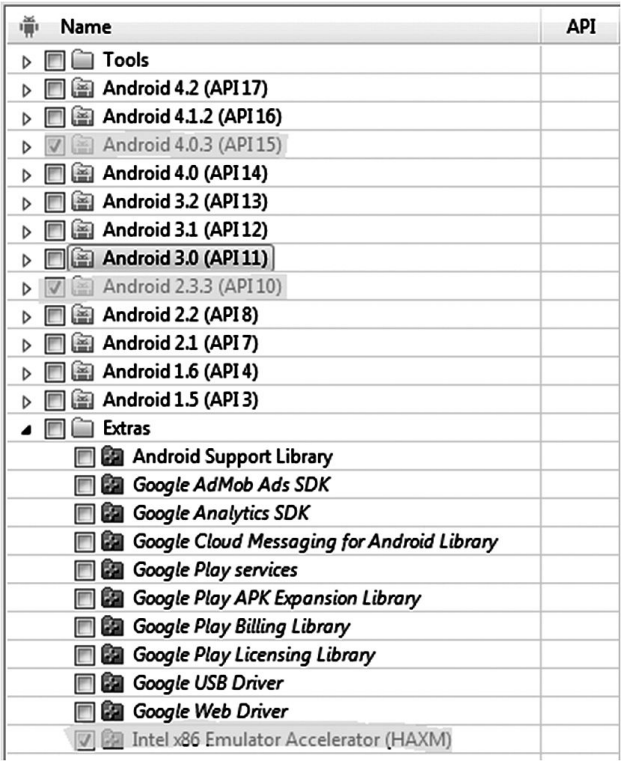


Рис. 4.6. Установка необходимых компонент.

Для настройки виртуального устройства необходимо перейти в Android SDK Manager и установить все элементы к версиям Android 2.3.3 и 4.0.3, обязательно проверить, что в версии 4.0.3 в эти элементы входит Intel x86 Atom System Image (рис. 4.6). Именно с версии 4.0.3 началась официальная поддержка Intel x86 Atom System Image. А также, нам понадобится Intel x86 Emulator Accelerator

(HAXM). The Intel Hardware Accelerated Execution Manager (Intel® HAXM) – аппаратная поддержка визуализации, которая использует технологии Intel Virtualization Technology (Intel® VT) для ускорения эмуляции Android.

Далее нужно перейти в Android Virtual Manager, где мы создадим новое виртуальное устройство для тестирования нашего приложения. Войдя в меню Android Virtual Manager создаем новое устройство с необходимыми нам параметрами (рис. 4.7). Т.к. мы будем создавать приложение под планшет Intel, то желательно создать виртуальное устройство с приближенными параметрами.

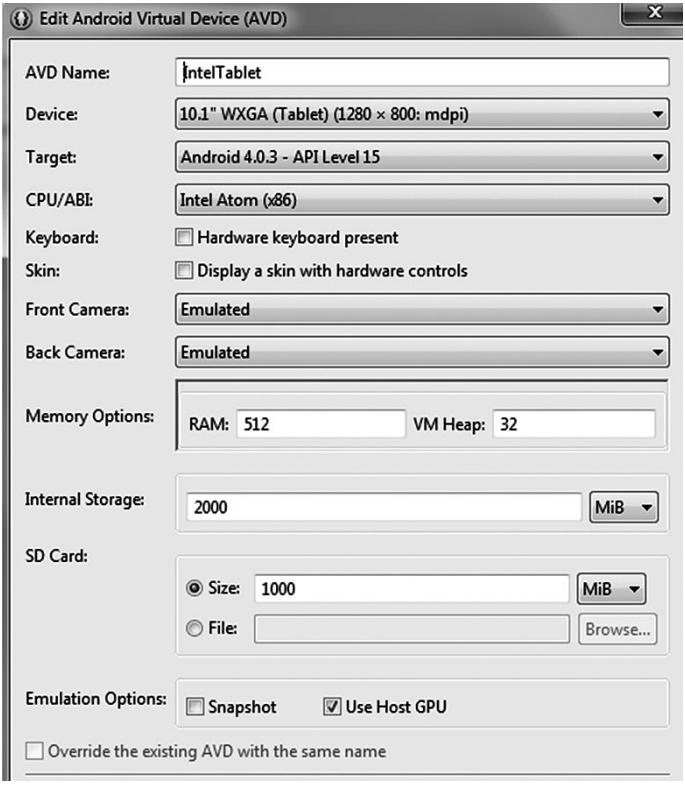


Рис. 4.7. Установки виртуального устройства.

Пропишем необходимые параметры виртуального устройства на рис. 4.7. В качестве имени можете ввести любое. Наше устройство ближе является планшетом с сенсорным экраном 10.1, выбираем платформу Android 4.0.3. с API Level 15. При выборе Android 4.0.3 появляется возможность поставить Intel Atom (x86) в графе CPU (выбор процессора), это нам и нужно для нашего устройства.

Галочки в графах Keyboard и Skin мы не устанавливаем. Camera Front и Back оставим виртуальные, т.к. в нашем приложении камера пока не используется хотя в реальном устройстве она есть. Оперативную память устанавливаем 512Mb, а память используемая приложением VM Heap – 32. Такие параметры оказались оптимальными для эмуляции на нашем ноутбуке. Также устанавливаем внутреннюю память 2000 Mb и внешнюю 1000 Mb память. Также, ставим галочку на Use Host GPU (Graphics processing unit) для ускорения работы нашего эмулятора. Осталось только запустить Intel x86 Emulator Accelerator (HAXM) и можно работать с эмулятором под Intel Atom – ОС Android. Это можно сделать, перейдя в каталог C:\Program Files (x86)\Android\sdk\extras\ intel\ Hardware_ Accelerated_Execution_Manager и запустив IntelHaxm.exe. Теперь можно запустить наше виртуальное устройство перейдя в меню Android Virtual Manager. Выбираем наше устройство и нажимаем *start*.

Контейнер платформы JADE запускается при записи аргумента как *–container*. Чтобы запустить контейнер в Eclipse IDE необходимо сначала запустить Main Container, в нашем примере с главным контейнером компилируется и агент Ping (см. лаб. 3). После запуска Main Container надо перейти обратно в проект Eclipse и запустить его с аргументом *–container*, это можно сделать перейти во вкладку *RunConfiguration* (рис. 4.8).

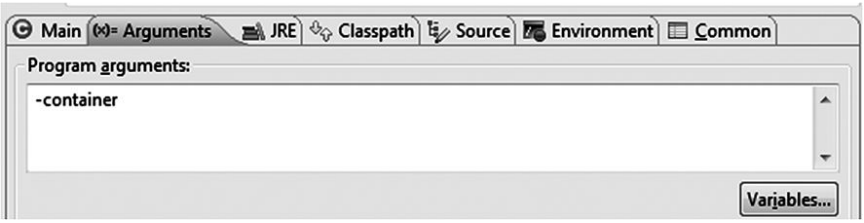


Рис. 4.8. Изменение Program argument.

После запуска в пользовательском интерфейсе платформы получаем дополнительный контейнер Container-1, который пока не содержит агентов. Нажав, левой кнопкой мыши на новый контейнер, перейдя в StartNewAgent запускаем агента Ping (см.лаб. 3)

Для того чтобы передать простое сообщение на созданного нами Ping агента нам необходимо открыть агента DummyAgent. Такой агент можно открыть с мобильного устройства с ОС Android. Перед тем как использовать наш планшет запустим DummyAgent в настроенном нами виртуальном устройстве. В открытом рабочем пространстве необходимо импортировать проект DummyAgent из скаченного нами ранее архива Jade-Android. Заходим во вкладку *File -> Import -> General - > Existing Projects into Workspace*. Находим нужный нам проект в папке *jade/add-ons/jade4android/demo* и нажимаем Import. Теперь необходимо добавить нужные библиотеки Jade. Они находятся в папке *add-ons/jade4android/lib*. Добавить библиотеки можно через вкладку *Project -> Properties -> JavaBuild -> Library*.

Для запуска агента на Виртуальном устройстве перейдите во вкладку RunConfiguration. Выберите свой проект и виртуальное устройство. Нажмите кнопку *Apply* а затем *Run*. Запустится виртуальное устройство и DummyAgent. Теперь отправьте сообщение на созданный нами Ping агент в Container-1.

После запуска приложения мы можем найти готовый установочный файл .apk в папке .bin нашего проекта.

4.6.3. Задания для самостоятельной работы

1. Запустить установочный файл на вашем устройстве.
2. Передать сообщение на Ping агент.
3. Запустить готовый пример ChatClient скачав его по ссылке http://jade.tilab.com/ в каталог examples.

4.7. Выводы

В этой главе были рассмотрены приемущества мультиагентной сети группы лёгких БПЛА по сравнению с группой одиночных комплексов БПЛА. За счёт мультиагентного взаимодействии в группе, каждый БПЛА-агент способен автономно корректировать свою задачу при изменении обстановки в окружающей среде или при исзменении глобальной задачи. Для создания мультиагентной сети БПЛА приведен новая трехуровневая система управления группой. Аппаратная реализация такой системы управления реальна за счёт внедрения среднего слоя – микрокомпьютера, который обменивается данными с другими микрокомпьютери БПЛА и базовой станцией, а также корректирует задачу автопилоту. Рассмотренны примеры двух основных типов задач для группы БПЛА – задача мониторинга местности и задача оптимизации полёта БПЛА. Алгоритм мониторинга местности показан на примере исследование экологической обстановки в акватории. Показано, что для любого визуального исследование территории с помощью группы БПЛА алгоритм будет отличатся только в типе источника сигнала, а в аппаратной части самолёта только отличие будет состоять в типе оборудование [14-16].

4.8. Контрольные вопросы

1) Комплекс электронного оборудования в современных радиоуправляемых самолетах может включать в себя следующих элементов:
   - исполнительные механизмы
   - мотоустановку
   - приемник радиосигнала
   - телеметрии
   - стабилизирующую электронику
   - дополнительные исполнительные механизмы
   - автопилот

Варианты ответов:

1. только 4 первых элемента

2. только 5 первых элементов
3. только 6 первых элементов
4. могут включать все эти элементы

2) Современный комплекс наземного оборудования для управления беспилотным летательным аппаратом может состоять из следующих элементов:
   - пульт управления
   - излучатель радиосигнала
   - приемник радиосигнала для телеметрии
   - наземную базовую станцию
   - другие средства связи (радиомодем, Wi-Fi, Bluetooth)

Варианты ответов:
1. только 2 первых элемента
2. только 4 первых элемента
3. могут включать все эти элементы
4. не может включать ни одного элемента

3) Современные исполнительные механизмы могут принимать следующие типы управляющих сигналов:

Варианты ответов:
1. аналоговый
2. цифровой
3. аналоговый и цифровой
4. ни один из этих сигналов

4) Какого типа тяговый электрический двигатель в основном используются в беспилотных летательных аппаратах?

Варианты ответов:
1. бесколлекторный двигатель
2. коллекторного двигателя
3. шаговый
4. с полым ротором

5) Какого типа аккумуляторные батареи, не могут работать при низких температурах (ниже минус 5 град. С)?

Варианты ответов:
1. батареи типа Li-Po

2. батареи типа $LiFePO_4$
3. батареи типа Pb, Ni-Mh, Ni-Cd
4. никакие

6) В современном навигационном оборудовании беспилотных летательных аппаратах применяют системы:

Варианты ответов:

1. ГЛОНАСС
2. GPS
3. обе системы
4. ни одной из них

7) Квадрокоптер, это:

Варианты ответов:

1. новый тип автомобиля
2. летательный аппарат с четырьмя моторами
3. подводная лодка
4. самолет

8) Какая схема построения управления комплексом стабилизации используется в БПЛА в представленном проекте?

Варианты ответов:

1. централизованная
2. сгруппированная
3. модульная
4. не используется

9) Какой датчик установлен практически в каждом мобильном устройстве?

Варианты ответов:

1. фотокамера
2. акустический датчик
3. тепловой датчик
4. градусник

10) Какие возможности дает дополнительный микрокомпьютер на борту мобильного робота?

Варианты ответов:

1. увеличивает инертность за счет дополнительного веса
2. ничего не дает

3. автономность и адаптивность
4. уменьшение сопротивления

11) Из чего, как правило, состоит комплекс БПЛА?
Варианты ответов:
1. Из одного БПЛА и микрокомпьютера
2. Из одного БПЛА и базовой станции под управлением человека
3. Из одного БПЛА и нескольких базовых станций

12) Как обычно происходит обмен данными, если используются несколько БПЛА при использовании двухуровневой системы управления?
Варианты ответов:
1. В воздухе между микрокомпьютерами БПЛА
2. На земле, между базовыми станциями
3. В воздухе между микрокомпьютерами БПЛА и на земле, между базовыми станциями

13) Что находится на верхнем уровне управления при использовании двухуровневой системы управления?
Варианты ответов:
1. Базовая станция
2. Микрокомпьютер
3. Автопилот

14) Что происходит на верхнем уровне управления при использовании двухуровневой системы управления?
Варианты ответов:
1. Определяется глобальная задача для БПЛА, задаются характеристики полета
2. Происходит управление полетом БПЛА
3. Происходит пересылка данных с БПЛА

15) Что предлагается добавить в двухуровневую схему управления для применения мультиагентного подхода к группе БПЛА?
Варианты ответов:
1. Средний уровень, который осуществляется за счет дополнительного средства связи

2. Средний уровень, который осуществляется за счет дополнительного автопилота
3. Средний уровень, который осуществляется за счёт дополнительного микрокомпьютера установленного в БПЛА

16) Что происходит на среднем уровне управления при использовании трехуровневой системы управления?

Варианты ответов:

1. Определяется глобальная задача для группы БПЛА, задаются характеристики полета
2. Обмен информацией с другими членами группы и базовой станцией
3. Происходит управление полетом группы БПЛА

17) Зачем БПЛА, летающим в группе, взаимодействовать друг с другом?

Варианты ответов:

1. Для более эффективного решения общей задачи, а также для избежания столкновений
2. Для пересылки информации на базовую станцию
3. Для перестроения глобальной задачи

18) В чем состоит основная задача автопилота?

Варианты ответов:

1. Обмен данными между БПЛА
2. Управление исполнительными механизмами
3. Формирование начальной программы полета для каждого БПЛА

19) В чем состоит основная задача автопилота базовой станции?

Варианты ответов:

1. Формирование начальной программы полета для каждого БПЛА, определение глобальной миссии, сбор и обработка информации, определение глобальной миссии
2. Изменение программы автопилота, сбор и обработка информации
3. Корректировка глобальной миссии, изменение программы автопилота, сбор информации

20) Где может быть сгенерирована новая программа для автопилота, если старая не поддерживает нужных требований для выполнения общей задачи?

Варианты ответов:

1. На базовой станции
2. На бортовом микрокомпьютере
3. На самом автопилоте

21) С каких устройств бортовой микрокомпьютер может получать данные?

Варианты ответов:

1. Информацию от своих бортовых датчиков и базовой станции
2. Информацию от своих бортовых датчиков, а также от микрокомпьютеров с других БПЛА
3. Информацию от своих бортовых датчиков, а также от микрокомпьютеров с других БПЛА и базовой станции

22) Каким образом осуществляется связь между микрокомпьютерами разных БПЛА?

Варианты ответов:

1. Связь не осуществляется
2. Через GPRS
3. За счет встроенного радиоприемника

23) Как осуществляется взлет БПЛА?

Варианты ответов:

1. С рук оператора или с катапульты
2. С земли
3. С земли или с рук оператора

24) Для чего предназначена специальная программа The Intel Hardware Accelerated Execution Manager (Intel® HAXM)?

Варианты ответов:

1. Для аппаратной поддержки визуализации
2. Для динамического расширения памяти
3. Для взаимодействия с модулями связи в режиме эмуляции

25) С какой версии ОС Android началась официальная поддержка

Intel x86 Atom System Image?

Варианты ответов:

1. 2.3.3
2. 3.0
3. 4.0.3

26) Какие действия необходимы для запуска дочернего контейнера в платформе JADE при работе в Eclipse IDE?

Варианты ответов:

1. Запустить платформу с аргументом *–container*
2. Запустить платформу с аргументом *–gui*
3. Запустить платформу дважды с аргументами *–gui* для первого запуска и *–container* при втором запуске

Список литературы

1. *Граничин О.Н, Павленко Д.В.* Рандомизация получения данных и L1- оптимизация (опознание со сжатием) (Обзор) // Автоматика и телемеханика . 2010. № 11.
2. *Амелин К.С., Антал Е.И., Васильев В.И., Граничина Н.О.* Адаптивное управление автономной группой беспилотных летательных аппаратов // Стохастическая оптимизация в информатике. Вып. 5. 2009. С. 157-166.
3. *Spall, J. C.,* Multivariate stochastic approximation using a simultaneous perturbation gradient approximation // IEEE Trans. Automat. Contr. Vol. 37. 1992. P. 332–341.
4. *Granichin, O.N.,* A stochastic recursive procedure with dependent noises in the observation that uses sample perturbations in the input // Vestnik Leningrad Univ. Math. 1989. Vol. 22. No. 1(4). P. 27–31.
5. *Granichin, O.N.,* Procedure of stochastic approximation with disturbances at the input // Automation and Remote Control. 1992. Vol. 53 No. 2, part 1. P. 232–237.
6. Официальный сайт по беспилотным летательным аппаратам // http://bp-la.ru/.

7. *Скобелев П.О.* Открытые мультиагентные системы для оперативной обработки информации в процессах принятия решений // Автометрия. 2002. № 6. С. 45-61.
8. *Baxter J.W., Horn G.S., Leivers D.P.* Fly-by-Agent: Controlling a Pool of UAVs via a Multi-Agent System. QinetiQ Ltd Malvern Technology Centre St Andrews Road. Malvern. UK. 2007.
9. *Амелин К.С.* Лёгкий беспилотный летательный аппарат для автономной группы // Стохастическая оптимизация в информатике. Вып. 6. 2010. С. 117-126.
10. *Haiyang Chao.* Cooperative Remote Sensing And Actuation Using Networked Unmanned Vehicles. PhD thesis. Utah state university. 2010.
11. Сайт Jade: http://jade.tilab.com/doc/index.html
12. Статьи по разработки адаптивного управления для группы автономных мобильных роботов
13. Официальный сайт разработчиков Android разработки http://developer.android.com/index.html https://sites.google.com/site/smartflyllc/file-cabinet
14. *Амелин К.С., Граничин О.Н.* Мультиагентное сетевое управление группой легких бпла // Нейрокомпьютеры: Разработка, Применение, 2011, № 6, С. 64-72.
15. *Amelin K., Granichin O.* Multiagent network control for the group of UAVs // In: Proc. of the 5th International IEEE Scientific Conference on Physics and Control (PhysCon 2011), León, Spain, Sept., 2011.
16. *Амелин К.С., Граничин О.Н.* Применение мультиагентного подхода для решения задач мониторинга местности группой легких БПЛА // В сб. тр. межд. научно- практ. конф. Теория активных систем. 14-16 ноября 2011 г., Москва, Россия. Том 3. Ред. – В.Н. Бурков, Д.А. Новиков. – М.: ИПУ РАН, 2011. – С. 209-214.

5. Мультиагентное управление

Мультиагентное управление в сетях. Балансировка загрузки децентрализованной вычислительной сети.

5.1. Введение

Совместное управление или управление распределенными динамическими системами на графах относится к ситуации, в которой каждый узел может получать информацию для проектирования управления только от самого себя и от своих соседей. Граф может задавать топологию сети связей, которая ограничивает связи между узлами. Это также называют мультиагентным управлением.

Несмотря на большое количество публикаций по этой тематике, пока удовлетворительные решения получены лишь для ограниченного класса практически важных задач, так как решение таких проблем существенно усложняется, с одной стороны, из-за обмена неполной информацией, которая, кроме того, обычно измеряется с помехами, а, с другой, из-за эффектов квантования (дискретизации), свойственных всем цифровым системам.

5.2. Цели управления

Остановимся на различных целях управления, которые решают системы мультиагентного управления (МАУ).

Синхронизация — совпадение или сближение переменных состояния двух или нескольких систем, либо согласованное изменение некоторых количественных характеристик систем.

Задача синхронизации отличается от задачи управления с эталонной моделью, поскольку в ней допускается совпадение различных переменных, взятых в различные моменты времени. Временные сдвиги могут либо быть постоянными, либо стремиться к постоянным. Кроме того, во многих задачах синхронизации связи ме-

жду системами являются двусторонними (двунаправленными). Это значит, что предельный режим в системе (синхронное решение) заранее не известен. Общей особенностью задач управления синхронизацией является то, что желаемое поведение однозначно не фиксировано, а его характеристики задаются лишь частично. В задачах синхронизации часто основным требованием является совпадение или согласованность колебаний всех подсистем, в то время как характеристики движения каждой подсистемы могут варьироваться в широких пределах [1]. В контексте МАУ под синхронизацией понимают согласованное поведение агентов, например, полное или частичное сближение со временем состояний агентов или их наблюдаемых выходов.

В МАУ такая цель управления чаще формулируется как задача *достижения консенсуса* [2,3] или *согласования характеристик* [4], в которых каждый агент стремится уменьшить отклонение своей целевой переменной от соответствующих переменных своих соседей. Если речь идет об отклонении от среднего арифметического состояний соседей, то ставится задача достижения *усредненного консенсуса*.

Задача по переводу группы агентов в некоторое общее состояние называется задачей о *рандеву* (англ. rendezvous). В ней требуется, чтобы все агенты оказались в заданном месте в заданное время.

Еще одним классом задач МАУ является *распределенное управление формациями* (англ. formation control) — управление локально взаимодействующими агентами, образующими некоторые геометрические конфигурации. Такие задачи обычно решаются применительно к мобильным роботам, беспилотным летательным аппаратам, подводным автономным устройствам и др. Задачи группового управления (управления формациями) [5] сводятся к задачам частичной координатной синхронизации с помощью метода виртуального лидера.

При большом числе агентов (а в ряде задач число агентов достигает тысяч и миллионов), требование заданного поведения всех без исключения агентов оказывается излишне жестким и трудновыполнимым. В таких случаях выделяется характерная точка в множестве состояний агентов (центр, лидер, центр тяжести), а же-

лательным поведением является заданное поведение центра при условии ограниченности отклонений от него состояний всех агентов. Для такого поведения вводятся понятия, заимствованные из биологии: *сбивание в стаи, роение* (англ. swarming, flocking), и устанавливаются условия возникновения в системе соответствующих свойств [6,7].

Отдельный класс задач МАУ — распределение ресурсов между разными возможными заданиями, задачи *диспетчеризации* (англ. scheduling), которые сводятся к задаче достижения консенсуса.

5.3. Балансировка загрузки узлов децентрализованной вычислительной сети при неполной информации

Распределенное взаимодействие в сетях динамических управляемых агентов привлекает в последние время внимание все большего числа исследователей. Во многом это объясняется широким применением мультиагентных систем в разных областях, включая автоматическую подстройку параметров нейронных сетей распознавания, управление формациями [1], роение [2], распределенные сенсорные сети [3], управление перегрузкой в сетях связи [4], взаимодействие групп беспилотных летательных аппаратов (БПЛА), относительное выравнивание групп спутников и др. Многие из таких задач легко переформулируются в терминах достижения консенсуса в мультиагентных системах [5-7].

Решение таких задач существенно усложняется при практическом применении, с одной стороны, из-за обмена неполной информацией, которая, кроме того, обычно измеряется с помехами, а, с другой, из-за эффектов квантования (дискретизации), свойственных всем цифровым системам [8-12].

Для группы взаимодействующих агентов, обменивающихся с задержкой неполной информацией в дискретные моменты времени, при изменяющейся топологии связей в [13] предложен и обоснован алгоритм стохастической аппроксимации для решения задачи о достижении консенсуса. Алгоритмы типа стохастического градиента использовали в такого типа задачах и ранее [14-17]. Стохастическая аппроксимация с убывающим размером шага

позволяет каждому агенту получать информацию о состоянии своих соседей при одновременном снижении воздействия помех. Популярным инструментом для доказательства состоятельности этих алгоритмов являются квадратичные функции Ляпунова, наличие которых гарантировано при фиксированной топологии сети.

В случае сети с переменной топологией квадратичная функция Ляпунова в [6] применяется для случая сбалансированной модели графа связности, матрицы которого стохастические по строкам и столбцам. В [9] для неориентированных графов стохастические по строкам и столбцам весовые матрицы строились на основе весов Метрополиса. Распределенные итеративные алгоритмы для построения на ориентированных графах (орграфах) стохастических матриц по строкам и столбцам были предложены в [18], но они не применимы при отсутствии полной информация о текущей топологии сети.

Практическое значение имеет рассмотрение моделей без свойства стохастичности матриц по строкам и столбцам, как в [13]. Кроме того, желательно иметь алгоритмы достижения консенсуса, работоспособные и в нестационарных условиях.

При динамических внешних изменениях состояний агентов (получение новых заданий и т. п.) алгоритмы стохастической аппроксимации с уменьшающимся размером шага не применимы. В [19-22] исследуется работоспособность алгоритмов стохастической аппроксимации с постоянным размером шага в условиях нестационарных функционалов качества (среднего риска). Их применимость для балансировки загруженности узлов централизованной вычислительной сети при доступности текущей зашумленной иформации о загруженности и производительности узлов была исследована в [23, 24] для разработки специальной программы - брокера загрузки.

В этой лекции будут рассмотрены возможности применения результатов [13] к задачам о балансировке загруженности узлов децентрализованной вычислительной сети при поступлении в каждый узел только зашумленной информации о загруженности и производительности соседей, причем топология связей в сети

может меняться со временеи, а информация от соседей доходить с задержкой по времени.

5.4. Задача консенсуса на графах

Поясним обозначения, использующиеся далее. Верхний индекс у переменных будем используется в качестве индекса, а не показателя степени. Для матрицы A элемент, находящийся на ее i-й строке и в j-м столбце называется (i, j)-м элементом и обозначается как $a^{i,j}$. Для вектора или матрицы M определим *норму Фробениуса* $|M| = [Tr(M^T M)]^{1/2}$. Будем использовать $1_k \in \mathrm{P}^k$, чтобы определить вектор-столбец из k единиц. Для вектор-столбцов $Z_1, \ldots, Z_l, [Z_1; \ldots; Z_l]$ определяет вектор-стоблец, полученный вертикальным соединением l векторов.

Для описания топологии сети будем использовать понятия теории графов. *Ориентированный граф (орграф)* $G = (N, E)$ состоит из множества узлов $N = \{1, \ldots, n\}$ и множества ребер E. Ребро определяется упорядоченной парой $(i, j) \in N \times N$, где $i \neq j$. *Направленный путь* (из узла i_1 в узел i_l) состоит из последовательности узлов $i_1, \ldots, i_l$, $l \geq 2$ таких, что $(i_k, i_{k+1}) \in E$. Орграф G называется *сильно связным*, если из любого узла в любой другой узел существует направленный путь. *Направленное дерево* – это орграф, в котором каждый узел i, кроме корня, имеет ровно одного родителя j такого, что $(j, i) \in E$. Будем называть $\overline{G} = (\overline{N}, \overline{E})$ *подграфом* G, если $\overline{N} \subset N$ и $\overline{E} \subset E \cap \overline{N} \times \overline{N}$. Будем говорить, что орграф G содержит *остовное дерево*, если существует направленное дерево $G_{tr} = (N, E_{tr})$ как подграф G. *Матрицей связности* графа G называется матрица размером $n \times n$ такая, что $A_G = (a^{i,j})_{1 \leq i, j \leq n}$, где $a^{i,j} = 1$, если $(i, j) \in E$ и $a^{i,j} = 0$ в остальных случаях. Если G – неориентированный граф, то каждое ребро определяется как неупорядоченная пара (i, j), где $i \neq j$.

Топология динамической сети, показывающая принимаемые сигналы моделируется с помощью последовательности орграфов $\{G_t = (N, E_t)\}_{t \geq 0}$, где $N = \{1, \ldots, n\}$, а каждое $E_t \subset E$ и случайно меняется во времени. Матрица связности A_{G_t} – матрица, которая полностью определяет E_t. Если $(j, i) \in E_t$, то говорим, что узел i получает информацию от узла j, который называется *соседом* узла i. Обозначим $N_t^i = \{ j \mid (j, i) \in E_t \}$ – множеством соседей узла i. Множество соседей подмножества $N_{\overline{N}}$ определяется следующим образом:

$$N_{\overline{N}} := \bigcup_{i \in \overline{N}} N_t^i = \{ j \in N : i \in \overline{N}, (i, j) \in E \}. \quad (1)$$

Пусть $x_t^i \in R$ определяет состояние узла i в момент времени $t \in \{0,1,2,\ldots\}$. Определим *информационный поток* в сети как $G_x = (G_t, X_t)$, где $X_t = [x_t^1, \ldots, x_t^n], t \geq 0$ набор значений состояний узлов сети, $X_t \in \mathrm{P}^n$, а G_t – *топология*. Состояниями узла могут быть, например, физические характеристики (положение, температура, напряжение и др.). В динамической сети с переменной топологией информационный поток G_x – это дискретное состояние системы, меняющееся во времени. Будем говорить, что узлы i и j *согласованы* в сети тогда и только тогда, когда $x_t^i = x_t^j$, и узлы *достигли консенсуса* тогда и только тогда, когда $x_t^i = x_t^j \ \forall i, j \in N, i \neq j$. Когда все узлы сети согласованы, общее значение узлов называется *групповым решением*.

Если узлы графа – *динамические агенты*, описываемые уравнениями:

$$\dot{x}_t^i = f^i(x_t^i, u_t^i), \ i \in N, \quad (2)$$

то *динамический граф*(или *динамическая сеть*) – это динамическая система с состояниями (G_t, X_t), в которой X_t меняется в

соответствии с *динамикой сети* $\dot{X}_t = F(X_t, U) = [f^1(x_t^1, u_t^1, \ldots, f^n(x_t^n, u_t^n]$, где $U = [u_t^1, \ldots, u_t^n]$.

Будем считать, что в момент времени t, если $N_t^i \neq \varnothing$ узел i получает, возможно, устаревшую информацию от своих соседей, моделируемую следующим образом:

$$y_t^{ik} = x_{t-d_t^{ik}}^k + w_t^{ik},\ k \in N_t^i, \qquad (3)$$

где w_t^{ik} – помехи, а $d_t^{ik} \geq 0$ – целочисленная случайная задержка. Так как система начинает работу при $t = 0$, неявным требованием к множеству соседей будет:

$$k \in N_t^i \Rightarrow t - d_t^{ik} \geq 0. \qquad (4)$$

Каждый узел использует информацию о своем собственном состоянии (может быть и зашумленную), а также свои зашумленные измерения для u_t^i. Будем называть обратную связь по наблюдениям состояний

$$U_t = k_t(X_t), u_t^i = k_t^i(y_t^{j_1}, \ldots, y_t^{j_{m_i}}) \qquad (5)$$

протоколом с топологией G_t, $\{j_1, \ldots, j_{m_i}\} \in \overline{N}^{i,t} \subseteq \{i\} \cup N_t^i\ \ \forall i$ из узлов $j_1, \ldots, j_{m_i} \in N$ удовлетворяет свойству: $\overline{N}^i \subseteq \{i\} \cup N_t^i$. Если $|\overline{N}_i| < n\ \ \forall i \in N$, то (5) называется *распределенным протоколом*.

Пусть $\chi : \mathrm{P}^n \to \mathrm{P}$ – некоторая функция n переменных. *Задача* χ*-консенсуса* в динамическом графе заключается в распределенном вычислении $\chi(X_0)$, применяя входы u_t^i.

Определение 1: Протокол (5) асимптотически решает задачу χ-консенсуса тогда и только тогда, когда существует асимптотически устойчивое равновесие X^* для $\dot{X}_t = F(X_t, k_t(X_t))$, удовлетворяющее $x^{*,i} = \chi(X_0))\ \ \forall i \in N$.

Отметим особые случаи, когда $\chi(X) = Ave(X) = 1/n(\sum_{i=1}^n x^i)$, $\chi(X) = \max_i x^i$ и $\chi(X) = \min_i |x^i|$, называемые *консенсус*

усреднения, $\max$*-консенсус* и $\min$*-конснсус* соответственно [6]. Эти случаи широко применяются в распределенных системах принятия решений для мультиагентных систем.

Решение задачи консенсуса усреднения является примером распределенного вычисления линейной функции $\chi(X) = Ave(X)$, используя сеть динамических систем (или интеграторов).

Пусть (Ω, F, P) – основное вероятностное пространство и будем считать, что часть или все определенные выше переменные, вектора и матрицы – случайные величины.

Обозначим максимальное множество каналов связи $E_{\max} = \{(k,i) \mid sup_{t\geq 0}((k,i) \in E_t) > 0\}$.

Для удобства статистического моделирования предположим следующее: w_t^{ik} и d_t^{ik} определены для всех $(k,i) \in E_{\max}$. Если (k,i) не появляется в E_t, тогда (3) физически не работает и w_t^{ik} и d_t^{ik} можно считать нулями. Если $(k,i) \notin E_t$, положим $d_t^{ik} = 0$. Пусть $w_t^{ik} \mid (k,i) \in E_{\max}$ перечислены в определенном порядке (k,i), тогда получаем вектор помех W_t размерности n_1.

Определение 2: n узлов достигают *среднеквадратичного консенсуса*, если $E \mid x_t^i \mid^2 < \infty, t \geq 0, 1 \leq i \leq n$ и существует случайная переменная x^* такая что $\lim_{t\to\infty} E \mid x_t^i - x^* \mid^2 = 0$ для $1 \leq i \leq n$.

Далее будет рассмотрен пример мультиагентной системы с описанными выше параметрами.

5.5. Задача балансировки загруженности сети

В последнее время все чаще при вычислениях используются распределенные системы параллельных вычислений, для которых актуальна задача разделения пакета заданий между несколькими вычислительными устройствами.

Рассмотрим модель системы разделения однотипных заданий между разными узлами для параллельных вычислений с обратной связью. Обозначим $N = \{1,\ldots,n\}$ набор интеллектуальных агентов (вычислительных узлов), каждый из которых обслуживает поступающие заявки на вычисления по принципу очереди, то есть первый вошел - первый вышел. Будем считать, что всем агентам присылают однотипные задания, которые можно раздробить на одинакове по сложности атомарные единицы. Задания поступают в различные моменты времени и на разные узлы. Считается известным размер каждой задачи (или задания), то есть его трудоемкость, в секундах или циклах процессора.

В каждый момент времени t состояние агента i, $i = 1,\ldots,n$, описывается двумя характеристиками:

1) q_t^i – длина очереди из атомарных элементарных заданий в момент времени t

2) p_t^i – производительность узла (количество (или доля) выполненых атомарных заданий за предшествующий такт времени, при условии полной загрузки).

Обозначим $x_t^i = q_t^i / p_t^i$.

Задача для сети агентов – выполнять поступающие последовательно задания. Будем рассматривать две постановки задачи: стационарную и нестационарную.

1. Для начала предположим, что все задания поступают в систему на разные узлы в начальный момент времени. Если все задания выполняются только тем агентом, которому они поступили, и производительности агентов не изменяются со временем, то время выполнения всех заданий определяется так: $T_m = \max_i q_0^i / p_0^i$. Для минимизации времени выполнния всех заданий при возможности их перераспределения между агентами естественно рассматривать задачу оптимизации загрузки сети.

2. При тех же предположениях задача оптимизации загрузки остается актуальной и при поступлении в сеть новых заданий с течением времени. Новые задачи могут поступить прямо на любой из n узлов. Каждый узел i в определенный момент времени t

“видит” только соседей из множества N_t^i. При этом считаем, граф связности полный, т. е. из каждого узла существует цепочка в любой другой. Динамики изменений каждого из узлов при неизменных производительностях описываются следующими уравнениями

$$\begin{aligned} \dot{q}_t^i &= -p_t^i + u_t^i; \\ \dot{p}_t^i &= 0. \end{aligned} \qquad (6)$$

В каждый момент времени t узел i может получить от своих “видимых” соседей $j \in N_t^i$ следующую информацию:

1) наблюдения о том, какова его загрузка $\overline{y}_t^{ij}$;

2) p_t^j – производительность узла.

Задача состоит в том, чтобы составить протокол общения между агентами, при котором все узлы будут загружены равномерно, т. е. $q_t^i / p_t^i \to c_t^*$, независящей от i, т. е. если в систему не будут поступать новые заказы, то все узлы закончат работать одновременно.

На рис. 5.1 приведен пример вычислительной сети из шести агентов с указанием возможных каналов связи, часть из которых может “закрываться” и «открываться» с течением времени.

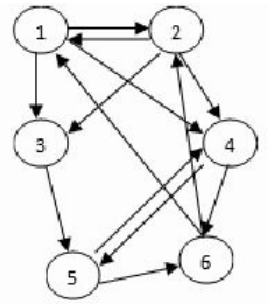


Рис. 5.1. Максимальное множество каналов связи ($E_{\max}$).

5.6. Протоколы консенсуса

В [6] показано, что консенсус асимптотически достижим в двух типичных случаях.

1) Для фиксированной или меняющейся топология и нулевом времени задержки связи при линейном протоколе консенсуса:

$$u_t^i = \sum_{j \in N_t^i} a^{i,j}(x_t^j - x_t^i) \qquad (7)$$

где множество соседей N_t^i узла i меняются в сети с переменной топологией.

2) Для фиксированной топологии и задержкам по времени связи $d_t^{ij} > 0$, соответствующим ребрам $i, j \in E$ при линейном протоколе консенсуса с задержной по времени связи:

$$u_t^i = \sum_{j \in N_t^i} a^{i,j}(x^j_{t-d_t^{ij}} - x^i_{t-d_t^{ij}}) \qquad (8)$$

В случае неориентированных графов для интуитивного обоснования протокола (7) можно воспользоваться следующими соображениями. Определим *потенциал Лапласа*, связанный с неориентированным графом $G = (N, E, A)$ следующим образом:

$$\Phi_G(X_t) = 1/2 \sum_{i,j=1}^{n} a^{i,j}(x_t^j - x_t^i)^2, \qquad (9)$$

и отметим, что обратная связь, основанная на методе скоростного градиента [25] $u_t = -1/2\nabla\Phi_G(X_t)$ совпадает с протоколом (7). В результате динамика сети для агентов с применением протокола (7) выглядит следующим образом:

$$\dot{X}_t = -1/2\nabla\Phi_G(X_t), \qquad (10)$$

которая является *градиентной системой* порожденной графом G.

Для ориентированных сетей вопрос об обосновании протокола консенсуса является более сложным.

Обычно рассматривают два типа протоколов консенсуса, которые решают задачу согласования в сети с непрерывным временем для агентов с динамикой:

$$\dot{x}_t^i = u_t^i \qquad (11)$$

или для модели агентов в дискретном времени:

$$x_{t+1}^i = x_t^i + \alpha u_t^i, \qquad (12)$$

с некоторым размером шага $\alpha > 0$. Мы далее будем рассматривать только дискретный случай.

Для решения сформулированной в разделе 2 общей постановки задачи о построениии протокола консенсуса в [13] предлагается следующий метод стохастической аппроксимации.

Определим матрицу $B_t = (b_t^{i,k})_{1 \le i,k \le n}$ следующим образом: если $N_t^i = \varnothing$, то

$$b_t^{i,k} = 0 \ \ \forall k \in N. \qquad (13)$$

Если $N_t^i \ne \varnothing$, то

$$\begin{cases} b_t^{i,k} \in [\underline{b}, \overline{b}], \quad k \in N_t^i \\ b_t^{i,k} = 0, k \notin N_t^i \cup \{i\} \\ \qquad b_t^{i,i} = -\sum\limits_{k \in N_t^i} b_t^{i,k} \end{cases}, \qquad (14)$$

где $0 < \underline{b} \le \overline{b} < \infty$ – две детерминированных константы. Пока последовательность $\{G_t\}_{t \ge 0}$ меняется случайно, $\{B_t\}_{t \ge 0}$ – матричный случайный процесс.

В момент времени $t \ge 0$, если $N_t^i = \varnothing$, множество $x_{t+1}^i = x_t^i$. Если $N_t^i \ne \varnothing$, узел i меняет свое состояние по правилу:

$$x_{t+1}^i = [1 + \alpha_t b_t^{i,i}] x_t^i + \alpha_t \sum_{k \in N_t^i} b_t^{i,k} y_t^{ik}, \ t \ge 0, \quad (15)$$

где $\{\alpha_t\}_{t \ge 0}$ – последовательность положительных размеров шагов. Будем называть $I + \alpha_t B_t$ – весовой матрицей. Поскольку все узлы обновляют свои состояния одновременно, (15) является синхронным алгоритмом.

Вернемся к задаче о балансировке загрузки узлов вычислительной сети. В качестве ненулевых $b_t^{i,j}$ берем p_t^i/p_t^j. В этом случае $\underline{b} = \min_t p_t^i/p_t^j$, $\overline{b} = \max_t p_t^i/p_t^j < \infty$.

Алгоритм (15) в задаче о балансировке загрузки узлов вычислительной сети принимает вид

$$q_{t+1}^i = [1+\alpha_t b_t^{i,i}]q_t^i + \alpha_t \sum_{j \in N_t^i} b_t^{i,j} \overline{y}_t^{ij}, \ t \geq 0, \quad (16)$$

Задачи балансировки загрузки встречаются не только в вычислительных сетях, но также в производственных, логистических, транспортных и других сетях [33-37].

5.7. Лабораторная работа № 5 «Имитационное моделирование алгоритма локального голосования для балансировки загрузки узлов вычислительной сети»

5.7.1. Цель лабораторной работы

Для иллюстрации теоретических результатов приведем пример имитационного моделирования работы алгоритма локального голосования для системы балансировки загруженности узлов децентрализованной вычислительной сети с использование платформы JADE.

5.7.2. Программная реализация сети агентов

В качестве примера рассмотрим вычислительную сеть, состоящую из шести вычислительных узлов. Топология сети в каждый момент времени случайная [38].

Разработка агентов в JADE похожа на разработку обычных Java-приложений и не требует от программиста специальных технических навыков. Для разработки достаточно иметь Java Development Kit, но для удобства мы использовали традиционную интегрированную среду разработки для Java – Eclipse. Любой агент в системе JADE является классом-наследником класса jade.core.Agent. В наследнике достаточно переопределить метод setup(), в котором агент настраивается для последующей деятельности.

В первую очередь определяется поведение агента. Мы используем интерфейс CyclicBehaviour, который задает циклическое поведение, необходимое для осуществления итераций.

Внутри интерфейса, в методе action(), описывается логика работы с сообщениями, а именно реализуются шаги вычислительной схемы, описанной выше.

Последующая деятельность агентов, в том числе доставка сообщений между ними, обеспечивается средой JADE. Управление агентами, их запуск и остановку, можно осуществлять через JADE Remote Agent Management GUI. Вывод результатов осуществляется в консоль Eclipse, для этого также можно создать специальное окно (рис. 5.2).

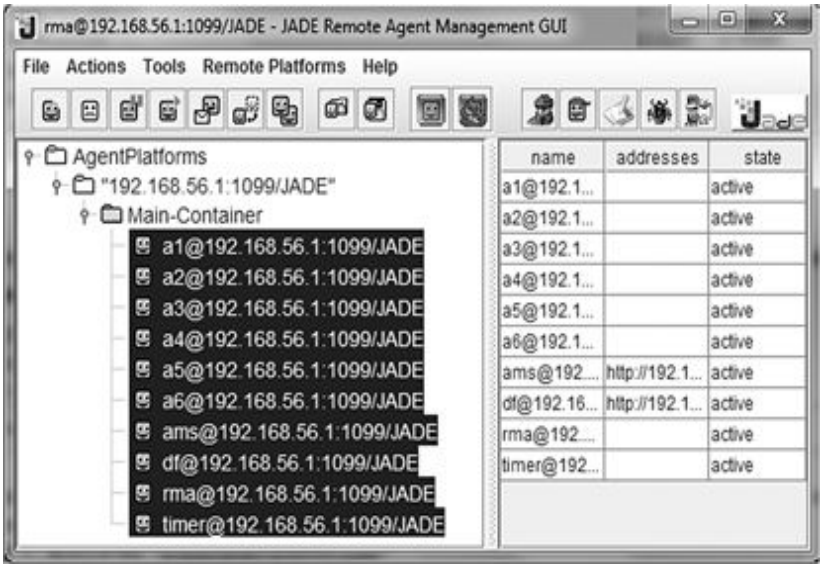


Рис. 5.2. Окно JADE Remote Agent Management GUI.

Для просмотра схемы передачи информации между агентами можно открыть окно Smiffer Agent, в котором стрелками указывается путь передачи единицы информации. Рис. 5.3 показывает sniffer agent для примера реализации алгоритма достижения консенсуса, описанного выше.

Созданы шесть агентов с жестко определенной топологией сети, т.е. каждый агент связан только с двумя другими агентами. Агент «timer» подает команду для начала нового цикла обменом информации. Каждый агент начальный момент времени имеет собственное целое число. После применения протокола локального

голосования и получения данных от других агентов, он записывает себе новое число и отправляет агенту «timer» сообщение со своим номером о том, что отработал такт. Новый такт не начинается, пока все агенты не «отчитаются» перед агентом «timer». Такт системы равен 1 сек. Для подсчета числа текущего такта агент сравнивает его с числом предыдущего такта. Если число агента не меняется в течении пяти тактов, то агент отправляет на «timer» сообщение об достижении консенсуса. Консенсус достигается при отсутствии изменений числа агента в пределе первого знака дробного числа.

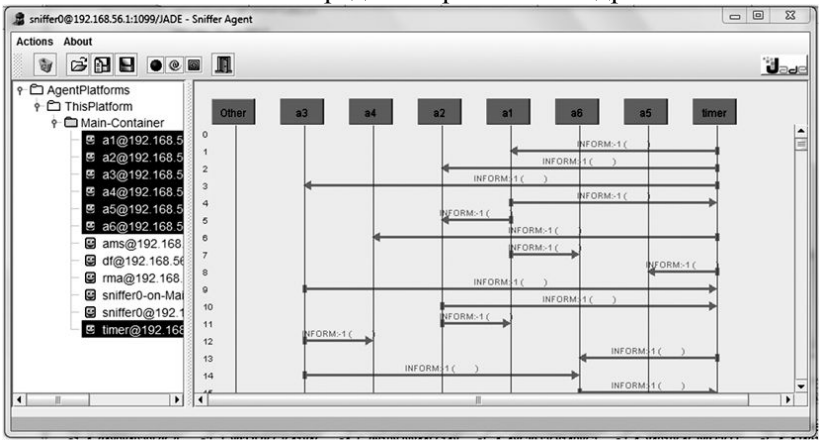


Рис. 5.3. Окно Sniffer Agent.

5.7.3. Инструкция по выполнению лабораторной работы

Рассмотрим два случая: стационарный и случай поступлении в тот или иной узел с течением времени новых заданий [39-41].

Для поставленной задачи будем использовать алгоритм стохастической аппроксимации для достижения консенсуса в следующем виде:

$$x_{t+1}^{i} = x_{t}^{i} + \alpha_{t} \sum_{j \in N_{t}^{i}} y_{t}^{j} / p_{t}^{j} - x_{t}^{i} / p_{t}^{i}, \qquad (17)$$

где α_t – последовательность положительно определенных размеров шагов. x_t^i – состояние узла i в момент времени t, p_t^i – производительность узла i.

Будем проводить имитационное моделирование для системы, показанной на рис. 5.1, состоящей из 6 вычислительных блоков. Были выбраны начальные состояния загруженности узлов:

$$x_0^1 = 3300,\ x_0^2 = 7500,\ x_0^3 = 5200,\ x_0^4 = 4150,\ x_0^5 = 6300,\ x_0^6 = 2100.$$

Производительности узлов:

$$p^1 = 2,\ p^2 = 0.75,\ p^3 = 1.2,\ p^4 = 1.7,\ p^5 = 3.5,\ p^6 = 2.1$$

и они не меняются со временем.

Начальная топология сети приведена на рис. 5.4(A). С течением времени топология сети дважды меняется: рис. 5.4(B) и 5.4(C).

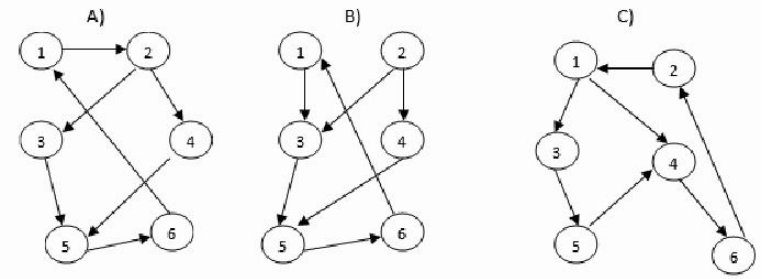


Рис. 5.4. Топология сети.

Матрицы связности для трех случаев выглядит так:

$$A_{G_A} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} A_{G_B} = \begin{pmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

$$A_{G_C} = \begin{pmatrix} -1 & 0 & 1/2 & 1/2 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix} \qquad (18)$$

1) Для простоты сначала рассмотрим стационарный случай – систему, в которой все узлы имеют определенную начальную загрузку, и новые задания не поступают.

График невязки при работе алгоритма (17) для такой системы приведен на рис. 5.5. Система сходится примерно за 75 шагов. Используем коэффициент $\alpha_t = 1/\sqrt{t}$.

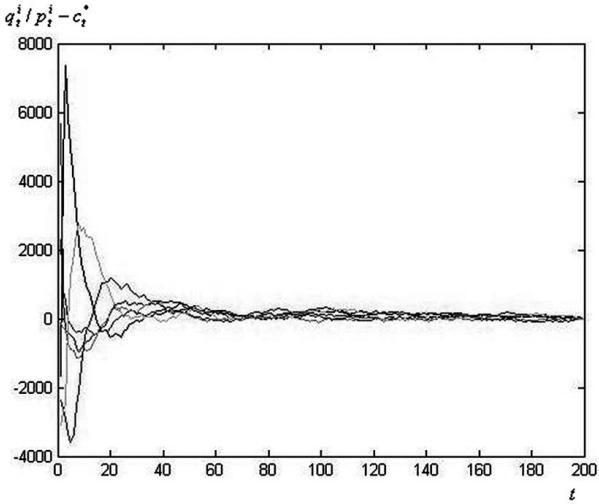


Рис. 5.5. График невязки для стационарного случая.

2) Рассмотрим случай, когда в процессе работы алгоритма в систему поступает новый заказ. Используем постоянный размер шага $\alpha_t = 0.1$. Моделирование этого случая приведено на рис. 5.6.

На рис. 5.7 приведены графики для величины ошибки оценивания с разными постоянными параметрами шага α .

Ошибка оценивания определяется следующим образом: $Err = \sum_i \sqrt{\frac{(x_t^i - x^*)^2}{n}}$.

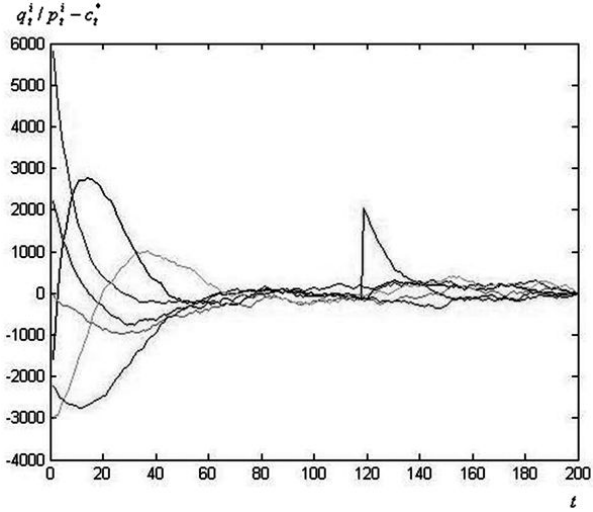


Рис. 5.6. График невязки для случая поступления нового заказа в процессе работы системы.

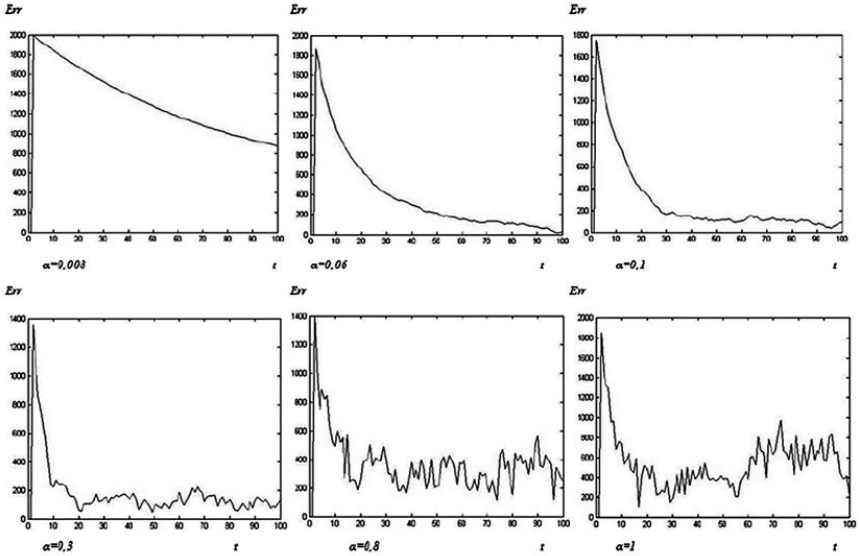


Рис. 5.7. Сравнение разных параметров шага α.

Видно, что чем больше α, тем состояния узлов быстрее сходятся к консенсусу, но с определенного момента поведение системы перестает быть стабильным.

5.8. Выводы

В этой главе была рассмотрена задача достижения консенсуса при неполном обмене информацией в условиях переменной топологии. Описан алгоритм стохастической аппроксимации для достижения среднеквадратического консенсуса. Проведено моделирование алгоритма для задачи балансировки загруженности узлов децентрализованной вычислительной сети при поступлении в каждый узел только зашумленной информации о загруженности и производительности соседей, причем топология сети может меняться со временем, а информация от соседей доходить с задержкой во времени.

5.9. Контрольные вопросы

1) Верно ли, что в задаче достижения консенсуса каждый агент стремится уменьшить отклонение своей целевой переменной от соответствующих переменных своих соседей?

Варианты ответов:

1. Да
2. Нет, каждый агент стремится уменьшить отклонение от среднего арифметического состояний соседей
3. Нет, каждый агент стремится уменьшить значение своей целевой переменной до нуля

2) Какие из задач мультиагентного управления чаще всего сводятся к задачам частичной координатной синхронизации с помощью метода виртуального лидера?

Варианты ответов:

1. Задачи достижения консенсуса
2. Задачи управления формациями
3. Задачи рандеву

3) Что понимается под задачей рандеву?

Варианты ответов:

1. Совпадение или сближение переменных состояния двух или нескольких систем, либо согласованное изменение некоторых количественных характеристик систем
2. Задача по переводу группы агентов в некоторое общее состояние
3. Стремление приближения состояний систем к нулю

4) Какие задачи чаще рассматриваются в системах с огромным числом агентов?

Варианты ответов:

1. Роение
2. Диспетчеризация
3. Рандеву

5) Для чего используется теория графов в описании задачи достижения консенсуса?

Варианты ответов:

1. Для описания внутренних свойств агентов

2. Для описания топологии сети
3. Для построения модели системы

6) Что означает, что узел j является соседом узла i в ориентированном графе?

Варианты ответов:

1. Узел i получает информацию от узла j
2. Узел i может передать информацию узлу j
3. Узел i может передать информацию узлу j, а узел i получает информацию от узла j

7) Что значит, что узлы i и j согласованы?

Варианты ответов:

1. Их состояния совпадают со средним состоянием системы
2. Их состояния совпадают
3. Их состояния равны нулю

8) В каком случае считается, что все узлы системы достигли консенсуса?

Варианты ответов:

1. Когда все состояния равны нулю
2. Когда состояния всех узлов совпадают и равны среднему значению по всех системе
3. Когда все узлы согласованы

9) В чем заключается задача балансировки загрузки сети?

Варианты ответов:

1. Выполнение всех заданий за наименьшее время
2. Планирование поступления заказов в систему
3. Поддержание равномерной загрузки всех узлов сети

10) Какую информацию имеет каждый узел в каждый момент времени для формирования управления?

Варианты ответов:

1. Информацию о собственном состоянии и о состояниях соседей
2. Зашумленные наблюдения о состояниях соседей
3. Зашумленные наблюдения о состояниях соседей и о

собственном состоянии

11) Как называется коэффициент α в протоколе консенсуса?

Варианты ответов:

1. Размер шага протокола управления
2. Шаг дискретизации
3. Весовой коэффициент

12) Чем отличается стационарный случай от нестационарного в задаче балансировки загрузки сети?

Варианты ответов:

1. В стационарном случае информация поступает без помех и задержек
2. В нестационарном случае новые задания могут поступать в систему с течением времени
3. В стационарном случае задания не перераспределяются между узлами

13) Применяются ли алгоритмы стохастический аппроксимации с убывающим размером шага для управления стохастическими системами?

Варианты ответов:

1. Да, применяются
2. Да, но только с неубывающим размером шага
3. Нет, не применяются

14) В каких случаях алгоритмы стохастический аппроксимации с убывающим размером шага показывают плохую работоспособность?

Варианты ответов:

1. В централизованных системах
2. При динамическом перераспределении заданий между агентами
3. При нестационарных условиях

15) Как влияет коэффициент α на скорость сходимости к консенсусу?

Варианты ответов:

1. Чем больше α, тем состояния узлов медленнее сходятся к

консенсусу

2. Коэффициент α не влияет на скорость схождения к консенсусу
3. Чем больше α, тем состояния узлов быстрее сходятся к консенсусу, но с определенного момента поведение системы перестает быть стабильным

Список литературы

1. *Фрадков А.Л.* Кибернетическая физика: принципы и примеры - СПб.: Наука. 2003. 208 с.
2. *Olfati-Saber R., Murray R.M.* Consensus problems in networks of agents with switching topology and time-delays // IEEE Trans. Automatic Control. Sept. 2004. Vol. 49. P. 1520-1533.
3. *Ren W., Beard R.W.* Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications — Communication and Control Engineering Series. — Springer Verlag. New York. 2007. 319 p.
4. *Агаев Р.П., Чеботарев П.Ю. Сходимость* и устойчивость в задачах согласования характеристик (обзор базовых результатов) // Управление большими системами. Спец. выпуск 30.1 "Сетевые модели в управлении". 2010. С. 470-505.
5. *Fax A., Murray R.M.* Information flow and cooperative control of vehicle formations // IEEE Trans. Automat. Contr. Sept. 2004. Vol. 49. P. 1465-1476.
6. *Tanner H.G., Jadbabaie A., Pappas G. J.* Flocking in fixed and switching networks // IEEE Trans. Autom. Contr. 2007. Vol. 52. No. 5. P. 863-868.
7. *Li W.* Stability analysis of swarms with general topology // IEEE Trans. Syst. Man. Cybern. B. 2008. Vol. 38. No. 4. P. 1084-1097.
8. *Fax, R.M. Murray* Information flow and cooperative control of vehicle formations – IEEE Trans. Automat. Contr., vol. 49, pp. 1465-1476, Sept. 2004.

9. *J. Toner, Y. Tu* Flocks, herds, and schools: a quantitative theory of flocking – Phys. Rev. E, vol. 58, no. 4, pp. 4828-4858, Oct. 1998.
10. *J. Cortes, F. Bullo* Coordination and geometric optimization via distributed dynamical systems – SIAM J. Control Optim., May 2003.
11. *F. Paganini, J. Doyle, S. Low* Scalable laws for stable network congestion control – presented at the Int. Conf. Decision and Control, Orlando, FL, Dec. 2001.
12. *Jadbabaie, J. Lin, and A. S. Morse* Coordination of groups of mobile autonomous agents using nearest neighbor rules – IEEE Trans. Automat. Contr., vol. 48, pp. 988-1000, June 2003.
13. *R. Olfati-Saber and R. M. Murray* Consensus problems in networks of agents with switching topology and time-delays – IEEE Trans. Automatic Control, vol. 49, pp. 1520-1533, Sep., 2004.
14. *W. Ren and R. W. Beard* Consensus seeking in multiagent systems under dynamically changing interaction topologies – IEEE Trans. Automat. Control, vol. 50, no. 5, pp. 655-661, 2005.
15. *T. C. Aysal and K. E. Barner* Convergence of consensus models with stochastic disturbances – IEEE Trans. Info. Theory, vol. 56, no. 8, pp. 4101-4113, 2010.
16. *L. Xiao, S. Boyd, and S.-J. Kim* Distributed average consensus with least-mean-square deviation – Journal of Parallel and Distributed Computing, vol. 67, pp. 33-46, 2007.
17. *D. Schizas, A. Ribeiro, and G. B. Giannakis* Consensus in ad hoc WSNs with noisy links-part I: distributed estimation of deterministic signals. IEEE Trans. Signal Process., vol. 56, pp. 350-364, Jan. 2008.
18. *F. Cucker and E. Mordecki* Flocking in noisy environments – J. Math. Pures Appl., vol. 89, no. 3, pp. 278-296, 2008.
19. *Kashyap, T. Basar, and R. Srikant* Quantized consensus – Automatica, vol. 43, pp. 1192-1203, 2007
20. *M. Huang* Stochastic Approximation for Consensus with General Time-Varying Weight Matrices – Proc. the 49th IEEE CDC, Atlanta, GA, USA, pp. 7449-7454, Dec. 2010.
21. *N. Tsitsiklis, D. P. Bertsekas, and M. Athans* Distributed asynchronous deterministic and stochastic gradient optimization

algorithms – IEEE Trans. Autom. Contr., vol. 31, no. 9, pp. 803-812, 1986.
22. *M. Huang and J. H. Manton* Coordination and consensus of networked agents with noisy measurements: stochastic algorithms and asymptotic behavior – SIAM J. Control Optim., vol. 48, no. 1, pp. 134-161, 2009.
23. *S. Kar and J. M. F. Moura* Distributed consensus algorithms in sensor networks with imperfect communication: link failures and channel noise – IEEE Trans. Sig. Process., vol. 57, no. 1, pp. 355-369, 2009.
24. *T. Li and J.-F. Zhang* Mean square average-consensus under measurement noises and fixed topologies – Automatica, vol. 45, no. 8, pp. 1929-1936, 2009.
25. *B. Gharesifard and J. Cortes* Distributed strategies for generating weight-balanced and doubly stochastic digraphs – SIAM J. Contr. Optim., Nov. 2010.
26. *Granichin O., Vakhitov A., and Vlasov V.* Adaptive control of SISO plant with time-varying coefficients based on random test perturbation – In Proc. of the 2010 American Control Conference, June 30-July 02, 2010, Baltimore, MD, USA. P. 4004-4009.
27. *Вахитов А.Т., Граничин О.Н., Гуревич Л.С.* Алгоритм стохастической аппроксимации с пробным возмущением на входе в нестационарной задаче оптимизации // Автоматика и телемеханика, 2009, № 11. С. 70-79.
28. *Granichin O., Gurevich L., and Vakhitov A.* Discrete-time minimum tracking based on stochastic approximation algorithm with randomized differences – Proc. of the Combined 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference, December 16-18, 2009, Shanghai, P.R. China. P. 5763-5767.
29. *Borkar V. S.* Stochastic Approximation. A Dynamical Systems Viewpoint – Cambridge University Press; 2008
30. *Граничин О.Н.* Стохастическая оптимизация и системное программирование // Стохастическая оптимизация в информатике. 2010. 6, с. 3-44

31. *Вахитов А.Т., Граничин О.Н., Паньшенсков М.А.* Методы оценивания скорости передачи данных в грид // Нейрокомпьютеры: разработка, применение, 2009, № 11, с. 45 - 52.
32. *Фрадков А.Л.* Схема скоростного градиента и его примнения в задачах адаптивного управления // Автоматика и телемеханика, 1979, № 9, с. 90 - 101.
33. *Granichin O., Skobelev P., Lada A., Mayorov I., Tsarev A.* Comparing adaptive and non-adaptive models of cargo transportation in multi-agent system for real time truck scheduling // In: Proc. of the 4th International Conference on Computational Intelligence. P. 282-285 (Evolutionary Computation Theory and Applications, ECTA'2012, October 5-7, 2012, Barcelona, Spain).
34. *Амелина Н., Лада А., Майоров И., Скобелев П., Царев А.* Исследование моделей организации грузовых перевозок с применением мультиагентной системы для адаптивного планирования мобильных ресурсов в реальном времени // Проблемы управления, 2011, № 6, с. 31-37.
35. *Амелина Н.О.* Диспетчеризация сети с переменной топологией при помехах и задержках в измерениях // Вестник СПбГУ. Сер. 1: Математика. Механика. Астрономия, 2012, № 2, с. 11-15.
36. *Амелина Н.О., Васильев В.И., Граничин О.Н., Кияев В.И.* Перспективный децентрализованный подход к балансировке загрузки в многоядерных процессорах // В сб.: Труды Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: поиск новых решений», г. Новороссийск, 17-22 сентября 2012 г., М: Изд-во Моск. ун-та. С. 345-347.
37. *Амелина Н.О.* Балансировка загрузки узлов децентрализованной вычислительной сети при неполной информации // Нейрокомпьютеры: разработка, применение, № 6, 2011, С. 56-63.
38. *Амелин К.С., Амелина Н.О., Граничин О.Н., Корявко А.В.* Применение алгоритма локального голосования для достижения консенсуса в децентрализованной сети интеллектуальных агентов // Нейрокомпьютеры: Разработка, Применение, 2012, № 11, С. 39-47.

39. *Amelin K., Amelina N., Granichin O., Granichina O.* Multi-agent stochastic systems with switched topology and noise // In: Proc. of 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, SNPD 2012. P. 438-443 (August 8-10, Kyoto, Japan, 2012).
40. *Амелина Н.О., Фрадков А.Л.* Приближенный консенсус в стохастической динамической сети с неполной информацией и задержками в измерениях // Автоматика и Телемеханика, 2012, № 11, с. 6-30.
41. *N. Amelina, A. Fradkov, and K. Amein* Approximate Consensus in Multi-agent Stochastic Systems with Switched Topology and Noise // Proc. of MSC IEEE 2012, October 3-5, 2012, Dubrovnik, Croatia, pp. 445-450.

6. Создание объектов интеллектуальной собственности и авторское право

Особенности авторского права и патентования разработок встроенных систем.

6.1. Введение

Содержание предыдущих глав убедительно показывает, что создание мультиагентных систем является достаточно сложным делом и связано с разработкой соответствующего алгоритмического и программного обеспечения. Разнообразные МАС базируются на некоторой общей фундаментальной основе, включающей теоретические разработки и методы. Однако каждая МАС предназначена для решения конкретных задач – будь то система для управления командами роботов играющих в футбол или система обеспечения безопасности автомобиля на дороге. Отсюда следует, что такие МАС являются *уникальными объектами*, созданным разработчиком или коллективом разработчиков для достижения совершенно определенной цели. В связи с этим возникает ряд вопросов об авторстве разработки программного продукта и о правах на него. Разработан новый прикладной программный продукт или распределенная информационная система. Кто его автор и кто владелец?

И если с вопросом – кто владелец? – особых проблем нет (владельцем продукта может быть компания, заказавшая программный продукт разработчику, или пользователь, купивший его в магазине), то с вопросом об авторстве ответ не так прост, как может показаться на первый взгляд!

Особенностью программной индустрии является то, что разработка сложного программного обеспечения, безусловно, является творческим процессом, в котором участвует достаточно большое количество людей, и сам продукт не материален. Именно поэтому в большинстве случаев трудно определить, кому принадлежит авторство законченного продукта. Кроме того, программное обеспечение

с конца XX века стало продуктом массового "потребления", его легко и быстро можно скопировать, появились специализированные обменные сайты (торренты), с которых можно бесплатно скачать практически любое ПО – и поэтому убытки компаний-разработчиков от его пиратского использования и контрафактной продажи исчисляются миллиардами долларов!

В прошлом году организация Business Software Alliance выпустила отчет, в котором подробно описывались противоречивые мнения пользователей ПК из разных стран. Согласно этому опросу, около70% компьютерных пользователей в мире поддерживают права на интеллектуальную собственность, но при этом 47% из них всегда или практически всегда приобретают нелегальное ПО.

В ходе создания программного продукта реализуется его жизненный цикл, в котором участвует много разных специалистов – менеджеры высшего звена компании-разработчика, руководители проектов, архитекторы, спецификаторы, кодировщики, тестеры, интеграторы, технические писатели, внедренцы, эксплуатационники и т.д. Так кто же автор? Без участия любого из них продукт не будет реализован до конца. Конечно, требования к функциональности, архитектура продукта, алгоритмы, модульная и функциональная структура, тестовые покрытия, сценарии использования, системы автоматизации проектирования и тестирования создаются конкретными людьми и, безусловно, именно они являются авторами этих артефактов.

Естественно, автор может в опосредованном виде заявить и получить права (приоритет) на такой "подпродукт" в виде статьи в научном журнале или доклада на конференции, опубликованной монографии или учебного пособия, однако каждый из этих артефактов не является ни материальным, ни виртуальным целостным

программным продуктом, который в конечном счете может стать *объектом интеллектуальной собственности (ИС)*.

В большинстве случаев создание *инновационного продукта* в сфере информационных технологий практически всегда связано с появлением такого объекта. Здесь уместно привести слова заместителя генерального юрисконсульта подразделения корпорации Майкрософт по борьбе с глобальным пиратством и контрафактом Дэвида Финна, который считает, что "защита интеллектуальной собственности является опорой инновационной деятельности, так как эффективная защита стимулирует такую деятельность. Когда развитию идеи или продукта человек или компания уделяют много времени и сил, и все это ради того, чтобы однажды эти идеи украли, – цикл исследований и разработки нарушается, интерес к новому продукту падает, а инвестиции в новые идеи сокращаются. Пиратство душит инновации!" (http://brainon.ru/news/2398).

Именно поэтому необходимо знать некоторые важные положения, связанные с созданием объекта интеллектуальной собственности и формированием *авторских прав* на него, т.е. возникновением и формированием *объекта права интеллектуальной собственности*. Такая формулировка стала общеупотребительной с вступлением в силу 4-й Части Гражданского кодекса РФ, ниже мы будем использовать эту формулировку.

6.2. Общие принципы и основные понятия в сфере интеллектуальной собственности

Вопрос об авторском праве и о наследовании авторских прав является одним из самых долгоживущих в истории развития человечества. В Древней Греции, например, рукописи пьес античных авторов сдавались в специальные хранилища, чтобы всегда можно было установить подлинность текста. И хотя вопрос об интеллектуализации объекта собственности очень тесно связан с авторским

правом, понимание актуальности этого вопроса возникло на много веков позже.

Первым в истории законом об авторском праве и – косвенно – об интеллектуальной собственности стал принятый в Англии в 1710 году "Статут королевы Анны", закрепивший личное право на охрану опубликованного произведения. Прообразом современных патентных законов стал также принятый в Англии "Статут Якова I" ("Статут о монополиях") 1624 года. Данным статутом было установлено важное правило: королевская власть не может выдавать никаких патентов, кроме патентов на изобретения.

Международное право интеллектуальной собственности начинает складываться с конца XIX века. Были приняты акты, регулирующие право интеллектуальной собственности в отношениях между разными государствами (Парижская, Бернская, Женевская конвенции).

В России право интеллектуальной собственности стало складываться позже, чем в других странах. В 1911 году было принято "Положение об авторском праве", которое регламентировало права авторов произведений на основе лучших образцов западноевропейских законодательств того времени. В 1917 году был принят Декрет ЦИК "О государственном издательстве", которым вводилось разрешение объявлять государственную монополию сроком не более чем на 5 лет на сочинения, подлежащие изданию. После присоединения СССР (России) к международным актам (1965 г., 1973 г., 1995 г.) право интеллектуальной собственности стало соответствовать нормам международного права.

26 апреля 2012 года Всемирная организация по интеллектуальной собственности (World Intellectual Property Organization – WIPO)

в 12-й раз отметила Международный день интеллектуальной собственности. Решение об учреждении этого праздника приняла Генеральная ассамблея WIPO в октябре 1999 года. Этот международный день подчеркивает значение инноваций в повседневной жизни человека и развитии общества. Более того, в настоящее время проводится регулярная международная Олимпиада для школьников по интеллектуальной собственности. Значение этого мероприятия трудно переоценить!

Изобретения, патенты, товарные знаки, авторское право и смежные права, как результаты творческой деятельности и формирования знаний, являются мощными инструментами содействия экономическому и культурному развитию всего мирового сообщества.

Право интеллектуальной собственности – совокупность правовых норм и институтов гражданского права, регулирующих отношения в сфере возникновения, использования и защиты объектов интеллектуальной собственности [1].

Чтобы правильно определить, что является *объектом интеллектуальной собственности*, следует прежде всего обозначить, что такое *интеллектуальная деятельность*. Можно выделить следующие сложившиеся признаки интеллектуальной деятельности.

- Интеллектуальная деятельность носит идеальный характер. Результат интеллектуальной деятельности продуцируется сознанием человека путем логического построения мысли и отражает новизну мысли.
- Результатом интеллектуальной деятельности является выраженный в объективной форме ее продукт, именуемый в зависимости от его характера произведением науки, литературы, изо-

бразительного или иного вида искусства, изобретением, товарным знаком или промышленным образцом.

- Результаты интеллектуальной деятельности в отличие от объектов вещных прав имеют не материальную природу. Литературные, художественные, кинематографические, театральные, телевизионные произведения представляют собой систему литературных либо художественных образов, закрепленных в той или иной форме на некотором носителе (бумаге, холсте, пленке, компакт-диске и т.д.) То, в чем закреплены результаты интеллектуальной, сами по себе не являются результатами интеллектуальной деятельности.
- Подлежит защите не только форма выражения интеллектуальной деятельности (книга, картина, фильм, музыкальное произведение), но и ее содержание (основная мысль произведения). Результаты интеллектуальной деятельности не подвержены износу, амортизации. Они могут устаревать лишь морально.
- Продуктом интеллектуальной деятельности могут быть средства "индивидуализации" юридического лица или индивидуального предпринимателя, а также "индивидуализации" выполняемых работ или услуг (фирменные наименования, товарные знаки, знаки обслуживания и наименования мест происхождения товаров).

Из этого следует, что *интеллектуальная собственность* – совокупность *исключительных прав гражданина или юридического лица* на результаты творческой, интеллектуальной деятельности, а также приравненные к ним по правовому режиму средства индивидуализации юридических лиц, продукции, работ и услуг (фирменное наименование, товарный знак, знак вида или места обслуживания и т. п.).

Процесс эффективной интеллектуальной деятельности, завершается созданием новых, творчески самостоятельных результатов в области науки, техники, технологий, литературы и искусства. Право интеллектуальной собственности не регулирует сам процесс. Эта область гражданского права *охраняет результаты интеллектуальной деятельности*, которые представляют собой нематериальные продукты и блага.

Традиционно интеллектуальную собственность делят на две составляющие: промышленную собственность и авторское право. *Промышленная собственность* характеризуется такими ее составляющими, как изобретения, полезные модели, промышленные образцы товарные знаки, знаки обслуживания и фирменные наименования. *Авторское право* относится к произведениям искусства, литературным и музыкальным произведениям, творениям кинематографии, театра и телевидения, а также к научным открытиям и компьютерным разработкам. В области высоких технологий (High Technologies) промышленная собственность и авторское право часто тесно взаимодействуют и могут образовывать единое целое. Например, топология интегральной микросхемы относится к сфере авторского права, а реализованный на этой микросхеме чип является уже промышленным образцом (промышленной собственностью).

Исключительное право гражданина или юридического лица – это имущественное право на объекты результатов интеллектуальной деятельности. С 1 января 2008 года вступила в силу Часть 4 Гражданского кодекса РФ, которая регулирует вопросы интеллектуальной деятельности. Право интеллектуальной собственности является незыблемым. Никто не может быть лишен права интеллектуальной собственности или ограничен в его осуществлении, если оно законно принадлежит создателю или владельцу ИС.

Использование исключительного права в гражданском обороте возможно в двух формах:

а) заключение договора об окончательном отчуждении исключительного права на объекты интеллектуальной собственности:

б) заключение лицензионного договора, то есть на ограниченное во времени и по территории использование объектов интеллектуальной собственности.

Согласно нормативным документам Всемирной организации по интеллектуальной собственности (ВОИС), к *объектам права интеллектуальной собственности*, в частности, принадлежат [2]:

- литературные и художественные произведения;
- научные открытия и разработки;
- компьютерные программы, компиляции данных (базы данных), компоновка (топология) интегральных микросхем, доменные имена;

- рационализаторские предложения; изобретения, полезные модели, промышленные образцы;
- фонограммы, видеограммы, передачи (программы) организаций теле и радио вещания;
- исполнение музыкальных произведений и театральные постановки;
- результаты селекционной деятельности в биологии;
- коммерческие (фирменные) наименование, торговые марки (знаки для товаров и услуг), географические указания;
- коммерческие тайны.

Позднее в сферу деятельности ВОИС были включены исключительные права, относящиеся к географическим указаниям, новым сортам растений и породам животных и т.д.

Здесь следует отметить тонкое различие в правах – право интеллектуальной собственности (право на нематериальный продукт) и право собственности на вещь могут не зависеть одно от одного. Переход права на объект интеллектуальной собственности не означает перехода права собственности на вещь. И обратно, переход права собственности на вещь не означает перехода права на объект интеллектуальной собственности. Например, пользователь компакт-диска с программным продуктом имеет полное право на использование этого продукта в своих целях, однако он не может тиражировать и продавать этот программный продукт. Разработчик программного продукта имеет эти права, но он не может распоряжаться компакт-диском пользователя как вещью, потому что этот конкретный диск пользователя не принадлежит разработчику.

6.3. Законодательство в области права интеллектуальной собственности

Происхождение самого понятия *интеллектуальная собственность* часто связывают с французским законодательством конца XVIII века. Традиция проприетарного (основанного на частной собственности и на идее коммерческого использования авторских и иных прав) подхода к авторским и патентным правам не только ро-

дилась во Франции, но и опиралась на теорию естественного права, получившая наиболее последовательное развитие в работах французских философов-просветителей: Вольтера, Дидро, Гольбаха, Гельвеция и Руссо.

Во вводной части французского патентного закона 1791 года говорилось, что "всякие новые идеи, провозглашение и осуществление которых может быть полезным для общества, принадлежат тем, кто их создал, и было бы ограничением прав человека не видеть новое промышленное изобретение как собственность его творца" (http://www.nordreview.ru/2012/03/12/intellectual-property/).

В 1883 году была принята Парижская Конвенция по охране промышленной собственности, в частности, изобретений, промышленных образцов и товарных знаков, имеющая общепризнанный международный характер. Основана Конвенция была одиннадцатью странами. В настоящее время в нее входят более 130 стран.

Очередным шагом в охране интеллектуальной собственности стала Бернская Конвенция (Швейцария 1886 год) по защите литературных и художественных произведений, основанная десятью странами, включающая на сегодняшний день более 110 стран.

Следующим документом по охране интеллектуальной собственности явилось Мадридское соглашение «О международной регистрации знаков» (1891 год), к которому присоединились более 40 стран.

В начале XX века Й. Шумпетер выдвинул гипотезу о том, что двигателем экономического развития является инновационная деятельность, в основе которой лежат новшества, придуманные человеком – новации [3]. В дальнейшем

совокупность новаций стали именовать "интеллектуальным капиталом". В работах американских экономистов Патрика Салливана [4,5] и К.Е. Свейби [6] понятие интеллектуальный капитал определяется как "знание, которое может быть превращено в капитал", а "интеллектуальные активы" как "закодированные знания сотрудников предприятия, которые могут приносить прибыль". Стало понятно, что использования и защиты интеллектуального капитала нужны строгие юридические нормы и законы, регулирующие деятельность в этой области. И такие законы с начала XX века и соглашения стали появляться во многих промышленно-развитых странах мира. Можно упомянуть, например, Гаагское соглашение о международном депонировании промышленных образцов 1934 года, Лиссабонское соглашение об охране наименований мест происхождения и их международной регистрации, заключенное в 1958 году.

Дальнейшим развитием международного законодательства в этой сфере стала Конвенция, учредившая в 1967 году в Стокгольме "Всемирную организацию интеллектуальной собственности" (ВОИС). СССР принимал участие в ее создании, и одним из официальных языков Конвенции, наряду с английским, испанским, французским стал и русский. Конвенция вступила в силу в 1970 году.

ВОИС решает следующие вопросы:

- улучшение интегральной охраны интеллектуальной собственности в мире, гармонизации национальных законодательств в области ее экономической и правовой охраны;
- заключение международных договоров по охране интеллектуальной собственности;
- выполнение административных функций Парижского союза, специальных Союзов, образованных в связи с этим Союзом, и Бернского Союза;
- оказание технико-юридической помощи в области охраны интеллектуальной собственности;
- сбор и распространение информации об интеллектуальной собственности, проведения исследований в этой сфере и организации публикации их результатов;

- обеспечене деятельности служб, облегчающих международную охрану интеллектуальной собственности гармонизацию и интеграцию законодательства в этой сфере;

Наиболее значимым проектом ВОИС является Договор о патентной кооперации (PatentCooperationTreaty – PCT), к которому присоединились уже 139 стран. С 1974 года ВОИС является специализированным учреждением ООН по вопросам творчества и интеллектуальной собственности. При ВОИС действует центр по арбитражу и посредничеству. С 1999 года ВОИС предоставляет услуги по урегулированию споров, которые возникают при регистрации и использовании наиболее распространенных типичных названий доменов в Интернете (.com, .net, .org).

Отметим, что *сам объект права интеллектуальной собственности* может возникнуть только с момента, когда "интеллектуальный замысел" получил внешнее выражение в той или иной объективной форме. При этом обязательным основанием для предоставления авторско-правовой охраны является выражение произведения в некотором "вещественном" виде или форме. При этом не имеет значения, зафиксировано ли оно на информационном носителе или просто было оглашено «в месте, где присутствует значительное число лиц, не принадлежащих к обычному кругу семьи" [7]. Пока замысел автора не проявился вовне, нет объекта защиты, требуется любое выражение идеи, образа, мысли, доступное для восприятия других людей.

Пункт 1 статьи 2 Бернской конвенции об охране литературных и художественных произведений предусматривает, что во всех странах-участницах должна обеспечиваться охрана всем "произведениям в области литературы, науки и искусства, каким бы способом и в какой бы форме они ни были выражены". Однако пункт 2 статьи 2 Бернской конвенции предоставляет странам-участницам возможность "предписать, что литературные или художественные произведения или какие-либо определенные их категории не подлежат охране, если они не закреплены в той или иной материальной форме"[7].

Аналогично, законодательство Великобритании об авторском праве устанавливает, что необходима "письменная или иная материальная форма". В соответствии с параграфом 106 раздела 17 Сво-

да законов США для того, чтобы получить охрану по статутному праву, произведение должно быть "зафиксировано в любой осязаемой форме… обеспечивающей его восприятие, воспроизведение или сообщение его другим лицам как непосредственно, так и с помощью какого-либо механизма или устройства". Закон США об авторском праве 1976 года защищает "все виды письменного, зрительного или звукового материала, которые зафиксированы в ощутимом объективном способе выражения". В настоящее время охраняется любой результат, характеризуемый наличием творчества и оригинальности, независимо от того, был ли материал официально опубликован или выложен в Интернет.

Конкретная форма *объективного выражения* не имеет особого значения для решения принципиального вопроса о возможности предоставления правовой охраны. Всякий документ, опубликованный на бумаге, в аудио или видео форме, в принципе, может быть представлен в цифровом виде в двоичной системе счисления. Электронный документ, в таком случве, может быть определен как набор данных, записанных в машиночитаемом виде, для которых существует процедура, позволяющая однозначно преобразовать эти данные в документ традиционного режима [8].

Любая программа для ЭВМ не является "вещественной", она представляет собой определенную последовательность идентифицируемых числовых объектов на материальном носителе: на магнитном носителе — последовательность переходов изменения магнитного поля, на лазерном диске — последовательность участков, отражающих и не отражающих лазерный луч, и т.д. [9]. Именно поэтому компьютерная программа относится к сфере авторского права, а не к промышленной собственности.

Отметим, что произведения науки, литературы и искусства охраняются авторским правом независимо не только от способа их выражения, но также от их назначения и достоинства. Такой подход позволяет сохранить наиболее ценные достижения человеческого интеллекта, предоставив времени производить окончательный отбор.

Российская Федерация, как правопреемник СССР, является членом ВОИС – организации, оказывающей содействие экономиче-

ской и правовой охране интеллектуальной собственности во всем мире путем сотрудничества с Парижским и Бернским союзами по охране интеллектуальной собственности.

Систему источников права интеллектуальной собственности в России составляют [1]:

1. Конституция РФ – ст. 44 закрепляет, что каждому гарантируется свобода литературного, художественного, научного, технического и других видов творчества, преподавания. Право на свободу во всех сферах творческой деятельности – неотъемлемое право человека, гарантированное общепризнанными нормами международного права. Право человека заниматься творческой деятельностью может осуществляться как на профессиональной, так и на любительской основе. И тот и другой творческий работник *равноправны в области авторского права и смежных прав*, права на интеллектуальную собственность, охрану секретов мастерства, свободу распоряжения результатами своего труда, поддержку государства.; пункт "о" ст. 71 относит правовое регулирование интеллектуальной собственности к федеральному ведению;

2. Гражданский кодекс РФ – положения об интеллектуальной собственности содержатся в статьях:

- ст. 8 ГК РФ относит создание результатов интеллектуальной деятельности к основаниям возникновения гражданских прав и обязанностей;
- ст. 128 ГК РФ – результаты интеллектуальной деятельности, в том числе исключительные права на них (интеллектуальная собственность), отнесены к объектам гражданских прав;
- ст. 138 ГК РФ раскрывает содержание понятия исключительного права (интеллектуальной собственности), включающее результаты интеллектуальной деятельности и приравненные к ним средства индивидуализации).

Новшеством российского законодательства явилось принятие четвертой части ГК РФ, объединяющей в себе все отношения в области интеллектуальной собственности. Положения новой части ГК РФ вступили в силу с 1января 2008 г., соответственно с этого времени прекратили свое действие законы, регулирующие отдельные вопросы интеллектуальной собственности.

3. Федеральные законы:

а). Закон РФ от 9 июля 1993 г. № 5351I "Об авторском праве и смежных правах". В нем закрепляются отношения, возникающие в связи с созданием и использованием произведений науки, литературы и искусства (авторское право), фонограмм исполнений, постановок, передач организаций эфирного или кабельного вещания (смежные права). Устанавливаются сфера действия, объекты, сроки действия авторского права.

б). Патентный закон Российской Федерации от 23 сентября 1992 г. № 35171. Закон регулирует отношения, возникающие в связи с правовой охраной и использованием изобретений, полезных моделей и промышленных образцов. Установлены условия патентоспособности изобретений, полезных моделей и промышленных образцов. Определен круг субъектов прав на изобретения, полезные модели и промышленные образцы.

в). Закон РФ от 23 сентября 1992 г. № 3520 "О товарных знаках, знаках обслуживания и наименованиях мест происхождения товаров". Законом регулируются отношения, возникающие в связи с регистрацией, правовой охраной и использованием товарных знаков, знаков обслуживания и наименований мест происхождения товаров. В Законе даются определения товарного знака и знака обслуживания, наименования места происхождения товара.

г). Закон РФ от 23 сентября 1992 г. № 3523I "О правовой охране программ для электронных вычислительных машин и баз данных". Регулируются отношения, возникающие в связи с правовой охраной и использованием программ для ЭВМ, баз данных.

д). Закон РФ от 23 сентября 1992 г. № 35261 "О правовой охране топологий интегральных микросхем". Регулируются отношения, возникающие в связи с созданием, правовой охраной, а также использованием оригинальной топологии интегральной микросхемы, созданной в результате творческой деятельности автора и являющейся неизвестной автору и (или) специалистам в области разработки топологий на дату ее создания.

е). Закон РФ от 6 августа 1993 г. № 56051 "О селекционных достижениях". Устанавливает основы правового регулирования имуще-

ственных, а также связанных с ними личных неимущественных отношений, возникающих в связи с созданием, правовой охраной и использованием селекционных достижений.

ж). Федеральный закон от 29 июля 2004 г. № 98ФЗ "О коммерческой тайне".

з). Федеральный закон от 26 июля 2006 г. № 135ФЗ "О защите конкуренции".

и). Федеральный закон от 27 июля 2006 г. № 149ФЗ "Об информации, информационных технологиях и о защите информации".

4. Указы Президента РФ:

- О государственной политике в области охраны авторского права и смежных прав;
- О мерах по реализации прав авторов произведений, исполнителей и производителей фонограмм на вознаграждение за воспроизведение в личных целях аудиовизуального произведения или звукозаписи произведения;
- О правовой защите результатов научно-исследовательских, опытно-конструкторских и технологических работ военного, специального и двойного назначения;
- О государственной политике по вовлечению в хозяйственный оборот результатов научно-технической деятельности и объектов интеллектуальной собственности в сфере науки и технологий.

5. Постановления Правительства РФ:

- Постановления, о минимальных ставках авторского вознаграждения за отдельные виды использования объектов авторского права и смежных прав;
- Положение о патентных поверенных;
- Положение о пошлинах за патентование изобретений, полезных моделей, промышленных образцов, регистрацию товарных знаков, знаков обслуживания, наименований мест происхождения товаров, предоставление права пользования наименованиями мест происхождения товаров.

6. Международные договоры и соглашения:

- Парижская конвенция по охране промышленной собственности 1883 г. (для СССР и затем для России Конвенция вступила в силу с 1июля 1965 г.);
- Всемирная (Женевская) конвенция об авторском праве 1952 г. (вступила в силу в 1968 г.);
- Бернская конвенция об охране литературных и художественных произведений 1886 г. (действует на территории РФ с 1995 г.);
- Договор о патентной кооперации 1970 г. (вступил в силу для СССР в 1978 г.);
- Соглашение о сотрудничестве в области охраны авторского права и смежных прав 1993 г. (Россия совместно с Азербайджаном, Арменией, Беларусью, Казахстаном, Киргизией, Молдовой, Таджикистаном, Туркменией, Узбекистаном и Украиной; вступило в силу в том же году);
- Евразийская патентная конвенция 1994 г. (страны бывшего СССР, вступил в силу в 1995 г.);
- Конвенция, учреждающая Всемирную организацию интеллектуальной собственности 1967 г. (вступил в силу для СССР в 1973 г.);
- Конвенция об охране интересов производителей фонограмм 1971 г. (действует на территории РФ с 1995 г.);
- Международная конвенция по охране новых сортов растений 1961 г. (Россия присоединилась в 1997 г.);
- Международная (Римская) конвенция об охране прав исполнителей, изготовителей фонограмм и вещательных организаций 1961 г. (вступила в силу в 2003 г.).

Во всех случаях только объективно выраженный результат интеллектуальной деятельности может участвовать в экономическом обороте, представлять собой специфический товар – интеллектуальную собственность. Современная тенденция такова, что результаты интеллектуальной деятельности все более отчетливо приобретают черты товара, который можно выгодно продать. И все чаще объекты ИС, особенно в сфере высоких технологий, продукты интеллектуального труда изначально создаются именно для успешного функционирования на рынке.

6.4. Юридическая природа, содержание и защита исключительных прав

Исключительные права на интеллектуальную собственность – группа прав, отличная от права собственности на материальные объекты, выполняющая в отношении нематериальных объектов функции, аналогичные функциям права собственности для материальных объектов [1].

Исключительными признаются права, относящиеся к:

- литературным, художественным и научным произведениям (авторские права);
- исполнительской деятельности артистов, звукозаписям, радио и телевизионным передачам (смежные права);
- изобретениям, полезным моделям и промышленным образцам (патентные права);
- селекционным достижениям;
- топологиям интегральных микросхем;
- товарным знакам, знакам обслуживания,
- фирменным наименованиям,
- коммерческим обозначениям и наименованиям мест происхождения товаров (права на средства индивидуализации юридического лица, товаров, работ и услуг),
- секреты производства (ноу-хау).

Исключительные права на результаты интеллектуальной деятельности и средства индивидуализации обладают общими свойствами:

а) они возникают только при наличии прямого указания закона. Каждый новый охраняемый вид объектов исключительных прав должен быть назван в законе;

б) они представляют собой особую ветвь абсолютных прав. Для исключительного права характерно то, что оно возникает у правообладателя независимо от воли третьих лиц и что такому праву корреспондирует обязанность всех окружающих воздерживаться от действий, способных нарушить это право. Правообладатель может самостоятельно осуществлять использование такого объекта тем

или иным способом, а также разрешить другому лицу использовать соответствующий объект.

Исключительное право по своему содержанию является *имущественным правом*. Оно передаваемо, отчуждаемо, легко обособляется от личности автора или иного правообладателя. Авторы (создатели) творческих результатов также имеют личные неимущественные права (право авторства, право на авторское имя), которым присущи неотчуждаемость и непередаваемость (ст.150 ГКР Ф).

Личные неимущественные права интеллектуальной собственности на объект, созданный по заказу, принадлежат создателю этого объекта. В случаях, предусмотренных законом, отдельные личные неимущественные права интеллектуальной собственности на такой объект могут принадлежать заказчику. Имущественные права интеллектуальной собственности на объект, созданный по заказу, принадлежат творцу этого объекта и заказчику совместно, если другое не установлено договором, или заказчику, если это предусмотрено условиями контракта.

Исключительные права могут переходить по наследству. Специфика заключается в том, что исключительные права переходят по наследству только на определенный срок, а по его окончании результаты интеллектуальной деятельности и средства индивидуализации становятся общественным достоянием.

Защита исключительных прав осуществляется общими способами. Особенности защиты исключительных прав устанавливаются специальными законами (например, в Законе о товарных знаках назван такой способ защиты, как удаление за счет нарушителя с контрафактных товаров, этикеток, упаковок незаконно используемого товарного знака и др.).

В табл. 6.1 перечислены виды охраняемых объектов, срок охраны и возможность продления этого срока.

Таблица 6.1.

Перечень охраняемых объектов

		Срок охраны	Продление
Объекты авторских и смежных прав	Произведения науки, литературы, искусства	Жизнь автора +70 лет	Нет
	Программы для ЭВМ	Жизнь автора +70 лет	Нет
	Базы данных (как объект смежного права)	15 лет	Да
	Исполнения	Жизнь исполнителя, но не >50 лет	Нет
	Фонограммы	50 лет	Нет
	Сообщения в эфир или по кабелю радио- и телепередач	50 лет	Нет
Промышленная ИС	Изобретения	20 лет	Нет
	Полезные модели	10 лет	Да, до 3 лет
	Промышленные образцы	15 лет	Да, до 10 лет
	Селекционные достижения	30 (35) лет	Нет
	Топологии интегральных микросхем	10 лет	Нет
	Секреты производства (ноу-хау)	Пока сохраняется в тайне	Нет
Средства индивидуа-лизации	Фирменные наименования	Неограничен	
	Товарные знаки и знаки обслуживания	10 лет	Да
	Наименования мест происхождения товаров	10 лет	Да
	Коммерческие обозначения	Пока сохраняется известность	Нет

Понятие *авторское право* понимается в объективном и субъективном смысле. В объективном смысле авторское право – это совокупность правовых норм, регулирующих отношения, возникающие в связи с созданием и использованием произведений литературы, науки и искусства (выступает в качестве подотрасли гражданского права). В субъективном смысле авторское право – совокупность субъективных прав, возникающих у автора в связи с созданием конкретного произведения литературы, науки и искусства. Термин "авторское право" понимается не только как право автора (создателя) произведения, но и как право законного обладателя авторского права, которое может быть передано ему от автора.

Основной задачей авторского права является, с одной стороны, обеспечение интересов авторов и их правопреемников, а с другой стороны – интересы общества в целом.

Принципы авторского права:

а) принцип свободы творчества – означает, что каждый свободен в выборе сферы творческой деятельности, способа ее осуществления, резюмируется равенство в осуществлении своих прав автора;

б) принцип сочетания личных интересов автора с интересами общества – заключается в том, что наряду с исключительным правом автора использовать свое произведение (воспроизводить любым способом, передавать права третьим лицам и т. д.) законом предусмотрены случаи, когда обеспечивается общественный интерес (случаи использования произведения без согласия автора и без выплаты ему авторского вознаграждения).

Общество заинтересовано не только в эффективной защите авторского права, но и в свободном доступе к авторским произведениям;

в) принцип неотчуждаемости личных неимущественных прав автора – означает, что ни при каких обстоятельствах права, относящиеся к личным неимущественным, не подлежат какой-либо передаче третьим лицам в силу их неотчуждаемой природы. Личное неимущественное право автора неотъемлемо от личности, неразрывно связано с личностью автора и не передается третьим лицам даже в случае волеизъявления автора на такую передачу. В целях эффективного использования произведения и защиты авторского права из этого принципа есть изъятия (право обнародовать произведение, право на защиту чести, достоинства и деловой репутации автора);

г) принцип свободы авторского договора – данный принцип означает, что автор свободен в выборе контрагентов, условий авторского договора, а также предусматривает недействительность договора при отсутствии добровольного волеизъявления автора заключить договор.

Выше было показано, что в настоящее время результаты интеллектуальной деятельности отчетливо приобретают черты товара, свободно обращающегося на рынке. Из этого с очевидностью следует необходимость защиты объектов права интеллектуальной собственности. Защита исключительных (авторских, смежных, патентных и др.) прав – это совокупность мер, целью которых является восстановление и признание этих прав в случае их нарушения.

В зависимости от отрасли права, обеспечивающей защиту авторских и смежных прав, выделяют следующие способы защиты.

1. Гражданско-правовой способ защиты – возмещение имущественного ущерба автору или иному правообладателю. Прежде всего, защита авторских и смежных прав происходит способами, предусмотренными ст.12 ГК РФ. Защита авторских и смежных прав данным способом имеет ряд специфичных черт. Личные неимущественные права подлежат защите независимо от вины правонарушителя.

Например, в случае если издательство не знало и не могло знать о том, что публикует произведение, присвоенное другим автором, оно должно принять все меры по устранению правонарушения. Если допущена ошибка в имени автора, то автор вправе требовать внесения изменений в тираже, либо публикации, осведомляющей о допущенной ошибке и какое имя считать правильным, либо требовать запрещения выпуска произведения в свет. На требования о защите личных неимущественных прав не распространяется исковая давность.

При нарушении имущественных авторских прав:

а). может быть наложен арест на контрафактные (т. е. изготовленные с нарушением авторских прав) экземпляры до рассмотрения дела по существу;

б). автор или иной обладатель исключительных прав вправе требовать по своему выбору от нарушителя вместо возмещения убытков выплаты компенсации (размер – от 10 тыс. руб. до 5 млн. руб., двукратный размер стоимости экземпляров произведений или объектов смежных прав либо двукратный размер стоимости прав на использование произведений или объектов смежных прав);

в). автор или иной обладатель исключительных прав вправе требовать возмещения морального вреда, а также компенсации упущенной выгоды;
г). суд может вынести решение о конфискации контрафактных экземпляров произведения, а также материалов и оборудования, используемых для их воспроизведения.

2. Административно-правовой способ защиты – путем обращения в вышестоящие организации по отношению к организации нарушителю, в антимонопольный орган или творческий союз (например, ст. 14.20, 15.25, 16.19 КоАП РФ).

3. Уголовно-правовой способ защиты – за наиболее серьезное нарушение авторских и смежных прав. УК РФ предусмотрена ответственность, например, за плагиат (ст. 146), если это деяние причинило крупный ущерб автору или иному правообладателю или совершено в крупном или особо крупном размере.

6.5. Понятие и принципы патентного права

Патентное право – совокупность правовых норм, регулирующих имущественные, а также связанные с ними личные неимущественные отношения, возникающие в связи с созданием и использованием объектов патентного права: *изобретений, полезных моделей и промышленных образцов*.

Объединение их в данной подотрасли гражданского права обусловлено тем, что: а) данные объекты интеллектуальной собственности сходны между собой и существенно отличаются от иных объектов; б) охрана данных объектов осуществляется в единой форме – путем выдачи патента; в) правовое регулирование данных объектов имеет определенную сходность [1].

Принципы патентного права закреплены в законе Российской Федерации от 23 сентября 1992 г. № 35171, который регулирует отношения, возникающие в связи с правовой охраной и использованием изобретений, полезных моделей и промышленных образцов. Закон состоит из 8 разделов и 45 статей.

Раздел 1 «Общие положения» определяет отношения, регулируемые Патентным законом, устанавливает место и роль федерального органа исполнительной власти по интеллектуальной собственности, а также закрепляет общие положения правовой охраны изобретения, полезной модели, промышленного образца.

Раздел 2 «Условия патентоспособности» определяет условия патентоспособности изобретения, полезной модели, промышленного образца, устанавливая их существенные признаки, критерии охраноспособности, что является изобретением, полезной моделью, промышленным образцом, а что нет.

Раздел 3 «Авторы и патентообладатели» посвящен субъектам патентных правоотношений. Здесь устанавливается, кто может быть автором изобретения, полезной модели, промышленного образца, патентообладателем, кому принадлежит право на получение патента.

Раздел 4 «Исключительное право на изобретение, полезную модель или промышленный образец» закрепляет правовой статус патентообладателя, устанавливает исчерпывающий перечень действий, не признаваемых нарушением исключительного права патентообладателя, определяет, что такое право преждепользования и др.

Раздел 5 «Получение патента» устанавливает детальный порядок предоставления патента компетентными органами. Закрепляется порядок подачи заявки на выдачу патента, требования к заявкам, порядок проведения экспертизы заявок, осуществление временной правовой охраны и др.

Раздел 6 «Прекращение и восстановление действия патента» регулирует порядок признания недействительным патента, досрочного прекращения и восстановления его действия.

Раздел 6.1 «Особенности правовой охраны секретных изобретений» закрепляет положения о порядке получения патента на секретное изобретение, а также регистрации и выдачи патента, признания его недействительным, уста навливает процедуру изменения степени секретности и рассекречивания изобретения.

Раздел 7 «Защита прав патентообладателей и авторов» устанавливает ответственность за нарушение патентного законодательства, а также закрепляет перечень споров, рассматриваемых в судебном порядке.

Раздел 8 «Заключительные положения» закрепляет нормы о патентных пошлинах, государственном стимулировании создания и использования изобретений, полезных моделей, промышленных образцов, патентных правоотношениях с участием иностранных лиц и за рубежом.

Закон закрепляет за патентообладателем исключительное право на использование запатентованного объекта. Это означает, что только патентообладатель может изготавливать, применять, ввозить, продавать и иным образом вводить в хозяйственный оборот запатентованную разработку. Другие лица должны воздерживаться от ее использования, не санкционированного патентообладателем, а патентообладатель вправе требовать этого.

Закон предоставляет охрану лишь тем разработкам, которые в официальном порядке признаны патентоспособными изобретениями, полезными моделями и промышленными образцами. Это означает, что для получения охраны заинтересованное лицо должно соблюсти процедуру получения патента: надлежащим образом оформить заявку, подать ее в орган, осуществляющий регистрацию и выдачу патента и пр. Только то изобретение, полезная модель и промышленный образец охраняется патентным правом, на которые соответствующим образом выдан патент, проведена процедура признания государством новизны и промышленной применимости изобретения, полезной модели и промышленного образца.

Законом признаются и охраняются права и интересы не только патентообладателей, но и действительных создателей изобретений, полезных моделей и промышленных образцов. Это означает, что патентное право закрепляет интерес лиц, не получающих патент на использование изобретений, полезных моделей и промышленных образцов, но непосредственно участвующих в их создании и разработке.

Закон предусматривает право получения вознаграждения за разработку изобретений, полезных моделей и промышленных образцов в порядке служебного задания. За создателями объектов патентного права сохраняются личные неимущественные права, являющиеся бессрочными и непередаваемыми.

Автором изобретения, полезной модели, промышленного образца признается физическое лицо, чьим творческим трудом они созданы. Лицо, указанное в качестве автора в заявке на выдачу патента, считается автором изобретения, полезной модели, промышленного образца, если не доказано иное. К этой категории лиц относятся также иностранные лица и лица без гражданства. Иностранные физические и юридические лица осуществляют патентные права в отношении изобретений на основании международных договоров, участником которых является Российская Федерация.

Закон не устанавливает возрастных ограничений для признания гражданина автором изобретения. Однако реально осуществлять свои права в отношении созданного изобретения граждане могут с 14 лет (ст. 26 ГК РФ).

В качестве авторов признаются только лица, *внесшие личный творческий вклад* в создание изобретения. Не признаются авторами физические лица, не внесшие творческого вклада в создание изобретений и оказавшие авторам только техническую, организационную или материальную помощь либо только способствовавшие оформлению права на него и его использованию.

Под технической помощью могут пониматься изготовление чертежей, образцов, осуществление расчетов, проведение экспериментов по программе, указанной изобретателем, подбор информационных материалов по просьбе изобретателя и т. д. Не могут рассматриваться в качестве соавторов лица, которые ограничились тем, что только высказали общую идею изобретения, полезной модели, промышленного образца, не материализовав ее.

Патентообладатель – лицо, владеющее патентом на объект промышленной собственности и вытекающими из патента исключительными правами на использование этого объекта. Патентообладателем может быть автор разработки, его наследники, работодатель или иные лица. Патент может быть выдан любому гражданину или юридическому лицу, обладающему на момент подачи заявки исключительными правами на использование объекта промышленной собственности. Исключительное право может переходить в порядке универсального правопреемства. Например, права могут быть переданы юридическому лицу за вознаграждение. Правопреемником автора изобретения, полезной модели или промышленного об-

разца или работодателя, имеющего право на получение патента, можно стать также в результате наследования.

Право на получение патента на служебное изобретение, служебную полезную модель, служебный промышленный образец принадлежит работодателю, если законом или договором между ним и работником (автором) не предусмотрено иное.

Возникновение и существование патентных прав связывается с получением патента. Для получения патента на изобретение, полезную модель или промышленный образец необходимо подать в регистрирующий орган заявку. Для того чтобы подать заявку, необходимо обладать правом на получение патента в соответствии с Патентным законом РФ [1].

К лицам, имеющим право подать заявку, относятся:

а) автор;

б) любое лицо, которому автор предоставил право на получение патента;

в) патентные поверенные.

Заявка на выдачу патента на объект промышленной собственности должна относиться к одному изобретению или группе изобретений, связанных между собой настолько, что они образуют единый изобретательский замысел, и должна содержать:

- заявление о выдаче патента с указанием автора (авторов) объекта промышленной собственности и лица (лиц), на имя которого (которых) испрашивается патент, а также их место жительства или местонахождения;
- описание объекта промышленной собственности, раскрывающее его с полнотой, достаточной для осуществления;
- формулу объекта промышленной собственности, выражающую его сущность и полностью основанную на описании;
- чертежи и иные материалы, если они необходимы для понимания сущности объекта промышленной собственности;
- реферат.

К заявке на объект промышленной собственности прилагается документ, подтверждающий уплату патентной пошлины в установленном размере, или документ, подтверждающий основания для

освобождения от уплаты патентной пошлины, либо уменьшения ее размера, либо отсрочки ее уплаты.

После получения заявки проводится формальная экспертиза. Если не достает каких-либо документов, заявителю направляется запрос об устранении недостачи, который должен быть исполнен в течение двух месяцев со дня его получения. О положительном результате формальной экспертизы заявитель уведомляется незамедлительно.

После выдачи патента федеральный орган исполнительной власти по интеллектуальной собственности публикует в своем официальном бюллетене сведения о выдаче патента, включающие имя автора (авторов), если последний (последние) не отказался быть упомянутым в качестве такового (таковых), и патентообладателя, название и формулу изобретения или полезной модели или перечень существенных признаков промышленного образца и его изображение.

У охраны объектов патентных прав есть специфические особенности, о которых стоит упомянуть:

- патент действует в той стране, где выдан;
- защитить технологию во всем мире не получится, нужно делать выбор;
- длительность процедуры патентования;
- ограниченный срок действия патента, необходимость поддерживать в силе;
- ограниченный срок для подачи заявок в других странах, за пределами которого утрачивается новизна;
- патентовать объект не всегда целесообразно, можно найти другие способы коммерциализации;
- практическая сложность использования патентов без сопутствующих ноу-хау и знаний ключевых работников;
- ограниченность информации при проведении патентных экспертиз: информация о заявках, поданных в мире, как правило, недоступна.

Патент на изобретение действует до истечения двадцати лет с даты подачи заявки в федеральный орган исполнительной власти по интеллектуальной собственности, на полезную модель – до истечения пяти лет, на промышленный образец – до истечения десяти лет.

6.6. Способы коммерциализации объектов интеллектуальной собственности

Создание объекта интеллектуальной собственности (ОИС), получение прав на него, доведение объекта до "товарного" состояния – все эти действия связаны с немалыми затратами. Существуют два основных пути коммерциализации объектов интеллектуальной собственности (рис. 6.1) [10]:

- непосредственная коммерциализация через передачу (продажу) прав на объекты интеллектуальной собственности, как инновационные продукты;
- опосредствованная коммерциализация путем продажи инновационной продукции, в производстве которой использованы объекты интеллектуальной собственности.

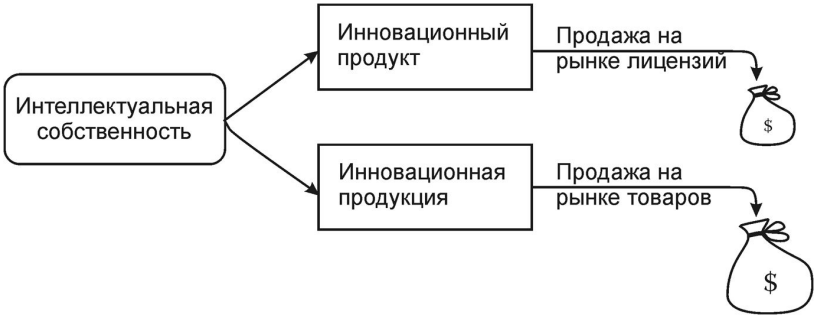


Рис. 6.1. Пути коммерциализации инновационного продукта.

Коммерциализация объекта интеллектуальной собственности, как инновационного продукта, осуществляется либо через использование прав в производстве инновационной продукции, либо через передачу прав на них [10]. Передать права на ОИС можно путем:

- передачи (уступки) всех имущественных прав другому лицу;
- внесения прав в уставный капитал предприятия;
- передачи прав пользования другому юридическому или физическому лицу.

Способы коммерциализации объектов интеллектуальной собственности представлены на рис. 6.2.

В специальной литературе подробно описаны указанные на рис. 6.2 способы коммерциализации ОИС. В этом смысле показательна книга П.Н. Цыбулева "Управление интеллектуальной собственностью", изданная в 2008 году в Киевском институте интеллектуальной собственности [10]. В книге подробно описаны многие аспекты управления ОИС и пути её коммерциализации. Наиболее выгодным из них с точки зрения прибыли является коммерциализация объектов интеллектуальной деятельности путем их использования в производстве инновационной продукции. Поскольку объем произведенной продукции может быть большим, то и пропорциональная ему прибыль может во много раз превышать стоимость прав на ОИС в случае их коммерциализации в качестве инновационных продуктов.

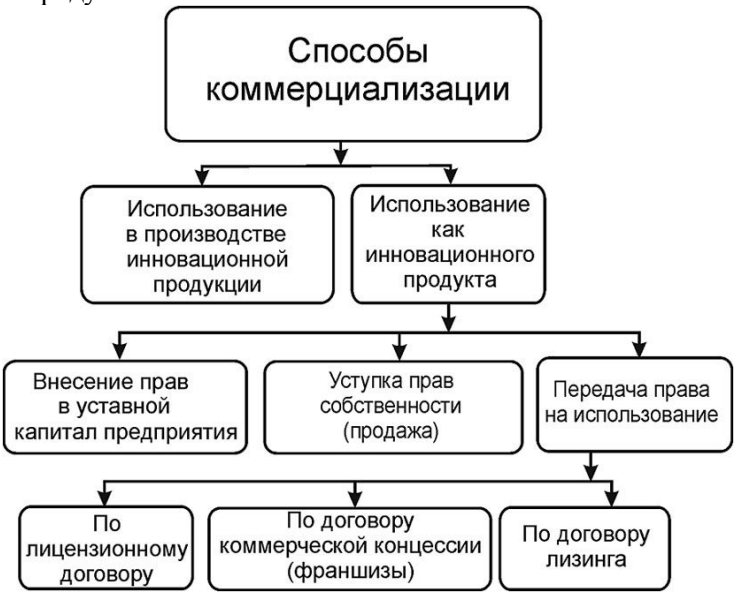


Рис. 6.2. Способы коммерциализации объекта интеллектуальной собственности.

Конечно, этот путь коммерциализации связан со значительными стартовыми затратами на доводку ОИС стадии его применимо-

сти. Однако, в случае успеха выпуск новой продукции на рынок дает возможность через определенный период не только компенсировать первоначальные затраты, но и получить значительную прибыль для дальнейшего развития производства. При этом вся полученная прибыль остается на предприятии.

Принятие решения о производстве инновационной продукции связано с многочисленными рисками. Поэтому, прежде чем принять такое решение, необходимо сравнить свою продукцию с аналогичной продукцией конкурентов, сделать экономическую оценку предполагаемого проекта.

Необходимо тщательно продумать схему коммерциализации. В условиях плановой экономики, которая существовала в Советском Союзе, была построена довольно четкая, но недостаточно эффективная схема введения ОИС в хозяйственный оборот.Научные центры (академические институты, университеты, научные лаборатории и др.) представляли результаты НИР, которые считались инновационной продукцией. Отраслевые научно-исследовательские институты отбирали наиболее перспективные разработки и дорабатывали их до технологий. Готовые технологии передавались промышленным предприятиям для использования.

Однако, в условиях рыночной экономики из цепи: "*научный центр => научно-исследовательский институт => промышленное предприятие*" выпало решающее звено – научно-исследовательский институт. И потому на современном этапе возникла необходимость в поиске новых путей введения результатов НИР в хозяйственный оборот. В современных условиях перспективной является схема коммерциализации ОИС, изображенная на рисунке 6.3.

По этой схеме ОИС разрабатывается в исследовательском центре (университете, академии наук, научно-исследовательском институте и т.п.), потом вместе с партнером (посреднической фирмой), которая имеет опыт по коммерциализации ОИС, создается новая компания, способная довести идею, разработанную в центре, "под ключ". Партнер (инновационный менеджер) находит промышленную компанию, которая заинтересована в использовании новой технологии, а также инвестора для новой компании. Новая компания выплачивает исследовательскому центру лицензионные плате-

жи за использование ОИС. В случае успеха проекта, новая компания может предоставить исследовательскому центру новый заказ на НИР и цикл повторяется.

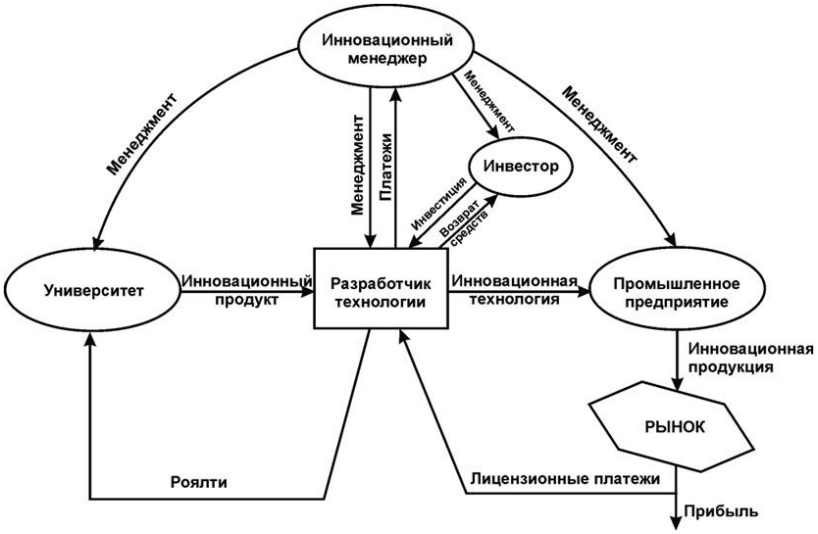


Рис. 6.3. Современная схема коммерциализации объектов интеллектуальной собственности через производство инновационной продукции.

Эта модель коммерциализации результатов НИР успешно используется в Великобритании и в США, прежде всего, для коммерциализации научных разработок, выполненных в Университетах.

Экономическая эффективность рассмотренных выше способов коммерциализации ОИС (рис. 6.2), может значительно (на порядки) отличаться для разных способов (табл. 6.2).

Таблица 6.2. Экономическая эффективность разных способов коммерциализации ОИС.

	Способ коммерциализации	Экономическая эффективность (тыс. долл. США)
1.	Продажа информации о разработке	5 - 20
2.	Передача прав на использование ОИС	15 - 50
3.	Использование ОИС для производства	Вся прибыль оста-

	инновационной продукции на предприятии	ется на предприятии
4.	Выращивание бизнеса для продажи большой корпорации	500 - 2 000
5.	Выращивание бизнеса для продажи на фондовой бирже	Больше чем 10 000

Из табл. 6.2 видно, что наименее эффективным является способ коммерциализации через продажу информации о разработке, когда продаются результаты НИР, права на которые не защищены охранным документом – патентом или свидетельством на изобретение.

Относительно низка эффективность такого способа коммерциализации, как передача прав на ОИС по лицензионным договорам (15-50 тыс. долл. США). Намного эффективнее способ коммерциализации ОИС посредством выращивания бизнеса на основе ОИС с дальнейшей продажей его большой компании или фондовому рынку, т.е. путем использования результатов НИР для производства инновационной продукции.

6.7. Базовые принципы управления ОИС

Объекты интеллектуальной собственности имеют двойную природу – правовую и экономическую. С одной стороны, это права на результат творческой деятельности, а с другой –результат этой деятельности.

Поскольку права интеллектуальной собственности имеют нематериальную природу, то они относятся к нематериальным активам, которыми, как и любыми другими активами, необходимо управлять.

Ниже приведены 12 принципов управления интеллектуальной собственностью, которые сформулированы на основе опыта многих предприятий [10,11]. Использование этих принципов позволит установить связь между интеллектуальной собственностью и финансовым состоянием предприятия, а также определить, каким образом управление ОИС обеспечит устойчивую прибыль от интеллектуальной собственности.

Принцип 1-й. Осознайте роль интеллектуальной собственности. В рутинных производствах руководители практически никогда не обращает внимания на интеллектуальную собственность. Но в таких областях производства, как биотехнологическая, фармацевтическая, радиоэлектронная, телекоммуникационная, разработка программных и информационных систем с использованием элементов искусственного интеллекта и т.п. принципиально невозможно создать конкурентный продукт без использования объектов права интеллектуальной собственности.

Например, в производстве сотового телефона фирмы NOKIA используется более 300 изобретений. Использование ноу-хау, изобретений и других объектов права интеллектуальной собственности позволяет обеспечить эффективную конкурентоспособность товара или услуги. Однако, это не означает напрямую, что больше объектов права интеллектуальной собственности будет на предприятии, тем большей будет его конкурентоспособность. Руководитель должен отличать интеллектуальную собственность как нематериальный актив от материальных активов и полезные для предприятия объекты интеллектуальной собственности от ненужных.

Катастрофой для руководителей может стать выпуск на рынок нового товара, который нарушает права других правообладателей. Судебные разбирательства по поводу нарушения этих прав интеллектуальной собственности способны привести к большим, а иногда к необратимым финансовым и моральным потерям. Поэтому очень важно, чтобы ведущую роль интеллектуальной собственности осознал руководитель предприятия, а не спускал эти насущные вопросы на нижние этажи менеджмента.

Принцип 2-й: Придерживайтесь баланса интересов субъектов права интеллектуальной собственности. В разработке и производстве инновационной продукции в общем случае, принимают участие три субъекта: заказчик, исполнитель и пользователь. Баланс интересов субъектов правоотношений в сфере интеллектуальной собственности, которая лежит в основе инновационной продукции, схематично представлен на рис. 6.4.

Рис. 6.4. Баланс интересов субъектов инновационной деятельности.

Если интересы находятся в зонах 1, 2 или 3, то они не будут восприняты двумя другими субъектами и такой проект обречен на провал. В зонах 4, 5 и 6 сходятся интересы двух субъектов. Но без заинтересованности третьего успех маловероятен. И только в зоне 7 интересы всех трех субъектов инновационной деятельности совпадают. Именно здесь можно рассчитывать на успех.

Относительно объектов интеллектуальной собственности интересы заключаются в справедливом распределении прав собственности между субъектами интеллектуальной собственности. Причем распределение прав должно производиться не в конце инновационного процесса, когда продукция войдет на рынок, а в самом начале.

Практика показывает, что вначале проекта, когда еще нет финансовой отдачи от объектов интеллектуальной собственности, а участниками движет энтузиазм – договориться о разделе прав значительно легче, чем в конце, когда появятся финансовые потоки, генерируемые объектами интеллектуальной собственности. Несоблюдение этого принципа стало причиной краха многих инновационных проектов.

Принцип 3-й: Сделайте интеллектуальную собственность частью стратегического бизнеса-планирования. Успешный бизнес начинается с прогнозирования будущего продукта на еще не существующем рынке или оценка его конкурентоспособности на имеющемся рынке. Поэтому с самого начала нужно определить, какие

именно объекты права ИС необходимо создать и/или приобрести для того, чтобы обеспечить конкурентоспособность товаров или услуг, которые, соответственно, будут производиться с их участием.

Разрабатываемая стратегия относительно оценки ОИС должна охватывать следующие позиции:

- разработка и внедрение процедур, которые обеспечивают контроль рисков относительно прав на объекты права интеллектуальной собственности других лиц, так как закрепленные за ними юридические права могут нести в себе угрозу долгосрочной прибыльности и росту бизнеса;
- максимизация прибыли, которая должна быть получена от любого имеющегося у предприятия объекта права интеллектуальной собственности, путем осуществления соответствующих управленческих процедур;
- осведомленность о новейших технологиях, которые были созданы в собственном бизнесе и разрабатываемых другими технологиях, а также оценка последних как потенциальной конкурентной угрозы;
- приобретение прав на созданные собственными силами объекты права интеллектуальной собственности;
- поиск объектов права интеллектуальной собственности, которые касаются важных технических новинок и которые имеют надежную правовую охрану, а также оценка того, следует или не следует приобрести эти права интеллектуальной собственности, или, по меньшей мере, право на их использование [12].

Для предприятий-лидеров, а также для тех, кто стремится к лидерству, вопрос стратегического планирования относительно интеллектуальной собственности будет одним из важнейших.

Принцип 4-й: Выявите собственные объекты интеллектуальной собственности. На российских предприятиях, к большому сожалению, стала типичной ситуация, когда руководители высшего звена не владеют информацией о том, какие объекты права интеллектуальной собственности существуют на их предприятии, кому принадлежат права на эти объекты, не истек ли их юридический срок действия и т.п. Без ответов на эти вопросы невозможно управление интеллектуальной собственностью.

Очевидно, для разных предприятий "портфель" интеллектуальной собственности будет разным. Для одних это будет фирменное наименование и, возможно, некоторые простые ноу-хау. Для других бизнес будут определять комплексные технологии, включающие в себя десятки, даже сотни объектов права интеллектуальной собственности: изобретения, полезные модели, промышленные образцы, торговые марки, коммерческие наименования и т.п. Важно идентифицировались все объекты права интеллектуальной собственности, даже если они будут незначительными или связанными с устаревшими технологиями.

Результатом применения этого принципа управления интеллектуальной собственностью является то, что руководство предприятия будет иметь *правильное и всеобъемлющее представление об интеллектуальной собственности в их бизнесе*, о связанных с нею юридических правах и как эта интеллектуальная собственность применяется для обеспечения ежедневной деятельности по управлению предприятием.

Принцип 5-й: Соберите информацию об интеллектуальной собственности конкурентов. Важным элементом стратегии управления интеллектуальной собственностью является информированность относительно прав интеллектуальной собственности, которые могли бы сделать конкурентов способными мешать вашему бизнесу в использовании новых товаров, технологических процессов или услуг.

Цель заключается в том, чтобы установить, может ли любое из прав интеллектуальной собственности других лиц препятствовать любой осуществляемой вами коммерческой или исследовательской деятельности. Желательно это сделать до того, как будут осуществлены значительные инвестиции в разработку инновационной продукции, которые могут оказаться напрасными, если в последующем придется отказаться от разработки, поскольку не удается преодолеть барьеры, созданные правами интеллектуальной собственности конкурентов.

Принцип 6-й: Идентифицируйте нужную вам интеллектуальную собственность. При внимательном исследовании товара, который ваша компания готовит для выхода на рынок, может оказаться,

что не все элементы товара защищены правом интеллектуальной собственности. Еще хуже, если какие-то элементы вашего товара подпадут под действие чужих прав интеллектуальной собственности. В этом случае говорят, что товар (изделие) не обладает *патентной чистотой*. Такой товар опасно выводить на рынок, ибо в случае нарушения чужих прав производителю товара грозят большие неприятности (конфискация товара, штраф и др.).

Поэтому необходимо определить, какие именно объекты интеллектуальной собственности необходимо создать или приобрести, чтобы сделать товар максимально защищенным правом интеллектуальной собственности.

Принцип 7-й: Создайте или приобретите право на нужные вам объекты права интеллектуальной собственности. После того, как будут выявлены объекты права интеллектуальной собственности, которые необходимы дополнительно для обеспечения конкурентоспособности инновационной продукции, следует решить, где их взять. Есть два пути для этого. Первый - создать их собственными силами или на заказ. Второй – приобрести права на такие объекты через договор уступки прав или лицензионный договор.

Второй путь имеет ряд преимуществ. Во-первых, он дает возможность выиграть время. Во-вторых, он менее рискован, ибо уже пройдена стадия разработки. Однако всегда остается опасность, что вам продадут устаревшую разработку, поскольку мало кто захочет продать новые разработки. Тенденция такова, что продают права на результаты интеллектуальной деятельности, которые уже заменяют новыми, более совершенными.

Принцип 8-й: Оцените экономическую эффективность вашей интеллектуальной собственности. Оценка стоимости прав на объекты интеллектуальной собственности необходима, по крайней мере, для достижения таких довольно важных и четких целей:

- узнать о расходах на создание, обретение правовой охраны и поддержание действия прав на объект права интеллектуальной собственности;
- оценить размер денежных потоков, которые способны генерировать эти объекты при использовании их в собственном производстве;

- определить цену, за которую можно продать (уступить) права на объект права интеллектуальной собственности или передать права пользования этим объектом по лицензионному договору.

Инвестирование в интеллектуальную собственность оправдано только тогда, когда это может принести дополнительную прибыль на инвестицию. Возможно, есть единственное исключение из этого правила, когда объект права интеллектуальной собственности может быть использован для блокирования конкурента или для укрепления собственных позиций на рынке.

Принцип 9-й: Думайте о налоге на интеллектуальную собственность. В процессе управления интеллектуальной собственностью необходимо учитывать вопросы бухгалтерского учета и налогообложения. Многие руководители также не задумываются об этом – и это может впоследствии привести к неприятным "разборкам" с налоговыми органами.

Законодательство разрешает ставить права на объекты интеллектуальной собственности на бухгалтерский учет предприятия и производить их амортизацию. Это может привести, по крайней мере, к двум противоположным последствиям в налогообложении.

С одной стороны, постановка на бухгалтерский учет автоматически запускает механизм амортизации объектов права интеллектуальной собственности и тем самым уменьшает на величину амортизации прибыль, которая является базой для налогообложения. При этом предприятие не отдает часть денег в бюджет, а оставляет их в своем распоряжении.

С другой стороны, на величину стоимости поставленных на баланс объектов права интеллектуальной собственности увеличиваются активы предприятия, т.е. возникает дополнительная стоимость, которая облагается налогом на добавленную стоимость. Даже если права на ОИС приобретены предприятием бесплатно, то они отображаются в бухгалтерском учете со справедливой стоимостью, а в связи с ведением налогового учета, сумма стоимостей учтенных бесплатно полученных ОПИС увеличивает валовой доход, что приводит к увеличению базы налогообложения, а следовательно к дополнительному налогу.

Размер налога будет зависеть от того, будут ли эти объекты созданы собственными силами, или они будут приобретены по лицензионному договору. И если да, то от кого - физического или юридического лица? Эти нюансы необходимо учитывать при управлении интеллектуальной собственностью. Базовый принцип: необходимо минимизировать налоги, разумеется, соблюдая нормы действующего законодательства.

Принцип 10-й: Будьте готовы защищать права на свою интеллектуальную собственность. На каком-то этапе ваша интеллектуальная собственность привлечет внимание конкурентов. Они забеспокоятся, когда ваш успех зацепит их бизнес. Это может произойти, когда ваш товар выйдет на рынок. Или если сведения о разработке вами объектов права интеллектуальной собственности, которые представляют потенциальную опасность для конкурента, станут доступными для него.

У недобросовестного конкурента может появиться соблазн нарушить ваши права, т.е. использовать их без вашего разрешения. В этом случае возникает необходимость в защите своих прав, в том числе в судах, и к этому нужно готовиться заранее. Хрестоматийный пример – Чарльз Гудьир после десяти лет настойчивых исследований запатентовал первую в мире технологию вулканизации резины (патент США № 3633 от 15.06.1844 г.). Он мог невероятно разбогатеть, когда Генри Форд поставил производства автомобилей на конвейер! Однако технологический процесс оказался простым для копирования, конкуренты быстро украли его, а Ч.Гудьир, не имея средств для защиты своих прав от недобросовестных конкурентов, умер в нищете. Такая же история произошла с Лео Зиппером, изобретателем застежки "молния".

Предприятие не должно умышленно или по незнанию нарушать чужие права на объекты интеллектуальной собственности, поскольку это может обернуться для него как значительными материальными, так и моральными потерями. Во всяком случае, предприятие должно выработать правила поведения относительно недобросовестных конкурентов и предусмотреть средства, которые могут быть эффективными для отстаивания своих прав в суде. Если у вас есть права, но нет средств для их защиты, то ваши права скорее всего будут безнаказанно нарушаться.

Принцип 11-й: Измеряйте эффективность управления интеллектуальной собственностью. Если расходы на управления интеллектуальной собственностью будут превышать выгоды, полученные от управления, то такое управление следует считать неэффективным, и наоборот.

Для оценки эффективности управления, необходимо еще до введения системы управления установить численные показатели (метрики), которые будут свидетельствовать об эффективности управления. Отправными точками могли бы стать затраты на поддержание контроля над интеллектуальной собственностью, объем продаж прав на них по лицензионным договорам и др. Периодическая оценка стоимости "гудвилла" предприятия также может служить интегральной характеристикой эффективности управления интеллектуальной собственностью.

Принцип 12-й: Сформируйте команду компетентных, творчески мыслящих и целеустремленных сотрудников. Этот общий для многих аспектов управленческой деятельности принцип особо важен, ибо без его соблюдения вряд ли будут эффективно работать остальные принципы. Если на предприятии нет умных, энергичных, творчески мыслящих людей, то любые инвестиции, вложенные как в само предприятие, так и в интеллектуальную собственность, скорее всего, уйдут в песок. Важность этого принципа заключается еще и в том, что именно творческие люди создают такие составляющие интеллектуального капитала, как информация и объекты права интеллектуальной собственности.

Сформулированные принципы будут эффективны лишь в том случае, когда они идентифицируются по отношению к стратегическим целям предприятия, осознаются высшим менеджментом и последовательно и комплексно внедряются в повседневную деятельность.

Список литературы

1. *Резепова В.Е.* Право интеллектуальной собственности (http://be5.biz pravo/i003/toc.htm)

2. Общие положения о праве интеллектуальной собственности (http://internetandlaw.blogspot.ru/2012/09/blog-post_4.html)
3. *Шумпетер Й.* Теория экономического развития// Пер. с англ. – М.: Прогресс, 1992. – 456 с.
4. *Sullivan Patrick A.* Profiting from Intellectual Capital: Extracting Value from Innovation. – John Willey&Sons, inc., 1998.
5. *Sullivan Patrick A.* Value-driven Intellectual Capital: How to convert intangible corporate assets into market value. - John Willey&Sons, Inc., 2000.
6. *Sveiby K.-E.* Intellectual Capital and Knowledge Management. (http://www.sveiby.com.am / Intellectual Capital. html)
7. *И. А. Близнец, К. Б. Леонтьев.* Понятие «интеллектуальная собственность»: формулировка проблемы (www.copyright.ru)
8. *Косовец А. А.* Правовое регулирование электронного документооборота // Вестник Московского университета. Сер. 11, "Право", 1997, № 4
9. *Комаров С.Н., Кочергина И. И.* Вопросы учета программ для ЭВМ и баз данных // Юридический консультант, 1998, № 2.
10. *П.Н.Цыбулев.* Управление интеллектуальной собственность. – К.: Гос. ин-т интел. собств., 2008. – 324 с.
11. *Барух Лев.* Нематериальные активы: Управление, измерение, отчетность. – М.: Квинто-консалтинг, 2003. – 228 с.
12. *Зинов В.Г.* Управление интеллектуальной собственностью: Учебное пособие. – М.: Дело, 2003. – 512 с.

Заключение

С внедрением компьютера практически во все виды деятельности человека решающей силой стали не только недвижимость или производственный капитал, но в большой степени – сам человек, его знания, информация, которыми он обладает.

В начале XXI века около 65 % доходов в развитых странах люди получают за счёт интеллектуальной деятельности, поддержанной компьютерными технологиями. Информация, знания и развитые коммуникации дают свободу и уверенность в своих силах, в правильности выбранной стратегии, являются непременным фактором успеха.

В материал этой книги вошли новые научные вошли результаты, полученные молодыми авторами Константином Амелиным (Amelin K., Granichin O. Randomized controls for linear plants and confidence regions for parameters under external arbitrary noise // In: Proc. of the 2012 American Control Conference, Montréal, Canada, June 27-June 29, 2012, pp. 851-856) и Натальей Амелиной (Amelina, N., A. Fradkov, and K. Amelin Approximate Consensus in Multi-agent Stochastic Systems with Switched Topology and Noise // Proc. of MSC IEEE 2012, October 3-5, 2012, Dubrovnik, Croatia, pp. 445-450). И Константин, и Наталья получили почетные дипломы финалистов конкурсов лучших аспирантских научных работ на двух последних конференциях, организуемых Обществом систем управления (Control Systems Society – CSS) Международного института инженеров по электротехнике и электронике (IEEE) – американской конференции по управлению (American Control Conference – ACC-2012) в Монреале и мультиконференции по системам и управлению (Multi-Conference on System and Control – MSC-2012) в Дубровнике.

Авторы надеются, что внимательное изучение книги и проработка лабораторных работ будут способствовать появлению у читателей новых интересных инновационных решений в области разработки программных приложений для мобильных платформ!

Н а у ч н о е и з д а н и е

К.С.Амелин, Н.О.Амелина, О.Н.Граничин, В.И.Кияев

Разработка приложений для мобильных интеллектуальных систем на платформе Intel Atom

Печатается без издательского редактирования

Оригинал-макет О.Н. Граничина, В.И. Кияева

Дизайн обложки В.И. Кияева

Подписано в печать 11.10.12. Формат 60 x 84/$_{16}$
Бумага офсетная. Печать офсетная

Усл. печ. л. Тираж 500 экз.

Заказ №