ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

1(161)
2010

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с ноября 1995 г.

УЧРЕДИТЕЛЬ
Издательство "Новые технологии"

СОДЕРЖАНИЕ

Информация о журнале доступна по сети Internet по адресу http://www.**informika.ru/text/magaz/it/** или **http://novtex.ru/IT.**

Журнал включен в систему Российского индекса научного цитирования.

Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

УДК 004.056.53

И. В. Аникин, канд. техн. наук, доц.,
Казанский государственный технический университет им. А. Н. Туполева,
e-mail: anikingor@evm.kstu-kai.ru

Метод количественной оценки уровня ущерба от реализации угроз на корпоративную информационную сеть

Предложен метод количественной оценки ущерба от угроз на корпоративную информационную сеть. Определение уровня критичности активов осуществляется на основе экспертных оценок, метода анализа иерархий и формальной модели корпоративной информационной сети. Предложен алгоритм "Экспертная оценка категорий" для количественной оценки критичности информационных активов.

Ключевые слова: *информационная безопасность корпоративных информационных сетей, категорирование активов, угрозы безопасности.*

Введение

Проблема обеспечения информационной безопасности (ИБ) корпоративных информационных сетей (КИС) часто решается [1, 2] с позиции управления рисками ИБ. Это требует разработки методов достоверного оценивания уровня ущерба от реализации угроз ИБ, а также вероятностей данной реализации. Методы оценивания ущерба широко представлены в [3—7], однако существует множество проблем, связанных с их практическим применением.

Проведенный анализ [7] показывает, что определение точного количественного уровня ущерба часто не представляется возможным. В связи с этим на практике для оценки рисков широкое распространение получили методы, использующие приближенные экспертные оценки.

Первая группа данных методов [6, 8] основана на определении уровней ущерба экспертами на порядковых шкалах. Они позволяют получить приближенные оценки ущерба при отсутствии необходимых данных для более точной количественной оценки. К основным недостаткам этих методов следует отнести невысокую точность получаемых результатов, а также сложность их использования в задачах управления рисками ИБ.

Вторая группа методов [2, 9] предполагает количественную оценку экспертом уровня ущерба в условных единицах. Данные методы позволяют формировать более точные оценки, однако для них остро встает задача обеспечения достоверности результатов, решение которой осложняется по следующим причинам:

- эксперт должен обладать точной и достоверной информацией о ценности активов КИС и особенностях воздействия на них угроз, что на практике обеспечить достаточно затруднительно;
- ущерб, наносимый угрозой, часто имеет качественный характер и его сложно оценить в количественном виде;
- ущерб во многом определяется характером взаимодействия различных активов КИС, который не учитывается большинством методов оценки.

Целью данной работы является повышение точности и достоверности оценок уровня ущерба от реализации угроз ИБ в КИС за счет разработки количественного метода, лишенного представленных выше недостатков.

Модель корпоративной информационной сети

Для формализации активов корпоративной информационной сети и особенностей взаимодействия между ними предлагается модель КИС, в которой рассматриваются следующие основные виды активов, подверженные угрозам ИБ:

- информационные активы;
- рабочие станции (АРМ) пользователей;
- серверы;
- телекоммуникационное оборудование;
- ИТ-сервисы, представляющие собой ИТ-услуги, поддерживающие реализуемые в КИС бизнес-процессы.

Взаимодействие данных активов представлено на рис. 1. Связи между активами должны быть учтены при определении уровней ущерба от реализации угроз.

Разработанная модель КИС включает в себя следующие основные компоненты: информационную модель $M_{\text{инф}}$, множество аппаратных активов $A_{\text{а.о}}$, логическую структуру $G_{\text{л.с}}$, модель ИТ-сервисов $M_{\text{ИТС}}$. Множество аппаратных активов КИС включает в себя множества АРМ

$A_{АРМ}$, серверов $A_{серв}$, телекоммуникационного оборудования $A_{тк}$.

Логическая структура КИС представлена в виде графа $G_{л.с} = (V, E)$ с двухцветной раскраской вершин и ребер. Вершинами графа являются взаимодействующие сегменты КИС и телекоммуникационное оборудование, а ребрами — контролируемые и неконтролируемые каналы связи (рис. 2, см. четвертую сторону обложки, рис. 3). Необходимость раскраски вершин и ребер обусловлена потребностями решения задачи выработки оптимальной стратегии защиты КИС. Множество каналов связи $A_{к.св}$ определяется ребрами графа $G_{л.с}$ с петлями в вершинах, соответствующих сегментам сети.

Рис. 1. Взаимодействие активов КИС

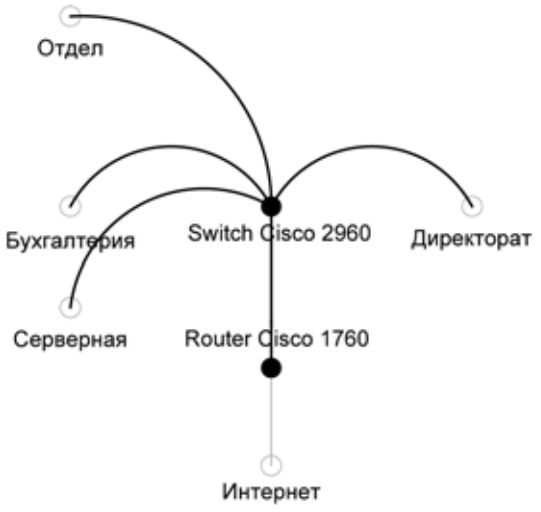


Рис. 3. Формализация логической структуры КИС в виде раскрашенного графа

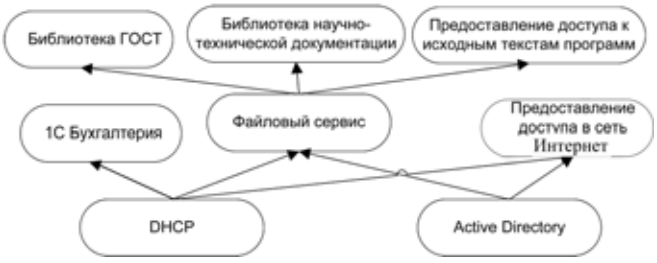


Рис. 4. Пример дерева зависимостей ИТ-сервисов

Рис. 5. Трехдольный граф $G_{угр}$

Для формализации модели ИТ-сервисов введены множества $A_{ИТС}$ реализуемых ИТ-сервисов и ориентированный граф (лес) $G_{ИТС}$ (рис. 4), представляющий деревья зависимостей ИТ-сервисов.

Каждому из элементов $a_{ИТС} \in A_{ИТС}$ поставлен в соответствие граф $G_{пред\_ИТС}$ предоставления ИТ-сервиса, являющийся ориентированным подграфом $G_{л.с}$ с возможными петлями. Определены два бинарных отношения: $R^{а.о}_{ИТС} \subseteq A_{ИТС} \times A_{а.о}$ — между ИТ-сервисами и реализующим их оборудованием и $R^{инф}_{ИТС} \subseteq A_{ИТС} \times A_{инф}$ — между ИТ-сервисами и предоставляемыми ими информационными активами.

Информационная модель КИС базируется на введенной модели *IP* информационных потоков и тернарном отношении $R^{а.о}_{инф}$, определяющем характер работы оборудования с информацией (хранение или обработка). Каждый из информационных потоков модели *IP* представляет собой тройку элементов $\{a_{инф}, a_{ИТС}, Path\}$, где $a_{ИТС}$ — ИТ-сервис, предоставляющий ресурс $a_{инф}$, $Path = \{p_j\}_{j=1}^{N_p}$ — множество цепей в графе $G_{л.с}$, представляющих собой информационные потоки для $a_{инф}$ от отправителя к получателю.

Модель угроз КИС определена в виде трехдольного графа $G_{угр}$ (рис. 5), вершинами которого являются элементы множества уязвимостей $A_{уязв}$, угроз $A_{угр}$, объектов защиты $A_{о.з} = A_{инф} \cup A_{а.о} \cup A_{к.св}$.

Ребра графа $G_{угр}$ между угрозами и активами помечены элементами множества $\{cc, ci, cd\}$. Ребро помечено элементом:

- *cc*, если угроза приводит к хищению информации;
- *ci*, если угроза к модификации или отрицанию подлинности;

- *cd*, если угроза приводит к уничтожению/блокированию актива.

Метод количественной оценки уровня ущерба от реализации угроз

Уровень ущерба от реализации угроз ИБ определяется критичностью активов КИС, на которые воздействуют угрозы. Поэтому в первую очередь предлагается определять следующие категории активов:

- конфиденциальность $cc_{\text{инф}}$, целостность $ci_{\text{инф}}$, доступность $cd_{\text{инф}}$ для информационных ресурсов;
- критичность $cc_{\text{АРМ}}$ для АРМ;
- критичность $cc_{\text{серв}}$ и доступность $cd_{\text{серв}}$ для серверов;
- доступность $cd_{\text{тк}}$ для телекоммуникационного оборудования;
- доступность $cd_{\text{ИТС}}$ для ИТ-сервисов.

Категория доступности *cd* характеризует ущерб, наносимый в результате блокирования доступа к данному активу в течение условного временного интервала (например, часа).

Для определения категорий критичности информационных активов введены частные показатели ущерба: $\text{ЧП}_j^{\text{к}}$, $j = 1, .., p^{\text{к}}$ — для конфиденциальности; $\text{ЧП}_j^{\text{ц}}$, $j = 1, ..., p^{\text{ц}}$ — для целостности; $\text{ЧП}_j^{\text{д}}$, $j = 1, .., p^{\text{д}}$ — для доступности. Данные показатели характеризуют различные виды ущербов при нарушении соответствующих свойств информационных активов.

Одной из основных проблем количественной оценки категорий критичности является качественный характер большинства частных показателей ущерба. В этих условиях для формирования оценок предложено использовать метод анализа иерархий [10]. Сформирована четырехуровневая иерархия, используемая для определения весовых коэффициентов критичности информационных активов. На первом уровне иерархии представлен исследуемый вид влияния — нарушение конфиденциальности, целостности или доступности. На втором уровне иерархии размещаются три группы частных показателей ущерба: внешний социополитический ущерб, внутренний ущерб, финансовый ущерб. На третьем уровне иерархии непосредственно размещаются частные показатели ущерба ЧП_j. На четвертом уровне иерархии размещаются исследуемые информационные активы.

Обозначим через $w_{\text{ГЧП}_i}$ — приоритеты групп частных показателей (ГЧП), $w_{\text{ЧП}_j}^{\text{ГЧП}_i}$ — приоритеты частных показателей (ЧП) в группе, $w_{asset_1}^{\text{ЧП}_1}$ — приоритеты активов по отношению к частным показателям ущерба. Тогда приоритеты критичности w_{a_i} активов определяются следующим образом:

$$w = (w_{a_1}, ..., w_{a_n}) = (w_{\text{ГЧП}_1}, w_{\text{ГЧП}_2}, w_{\text{ГЧП}_3}) \times$$

$$\times \left(\begin{pmatrix} w_{\text{ЧП}_1}^{\text{ГЧП}_1} & w_{\text{ЧП}_2}^{\text{ГЧП}_1} & \dots & w_{\text{ЧП}_m}^{\text{ГЧП}_1} \\ w_{\text{ЧП}_1}^{\text{ГЧП}_2} & w_{\text{ЧП}_2}^{\text{ГЧП}_2} & \dots & w_{\text{ЧП}_m}^{\text{ГЧП}_2} \\ w_{\text{ЧП}_1}^{\text{ГЧП}_3} & w_{\text{ЧП}_2}^{\text{ГЧП}_3} & \dots & w_{\text{ЧП}_m}^{\text{ГЧП}_3} \end{pmatrix} \times \right.$$

$$\left. \times \begin{pmatrix} w_{a_1}^{\text{ЧП}_1} & w_{a_2}^{\text{ЧП}_1} & \dots & w_{a_n}^{\text{ЧП}_1} \\ w_{a_1}^{\text{ЧП}_2} & w_{a_1}^{\text{ЧП}_2} & \dots & w_{a_n}^{\text{ЧП}_2} \\ \dots & \dots & \dots & \dots \\ w_{a_1}^{\text{ЧП}_m} & w_{a_2}^{\text{ЧП}_m} & \dots & w_{a_n}^{\text{ЧП}_m} \end{pmatrix} \right).$$

Использование метода анализа иерархий затруднено в случае, когда число элементов на одном уровне превышает 15. Поэтому в случае необходимости множество всех информационных активов разбивается на более мелкие группы, внутри которых осуществляется детальный анализ.

Предлагается ***алгоритм "Экспертная оценка категорий"*** для количественной оценки уровней критичности информационных активов.

1. Формируется экспертная группа, включающая в себя n экспертов E_i, $i = \overline{1, n}$. Каждому из экспертов E_i назначается вес важности $1 \leqslant w_i \leqslant 5$.

2. Осуществляется категорирование информационных активов на порядковой шкале согласно методу, предложенному в [11].

3. Формируются четырехуровневые иерархии для каждой из групп информационных активов — отдельно для оценки конфиденциальности, целостности и доступности.

4. Каждым из экспертов формируются приоритеты информационных активов путем анализа соответствующих иерархий. Далее определяются средневзвешенные приоритеты информационных активов для группы экспертов:

$$w_{a_k}^{Category} = \frac{\sum_{i=1}^{n} w_i (w_{a_k}^{Category})_i}{\sum_{i=1}^{n} w_i},$$

где a_k — исследуемый актив.

5. В каждой иерархии экспертным путем выбирается актив a_i, для которого осуществляется оценка степени критичности в условных единицах. Для согласования мнений экспертов в группе

используется метод Дельфи. Далее определяется критичность в условных единицах всех остальных активов a_j иерархии следующим образом:

$$C_j^{Category} = \frac{C_j^{Category}}{K_{ij}}, \quad (1)$$

где $K_{ij} = \frac{w_{a_i}^{Category}}{w_{a_j}^{Category}}$.

Актив a_i на шаге 5 алгоритма следует выбирать исходя из условия простоты количественной оценки критичности.

После оценки критичности информационных активов определяется критичность активов других видов в следующей последовательности.

Критичность АРМ определяется в виде

$$cc_{\text{АРМ}} = \Sigma cc + \Sigma ci,$$

где cc, ci — категории конфиденциальности и целостности информационных активов, обрабатываемых на АРМ.

Доступность ИТ-сервиса

$$cd_{\text{ИТС}} = \max(cd_{\text{ИТС эксп}}, \Sigma cd_{\text{инф}}, \Sigma cd_{\text{ИТС } tree}), \quad (2)$$

где

- $cd_{\text{ИТС эксп}}$ — формируется путем оценки частных показателей ущерба, связанных с нарушением доступности ИТ-сервиса, по алгоритму "Экспертная оценка категорий";
- $\Sigma cd_{\text{инф}}$ — сумма уровней доступности информационных активов, предоставляемых ИТ-сервисом;
- $\Sigma cd_{\text{ИТС } tree}$ — сумма уровней доступности зависимых ИТ-сервисов, определяемая согласно $G_{\text{ИТС}}$.

Определение доступности ИТ-сервисов осуществляется в строго определенном порядке. В первую очередь определяется доступность тех, которые являются висячими вершинами дерева зависимостей ИТ-сервисов $G_{\text{ИТС}}$. Нарушение доступности данных ИТ-сервисов не влечет нарушения доступности других, поэтому уровень $cd_{\text{ИТС}}$ определяется исходя из анализа частных показателей ущерба и значений $cd_{\text{инф}}$ предоставляемых информационных ресурсов. В последнюю очередь определяется доступность ИТ-сервисов, влияющих на другие.

Критичность сервера

$$cc_{\text{серв}} = cc + ci.$$

Доступность сервера определяется как сумма категорий доступности предоставляемых им ИТ-сервисов:

$$cd_{\text{серв}} = \sum_{j|(a^j_{\text{ИТС}},\, a_{\text{серв}}) \in R^{\text{а.о}}_{\text{ИТС}}} (cd^j_{\text{ИТС}}).$$

Доступность телекоммуникационного оборудования определяется как сумма категорий доступности ИТ-сервисов, предоставляемых посредством него:

$$cd_{\text{тк}} = \sum_{j|\, a_{\text{тк}} \in G^j_{\text{пред\_ИТС}}} (cd^j_{\text{ИТС}}).$$

После определения категорий критичности активов КИС вычисляется уровень ущерба от реализации угроз ИБ на данные активы следующим образом.

Определяется множество активов КИС, на которые воздействует угроза *threat* в виде

$$A_{threat} = \{a^j \mid \exists e_{\text{угр}},\ e_{\text{угр}}(threat,\ a^j)\},$$

где $e_{\text{угр}}$ — ребро в графе $G_{\text{угр}}$, связывающее угрозу *threat* и актив a^j.

Множество A_{threat} является объединением трех подмножеств $A_{threat} = A^{cc}_{threat} \cup A^{ci}_{threat} \cup A^{cd}_{threat}$, включающих в себя множество активов КИС, для которых угроза *threat* нарушает конфиденциальность, целостность, доступность соответственно. Один и тот же актив a^j может принадлежать одновременно нескольким подмножествам A^{cc}_{threat}, A^{ci}_{threat}, A^{cd}_{threat}.

Обозначим через cc_{threat}, ci_{threat}, cd_{threat} частные степени влияния угрозы на КИС при нарушении конфиденциальности, целостности и доступности активов соответственно. Тогда степень влияния угрозы *threat* вычисляется следующим образом:

$$IMPACT_{threat} = cc_{threat} + ci_{threat} + cc_{threat},$$

где $cc_{threat} = \sum_{a^j \in A^{cc}_{threat}} cc_{a^j}$; $ci_{threat} = \sum_{a^j \in A^{ci}_{threat}} ci_{a^j}$; $cd_{threat} = \sum_{a^j \in A^{cd}_{threat}} cd_{a^j}$.

Результаты экспериментов

Продемонстрируем работу разработанного метода для КИС, логическая структура которой представлена на рис. 2, а граф $G_{\text{л.с}}$ — на рис. 3. Данная КИС реализует ИТ-сервисы, дерево зависимостей которых представлено на рис. 4.

В качестве примера оценим уровень ущерба, наносимого реализацией следующих угроз:

1) отказ в доступе к файл-серверу;
2) отказ в доступе к сети Интернет;
3) отказ контроллера домена;
4) отказ маршрутизатора Cisco 1760.

Угрозы 1 и 2 реализуются на ИТ-сервисы КИС, а угроза 3 — на сервер, являющийся контроллером домена. Данный сервер реализует платформенные ИТ-сервисы DHCP и Active Di-

Количественные оценки доступности ИТ-сервисов

ИТ-сервис	Вес критичности	Количественная оценка доступности, у. е.
1С Бухгалтерия	0,2453	1614
Библиотека ГОСТ	0,076	500
Библиотека научно-технической документации	0,076	500
Предоставление доступа к исходным текстам программ	0,5658	3722
Предоставление доступа в сеть Интернет	0,0369	243
Файловый сервис	—	500 + 500 + 3722 = 4722
DHCP	—	6579 = 1614 + 4722 + 243
Active Directory	—	6579 = 1614 + 1000 + 243

rectory, нарушение доступности которых согласно графу $G_{ИТС}$ приводит к блокировке всех остальных ИТ-сервисов КИС.

С использованием алгоритма "Экспертная оценка категорий" и метода анализа иерархий сформированы весовые коэффициенты доступности ИТ-сервисов, соответствующих висячим вершинам графа $G_{ИТС}$, далее определена доступность этих сервисов в условных единицах. Доступность остальных ИТ-сервисов определена согласно (2). Полученные результаты представлены в таблице.

Исходя из полученных оценок получаем, что уровень доступности файлового сервиса — 4722, а контроллера домена — 6579.

Угроза 4 реализуется на телекоммуникационное оборудование (маршрутизатор Cisco 1760), через который предоставляется ИТ-сервис "Предоставление доступа в Интернет". В связи с этим уровень доступности данного маршрутизатора будет равен 243.

Выводы

Предложен метод количественной оценки уровня ущерба от реализации угроз ИБ на корпоративную информационную сеть. Данный метод обладает следующими преимуществами:

- оценка ущерба осуществляется в количественном виде, что позволяет формировать более точные оценки и эффективно использовать метод для решения задач управления рисками ИБ;
- от эксперта не требуется обладания точной информацией о ценности всех активов КИС;
- использование формальной модели КИС позволяет при оценке ущерба учесть взаимодействие активов;
- получение экспертных оценок с привлечением метода анализа иерархий позволяет обеспечивать согласованность ответов и достоверность получаемых результатов.

Недостатком предложенного метода является сложность его применения при значительном числе активов КИС.

Список литературы

1. **Петренко С. А., Симонов С. В.** Управление информационными рисками. Экономически оправданная безопасность. М.: Компания АйТи: ДМК Пресс, 2004. 384 с.
2. **Черешкин Д. С., Кононов А. А., Бурдин О. А.** Комплексная экспертная система АванГард как средство управления рисками нарушения информационной безопасности // Науч.-техн. информ. Сер. 2. 2000. № 12. С. 15—28.
3. **Модели** технических разведок и угроз безопасности информации / Под ред. Е. М. Сухарева. Кн. 3. М.: Радиотехника, 2003. 144 с.
4. **Герасименко В. А.** Организация работ по защите информации в системах электронной обработки данных // Зарубежная радиоэлектроника. 1989. № 12. С. 110—124.
5. **Герасименко В. А., Малюк А. А.** Основы защиты информации. М.: МИФИ, 1997. 537 с.
6. **Хоффман Л. Дж.** Современные методы защиты информации / Пер. под ред. В. А. Герасименко. М.: Сов. радио, 1980. 264 с.
7. **Домарев В. В.** Безопасность информационных технологий. Системный подход. Киев: ООО ТИД Диа Софт, 2004. 992 с.
8. **Risk** management guide for information technology systems // NIST Special Publication 800-30. URL: http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf.
9. **Кононов А. А., Бурдин О. А.** Аксиоматика оценки рисков нарушения информационной безопасности компьютеризированных организационных систем // Проблемы безопасности информации. Компьютерные системы. 2002. № 1. С. 27—30.
10. **Саати Т.** Принятие решение решений. Метод анализа иерархий. М.: Радио и связь, 1993. 278 с.
11. **Аникин И. В.** Модель категорирования ресурсов корпоративной информационной сети // Тез. докл. 6-й ежегодной научно-практической конф. "Инфокоммуникационные технологии глобального информационного общества", Казань, 4—5 сентября 2008. Казань: ООО "Центр оперативной печати". 2008. С. 47—49.

УДК 621.322

А. В. Суханов, канд. техн. наук, доц.,
Санкт-Петербургский государственный
университет информационных технологий,
механики и оптики,
e-mail: AVSihanov@eureca.ru

Количественная оценка свойства защищенности информационных систем

Дается комплексная оценка свойства защищенности информационных систем (ИС): во-первых, на основе методологии общих критериев определяется множество механизмов защиты (МЗ), соответствующее набору функций безопасности объекта оценки (ФБО); во-вторых, формируется рейтинговый показатель защищенности ИС, учитывающий размещение МЗ в структуре системы обеспечения информационной безопасности (СИБ). Рассматривается подход двухступенчатой оптимизации СИБ: с одной стороны, набора ФБО в условиях бюджетных ограничений, с другой стороны, размещения МЗ в структуре СИБ по критерию максимизации рейтингового показателя защищенности ИС.

Ключевые слова: *информационные системы, количественная оценка, критерий максимизации, механизмы защиты, свойство защищенности.*

Основными задачами средств мониторинга безопасности (МБ) информационных систем (ИС) являются:

- оперативная реакция на изменение множеств известных угроз и выявленных уязвимостей ИС;
- регулярное оценивание текущего уровня защищенности ИС для формирования количественного показателя свойства защищенности;
- оптимизация структуры и состава системы обеспечения информационной безопасности (СИБ) информационной системы путем максимизации целевой функции, в качестве которой выступает текущая оценка свойства защищенности.

Актуальность оперативной модификации средств МБ ИС связана с ростом числа сетевых атак, высокой динамикой множества угроз и выявления уязвимостей в распространенном программном обеспечении [1]. Вместе с тем, не менее важна оперативная и достоверная количественная оценка свойства защищенности ИС в целях оптимизации отношения "стоимость/защищенность" в условиях бюджетных ограничений.

Рассмотрим подход двухступенчатой оптимизации СИБ. На первом этапе формируется набор функций безопасности объекта оценки (ФБО), удовлетворяющий оговоренному в техническом задании на проектирование СИБ уровню защищенности ИС и оптимизированный по критерию минимизации стоимости. На втором этапе, исходя из оптимизированного набора ФБО, происходит размещение механизмов защиты (МЗ) в структуре СИБ по критерию максимизации рейтингового показателя защищенности ИС.

На первом этапе используем модель, описывающую поведение хозяйствующего субъекта (ХС) при изменении цен на рынке средств МБ или бюджета безопасности ХС. По аналогии с микроэкономикой [2] в качестве показателя выбрана *эластичность защищенности* — безразмерный коэффициент, отражающий изменение защищенности при изменении цены реализации (эластичность по цене) или при изменении бюджета ИС (эластичность по бюджету).

Одна из задач методологии общих критериев (ОК) [3] — формирование оптимизированного набора ФБО. Выбор набора ФБО равнозначен выбору степени защищенности ИС.

Функция защищенности ИС

Имеется n ранжированных показателей защищенности системы или средств МБ критических информационных систем, или информационные системы для критических сфер применения (КИС). Степень защищенности системы по i-му показателю обозначим x_i, а защищенность по всем показателям — вектором $X = (x_1, x_2, ..., x_n)$.

Пусть ХС выбирает набор степеней защищенности X, используя систему предпочтений, которая описывается скалярной непрерывно дифференцируемой *функцией полезности* $u(X)$. Обозначим затраты на достижение единицы защищенности по i-му показателю p_i (элементарные затраты); набор элементарных затрат по всем показателям — вектором $P = (p_1, p_2, ..., p_n)$. В экономических моделях аналогом элементарных затрат являются рыночные цены.

Возможности выбора набора ФБО ограничены *бюджетом* защищенности, т. е. величиной допустимых расходов I на обеспечение безопасности ИС. Выбор ХС можно описать оптимизационной задачей

$$u(X) \to \max \text{ при } PX = I,\ PX = \sum_{i=1}^{n} p_i x_i. \qquad (1)$$

Решение задачи — выбираемый ХС набор степеней защищенности (оптимальная защищен-

ность) — можно представить *функцией защищенности* от элементарных затрат P и бюджета I:

$$D(P, I) = (D_1(P, I), D_2(P, I), ..., D_n(P, I)) = \\ = \arg\max u(X) \text{ при } PX = I.$$

Каждая из скалярных функций $D_i(P, I)$ — оптимальная защищенность ИС по i-му показателю, зависящая от бюджета и элементарных затрат (цен). Условием экономического равновесия является пропорциональность частных производных функции полезности u_i (предельных полезностей в терминах микроэкономики) ценам для показателей защищенности:

$$u_i = \frac{\partial u}{\partial x_i} = \lambda p_i, \quad i = 1, ..., n. \tag{2}$$

Равенство (2) следует из необходимых условий экстремума в задаче (1), λ — множитель Лагранжа, порождаемый ограничением $PX = I$. Согласно (2) максимальная полезность достигается на таком распределении бюджета защищенности, при котором дополнительная сумма, затраченная на достижение заданной степени защищенности по каждому показателю, приносила бы одинаковую предельную полезность.

Косвенная полезность набора ФБО (по аналогии с микроэкономикой) — полезность набора D при решении задачи (1). Отражает полезность сочетания бюджета и элементарных затрат через *функцию косвенной полезности* $V(P, I)$:

$$V(P, I) = \max\{u(X) \mid PX = I\}, \\ V(P, I) = u(D(P, I)). \tag{3}$$

Если функция полезности $u(X)$ характеризует систему предпочтений ХС, то функция косвенной полезности $V(P, I)$ связана с оптимизирующим процессом выбора ХС (позволяет сопоставить различные комбинации цен и бюджета (P, I) и выяснить, какие из них оптимальны для ХС). В частности, в терминах косвенной полезности легко определить понятия эквивалентного (ΔI_e) и компенсированного (ΔI_c) приращений бюджета, соответствующих изменению цен ΔP:

$$V(P + \Delta P, I) = V(P, I + \Delta I_e); \\ V(P + \Delta P, I + \Delta I_c) = V(P, I).$$

Влияние изменения бюджета и цен на уровень удовлетворения ХС можно определить, почленно дифференцируя равенство (3):

$$\frac{\partial V}{\partial I} = \sum_{i=1}^{n} u_i \frac{\partial D_i}{\partial I}; \tag{4}$$

$$\frac{\partial V}{\partial p_i} = \sum_{i=1}^{n} u_i \frac{\partial D_i}{\partial p_i}. \tag{5}$$

Равенство, выражающее ограничение на величину допустимых расходов

$$\sum_{i=1}^{n} p_i D_i = I, \tag{6}$$

выполняется при любых I и P и его можно дифференцировать:

$$\frac{\partial}{\partial I} \sum_{i=1}^{n} p_i D_i = \sum_{i=1}^{n} p_i \frac{\partial D_i}{\partial I} = 1; \tag{7}$$

$$\frac{\partial}{\partial p_j} \sum_{i=1}^{n} p_i D_j = \sum_{i=1}^{n} p_i \frac{\partial D_i}{\partial p_j} + D_j = 0 \ (j = 1, ..., n)$$

или

$$\sum_{i=1}^{n} p_i \frac{\partial D_i}{\partial p_j} = -Dj \ \ (j = 1, ..., n). \tag{8}$$

Подставляя (2) в (4) и учитывая (7), находим, что

$$\lambda = \frac{\partial V}{\partial I}. \tag{9}$$

Почленно перемножая равенства (8) и (9) и сравнивая с (5), имеем

$$\frac{\partial V}{\partial p_j} = -D_j \frac{\partial V}{\partial I}. \tag{10}$$

Из равенства (10) следует, что если степень защищенности по j-му показателю равна D, а цены на достижение единицы защищенности ИС по этому показателю возросли на $\Delta p_j > 0$, то расходы на достижение защищенности увеличатся, казалось бы, на величину $\Delta p_j D_j$; увеличение бюджета на эту величину скомпенсировало бы ущерб от роста затрат, так что $\Delta I_c = \Delta p_j D_j$.

Однако изменение цен на достижение единицы защищенности ИС по j-му показателю затрагивает соотношение всех цен, и хотя добавка к исходному бюджету компенсирующих затрат $\Delta p_j D_j$ позволит ХС достичь исходной степени защищенности, но теперь такой выбор не оптимален, т. е. ХС в состоянии достичь большей защищенности. Можно утверждать, что $\Delta I_c \leqslant \Delta p_j D_j$.

Таким образом, если учитывать вышеупомянутый эффект, то равенство $\Delta I_c = \Delta p_j D_j$ нарушается для конечных приращений. Но оно, как следует из равенства (10), верно в пределе при $\Delta p_j \to 0$. Действительно,

$$\left.\frac{dI}{dp_j}\right|_{V = \text{const}} = D_j,$$

так что в дифференциальной форме $dI_c = D_j dp_j$. Аналогично, $dI_e = -D_j dp_j$.

Функция замещения

Под *замещением* подразумевается такое изменение степеней защищенности ИС по различным показателям (изменение набора ФБО), при котором общая полезность набора степеней защищенности не изменится. Применительно к изменению выбора ХС при изменении цен замещение выступает как средство анализа.

Для формального описания замещения рассмотрим оптимизационную задачу, сопряженную к задаче (1): нахождение набора степеней защищенности, требующего минимального бюджета для достижения заданного уровня полезности U:

$$PX \to \min \quad \text{при} \quad u(X) = U. \tag{11}$$

Решение задачи — набор степеней защищенности, рассматриваемый в зависимости от набора цен P и от заданного уровня полезности U — будем называть *функцией замещения* $Q(P, U)$:

$$Q(P, U) = (Q_1(P, U), Q_2(P, U), ..., Q_n(P, U)).$$

Изменение любой из цен p_j приводит к изменению всех Q_i, степень этой изменчивости можно назвать *эффектом замены*. В дифференциальной форме эффект замены выражается величиной $\frac{dQ_i}{dp_j}$.

Уравнение выбора

Пусть ХС, имеющий бюджет I, при ценах P выбрал набор $D(P, I)$ и получил полезность $U = V(P, I)$. Он не мог бы достичь той же полезности меньшими затратами, так что набор $D(P, I)$ — решение задачи замещения при данных ценах и уровне полезности. Поэтому имеет место тождество

$$D(P, I) = Q(P, V(P, I)). \tag{12}$$

Дифференцируя почленно тождество (12) по p_j, получим

$$\frac{\partial D_i}{\partial p_j} = \frac{\partial Q_i}{\partial p_j} + \frac{\partial Q_i}{\partial U}\frac{\partial V}{\partial p_j}$$

или, с учетом (10),

$$\frac{\partial D_i}{\partial p_j} = \frac{\partial Q_i}{\partial p_j} - \frac{\partial Q_i}{\partial U}\frac{\partial V}{\partial I} D_j. \tag{13}$$

Вместе с тем, дифференцируя (12) по I, получим:

$$\frac{\partial D_i}{\partial I} = \frac{\partial Q_i}{\partial U}\frac{\partial V}{\partial I}.$$

Сопоставляя результат с (13), получим *уравнение выбора*:

$$\frac{\partial D_i}{\partial p_j} = \frac{\partial Q_i}{\partial p_j} - \frac{\partial D_i}{\partial I} D_j.$$

Уравнение описывает изменение выбора ХС при изменении цен: первое слагаемое в правой части — *эффект замены*, второе — *эффект бюджета*.

Заметим, что вектор X выбора ХС можно рассматривать и как значение функции защищенности, и как значение функции замещения:

$$X = D(P, I) = Q(P, U), \text{ где } I = PX, \ U = u(X).$$

В тех случаях, когда мы будем оперировать величинами степеней защищенности по различным показателям, а не их зависимостями от переменных, нам будет безразлично, считать ли их значениями функции D или Q; мы будем их обозначать через x. В этих обозначениях уравнение выбора имеет вид

$$\frac{\partial D_i}{\partial p_j} = \frac{\partial Q_i}{\partial p_j} - \frac{\partial D_i}{\partial I} x_j. \tag{14}$$

В частности, при $i = j$ равенство (14) описывает изменение защищенности по некоторому показателю при изменении цен по этому показателю:

$$\frac{\partial D_i}{\partial p_i} = \frac{\partial Q_i}{\partial p_i} - \frac{\partial D_i}{\partial I} x_i.$$

Уравнение выбора может быть записано так же, как соотношение между эластичностями защищенности. Это понятие (по аналогии с микроэкономическими моделями) отражает степень чувствительности защищенности по i-му показателю к изменению цен по j-му показателю. В качестве меры изменения можно выбрать коэффициент эластичности, который определяется как отношение относительного изменения защищенности $\Delta D/D$ к относительному изменению цены $\Delta p/p$, или в пределе — произведение частной производной $\frac{\partial D}{\partial p}$ и отношения $\frac{p}{d}$.

Умножим обе части равенства (14) на $\frac{p_j}{x_i}$:

$$\frac{\partial D_i}{\partial p_j}\frac{p_j}{x_i} = \frac{\partial Q_i}{\partial p_j}\frac{p_j}{x_i} - \frac{\partial D_i}{\partial I}\frac{I}{x_i}\frac{p_j x_j}{I}.$$

Выражение в левой части — *эластичность защищенности* по i-му показателю (D_i) по цене j-го, *прямая* эластичность защищенности — при $i = j$, *перекрестная* — при $i \neq j$:

$$e_{ij} = \frac{\partial D_i}{\partial p_j}\frac{p_j}{x_i}.$$

Первое слагаемое в правой части — эластичность компенсированной защищенности по i-му показателю (Q_i) по цене j-го (при фиксированной полезности):

$$\tilde{e}_{ij} = \frac{\partial Q_i}{\partial p_j}\frac{p_j}{x_i}.$$

Выражение $E_i = \frac{\partial D_i}{\partial I}\frac{I}{x_i}$ — эластичность защищенности по бюджету, а $w_j = \frac{p_j x_j}{I}$ — доля j-го показателя защищенности в расходах. Используя эти обозначения, выразим уравнение выбора в терминах эластичностей защищенности:

$$e_{ij} = \tilde{e}_{ij} - w_j E_i. \qquad (15)$$

Перекрестный эффект замены

С задачей оптимального выбора при ограниченном бюджете связана функция $V(P, I)$, характеризующая полезность бюджета I при заданных ценах. Подобно этому, с задачей (11) связана функция

$$y(P, U) = \min\{PX \mid u(X) = U\},$$

которая является скалярным произведением P на $Q(P, U)$:

$$y(P, U) = PQ(P, U). \qquad (16)$$

Функция $y(P, U)$ характеризует минимальный бюджет, т. е. расходы ХС, минимально необходимые для достижения уровня полезности U при заданных ценах. Будем называть ее *функцией расходов*.

Равенство (16) можно почленно дифференцировать по p_j:

$$\frac{\partial y}{\partial p_j} = \sum_{i=1}^{n} p_i \frac{\partial Q_i}{\partial p_j} + Q_j. \qquad (17)$$

Так как задача (11) является сопряженной к задаче (1), ее решение удовлетворяет условию (2): цены по всем показателям защищенности пропорциональны частным производным u_i (предельным полезностям). Поэтому

$$\sum_{i=1}^{n} p_i \frac{\partial Q_i}{\partial p_j} = \frac{1}{\lambda}\sum_{i=1}^{n} u_i \frac{\partial Q_i}{\partial p_j}.$$

Заметим, что сумма в правой части — полная производная полезности dU/dp при замещении. Но при замещении по определению $U = \text{const}$, так что $dU/dp_j = 0$. Итак, сумма в правой части выражения (17) равна нулю, так что

$$\frac{\partial y}{\partial p_j} = Q_j.$$

Продифференцируем полученное равенство по цене i-го показателя:

$$\frac{\partial^2 y}{\partial p_i \partial p_j} = \frac{\partial Q_j}{\partial p_i}.$$

Но $\frac{\partial^2 y}{\partial p_i \partial p_j} = \frac{\partial^2 y}{\partial p_j \partial p_i}$, так что

$$\frac{\partial Q_j}{\partial p_i} = \frac{\partial Q_i}{\partial p_j}. \qquad (18)$$

Перекрестные эффекты замещения степени защищенности по i-му показателю по цене j-го и степени защищенности по j-му показателю по цене i-го равны друг другу. Это обстоятельство существенно для классификации взаимозависимости степеней защищенности по различным показателям. Если показатели защищенности являются взаимными заменителями, то перекрестный эффект замены положителен. При этом безразлично, идет ли речь об эффекте замены степени защищенности по i-му показателю по цене j-го или, наоборот, они равны друг другу. То же относится и к взаимодополняющим показателям — там обе характеристики отрицательны. Таким образом, перекрестные эффекты замещения являются характеристиками взаимной зависимости показателей.

Рассмотрим, как соотносятся эластичности защищенности. Из определения эластичности компенсированной защищенности следует (равенство перед (15))

$$\frac{\partial Q_i}{\partial p_j} = \tilde{e}_{ij}\frac{x_j}{p_i}.$$

Равенство (18) позволяет утверждать, что

$$\tilde{e}_{ij}\frac{x_i}{p_j} = \tilde{e}_{ij}\frac{x_j}{p_i} \quad \text{или} \quad \tilde{e}_{ij}\frac{1}{p_j x_j} = \tilde{e}_{ji}\frac{1}{p_i x_i}.$$

Умножая обе части на I, получим

$$\frac{\tilde{e}_{ij}}{w_j} = \frac{\tilde{e}_{ji}}{w_i}. \qquad (19)$$

Поскольку перекрестные эластичности компенсированной защищенности отличаются от соответствующих эффектов замены положительными множителями, величины $\tilde{e}_{ij}$ и $\tilde{e}_{ji}$ также совпадают по знаку.

Чтобы установить связь между "обычными" перекрестными эластичностями защищенности (при фиксированном бюджете), преобразуем (15):

$$\frac{\tilde{e}_{ij}}{w_j} = \frac{e_{ij}}{w_j} + E_i.$$

Теперь из равенства (19) можно получить:

$$\frac{e_{ij}}{w_j} + E_i = \frac{e_{ji}}{w_i} + E_j.$$

Таким образом, из уравнения выбора следует взаимная сопряженность перекрестных эластичностей защищенности по двум показателям: по ценам и их эластичностей по бюджету. Влияние дополнительных слагаемых — эластичностей по бюджету — приводит к тому, что величины e_{ij} и e_{ji} могут различаться не только абсолютной величиной, но и знаком.

На втором этапе воспользуемся моделью адаптивной защиты [4] для оптимизации размещения МЗ по структуре СИБ по критерию максимизации показателей свойства защищенности ИС. Применение модели, основанной на принципе биологической аналогии [5], позволяет [6]:

- обеспечить близкое к оптимальному соотношение "стоимость/эффективность" средств МБ за счет постепенного наполнения модели СИБ только необходимыми механизмами защиты;
- в динамике отслеживать наиболее задействованные механизмы защиты при изменении множества угроз;
- формировать спецификацию требований на отсутствующие МЗ;
- оценивать защищенность ИС через относительный ущерб и показатели активности МЗ, распределенных по структуре средств МБ.

Количественные показатели защищенности ИС

Рассмотрим методику формирования рейтинговых показателей защищенности ИС, в которой учитывается как величина *предотвращенного ущерба* ИС, так и *экспертные оценки*, сопоставляющие, во-первых, ущерб с множеством угроз, во-вторых, размер ущерба с местом реализации угрозы в структуре ИС.

1. В соответствии с заданием на проектирование СИБ используют множество МЗ, соответствующее набору ФБО, сформированному на первом этапе.

2. Разрабатывают модель распределения МЗ по иерархии СИБ.

3. Эксперты предметной области формируют оценки: *достоверности активации* МЗ для нейтрализации известных угроз, *частоты активации* угроз, *ожидаемого ущерба* от реализации угрозы (например, по отношению к допустимой для ХС величине.

4. Экспертные оценки для достоверности активации МЗ, частоты активации угроз и ожидаемого ущерба представляют матрицами "МЗ-угрозы", "угрозы—эшелоны".

5. Экспертные оценки в виде системы нечетких предикатных правил отображают в структуре нечетких нейронных сетей (НС) и *автоматически корректируют* в процессе адаптации на подмножестве известных угроз.

6. Для матриц "МЗ—угрозы", "угрозы—эшелоны" формируют *показатели значимости* по строкам и столбцам, которые определяют уровень активности механизмов защиты и эшелонов СИБ при нейтрализации угроз.

7. Интегральные оценки защищенности получают путем умножения матриц "МЗ—угрозы" и "угрозы—эшелоны" для формирования матрицы "МЗ—эшелоны" — матрицы *ME* достоверности активации МЗ по эшелонам СИБ

$$ME_{m \times n} = \begin{pmatrix} me_{11} & me_{12} & \vdots & me_{1n} \\ me_{21} & me_{22} & \vdots & me_{2n} \\ \cdots & \cdots & \vdots & \cdots \\ me_{m1} & me_{m2} & \vdots & me_{mn} \end{pmatrix},$$

где m — число механизмов защиты; n — число эшелонов СИБ.

Аналогично получают матрицы "эшелоны—МЗ" для частоты активации угроз и относительного ожидаемого ущерба, поэлементное умножение которых формирует матрицу ожидаемого ущерба за заданный интервал времени.

8. Для матриц "МЗ—эшелоны" и "эшелоны—МЗ" формируют показатели значимости по строкам и столбцам, которые характеризуют *активность* использования МЗ либо *эшелона* СИБ, а также ожидаемый ущерб в разрезе МЗ и эшелонов СИБ. Показатели значимости оценивают, например, по формуле

$$x_i = \sqrt{\sum_{j=1}^{n} me_{ij}^2}, \quad i = 1, \ldots, m.$$

Сопоставление показателей значимости позволяет в пределах строки выявить наиболее задействованные эшелоны (в пределах столбца — МЗ) и обосновать целесообразность использования механизма защиты в составе эшелона СИБ.

9. Умножение матриц "МЗ—эшелоны" и "эшелоны—МЗ" позволяют обобщить в главной диагонали квадратной матрицы (матрицы *MM* достоверного ущерба "МЗ—МЗ") показатели достоверности активации МЗ и ущерба от реализации атаки, а умножение матриц "эшелоны—МЗ" и "МЗ—эшелоны" — матрицу *EE* достоверного ущерба "эшелоны—эшелоны".

Обобщающими показателями являются векторы:

$P_{1xm} = (p_1, p_2, \ldots, p_m)$ — *вектор достоверного распределения ущерба по МЗ*, $p_i = mm_{ij}$, $i = j = 1, \ldots, m$, и $D_{1xn} = (d_1, d_2, \ldots, d_n)$ — *вектор*

достоверного распределения ущерба по эшелонам системы ИБ, $d_i = ee_{ij}$, $i = j = 1, ..., n$.

10. Интегральной оценкой защищенности ИС в разрезе МЗ является рейтинговый показатель R_M — модуль вектора P_{1xm}, а в разрезе эшелонов СИБ — показатель R_E — модуль вектора D_{1xn}:

$$R_M = |P_{1xm}| = \sqrt{\sum_{j=1}^{n} p_i^2}, \quad i = 1, ..., m;$$

$$R_E = |D_{1xn}| = \sqrt{\sum_{j=1}^{n} d_i^2}, \quad i = 1, ..., n.$$

Рейтинговые показатели используются методикой оптимизации СИБ для такого размещения МЗ в структуре системы защиты ИС, которое максимизирует эффективность расположения в структуре ИС средств реализации набора ФБО, оптимизированного на первом этапе.

Список литературы

1. **Котенко И. В., Степашкин М. В.** Интеллектуальная система моделирования атак на web-сервер для анализа уязвимостей компьютерных систем // Сб. докл. VI Международной конф. по мягким вычислениям и измерениям SCM'2003. СПб.: СПГЭТУ, 2003. Т. 1. С. 298—301.

2. **Гальперин В. М., Игнатьев С. М., Моргунов В. И.** Микроэкономика. В 2-х т. СПб.: Экономическая школа, 1994. Т. 1.

3. **РД** Безопасность информационных технологий. Общая методология оценки безопасности информационных технологий (проект). ФСТЭК России, 2005.

4. **Нестерук Г. Ф., Нестерук Ф. Г., Осовецкий Л. Г., Фахрутдинов Р. Ш.** К разработке модели адаптивной защиты информации // Специальная техника. 2005. № 2. С. 52—58.

5. **Осовецкий Л. Г., Нестерук Г. Ф., Бормотов В. М.** К вопросу иммунологии сложных информационных систем // Изв. вузов. Приборостроение. 2003. Т. 46. № 7. С. 34—40.

6. **Нестерук Г. Ф., Осовецкий Л. Г., Нестерук Ф. Г.** К оценке защищенности систем информационных технологий // Перспективные информационные технологии и интеллектуальные системы. 2004. № 1. С. 31—41.

УДК 004.322.067

М. В. Тарасюк, канд. техн. наук, доц.,
СПбГУ ИТМО,
Ю. С. Исаченко, системный аналитик,
ОАО "Институт сетевых технологий",
г. Санкт-Петербург, e-mail: miket@int.spb.ru

Подавление модуляции длин дейтаграмм в сетях шифрованной связи

Рассматриваются методы противодействия скрытым каналам утечки информации между защищенной и незащищенной частью пакетной сети шифрованной связи, вызванные возможностью нарушителя формировать IP-дейтаграммы произвольной длины. Рассматриваемые методы предполагают внесение в передаваемый поток аддитивных или мультипликативных помех, сформированных по случайному закону.

Ключевые слова: *скрытый канал, защита информации, утечка информации, маскирование трафика, криптомаршрутизатор*

Преимущества сетей пакетной коммутации, такие, как например, высокая эффективность использования ресурсов связи, снижение стоимости оборудования для построения сетей, хорошо известны. Однако есть и обратная сторона — построение криптографически защищенных сетей (с использованием криптомаршрутизаторов) не защищает от угрозы возникновения скрытых каналов (СК), позволяющих незаметно для пользователя передавать информацию, хранящуюся на его персональной электронной вычислительной машине (ПЭВМ), в незащищенную часть сети. Такие каналы утечки информации из защищенной в незащищенную часть сети появляются ввиду того, что параметры трафика защищенной сети (тип используемых протоколов, номера виртуальных каналов, адреса, классы приоритетности и пр.) доступны:

— для манипуляции в сети защищаемого объекта (программными закладками);

— для анализа в незащищенной части сети (в каналах связи и узлах транспортной сети).

Для формирования канала утечки информации достаточно внедрить в защищенную сеть объекта троянскую программу, которая находит нужные данные и обеспечивает их последовательное отображение на параметры трафика. Анализируя параметры трафика в открытой сети, нарушитель восстанавливает переданные данные при необходимости, воспользовавшись методами помехоустойчивого кодирования для исправления ошибок.

Для противодействия утечке информации можно применять туннельное шифрование, когда передаваемые из локальной сети *IP*-дейтаграммы целиком (вместе со всеми служебными данными) зашифровываются (на ключе парной связи, зависящем от параметров заголовка, например от адреса и номера порта). Зашифрованная дейтаграмма целиком вставляется во внешнюю дейтаграмму, которая снабжается параметрами заголовка для доведения через транспортную (незащищен-

ную) сеть. Таким образом, поля исходной дейтаграммы становятся недоступны для анализа в незащищенной части сети. Однако у нарушителя остается возможность использовать скрытый канал утечки за счет манипуляции интервалов времени между отправкой *IP*-дейтаграмм, длин *IP*-дейтаграмм (модель угрозы проиллюстрирована на рис. 1, см. четвертую сторону обложки), а также параметров управления коммутационным оборудованием открытой сети (для мультисервисных сетей, где требуется задание приоритетов обслуживания).

Некоторые методы генерации ложного трафика для противодействия СК модуляции длительности междейтаграммных интервалов для сетей, требующих малых задержек обработки *IP*-дейтаграмм в коммутационном оборудовании, изложены в работах [1], [2]. Для сетей, в которых некритичны дополнительные временные задержки, защита от модуляции междейтаграммных интервалов более эффективно блокируется за счет выравнивающей буферизации трафика.

Противодействие СК, использующим модуляцию длин *IP*-дейтаграмм, как и в случае каналов модуляции междейтаграммных интервалов, может быть обеспечено режимными мерами (затрудняющими внедрение в защищаемую сеть деструктивных программных функций), либо за счет включения в *IP*-дейтаграммы служебных данных, выравнивающих длины всех *IP*-дейтаграмм до максимальной длины. На практике режимные меры не всегда приемлемы, поскольку предполагают использование в составе защищенной сети только доверенного программного обеспечения (например, имеющего сертификат безопасности), значительно ограничивают технологические режимы обработки информации (например, исключается или дополнительно авторизуется использование внешних устройств — для исключения внедрения в объекты вредоносных программ на этапе эксплуатации сети).

Выравнивание *IP*-дейтаграмм до максимальной длины может привести к значительному росту служебного трафика и снижению эффективности использования каналов. В особенности это справедливо для мультисервисных сетей, в которых максимальная длина *IP*-дейтаграммы может значительно превышать среднюю длину (для *IP*-дейтаграмм, подлежащих шифрованию на одном ключе). На рис. 2 показана гистограмма типичного распределения длин *IP*-дейтаграмм для трафика протоколов SIP и HTTP. Вносимая избыточность при выравнивании *IP*-дейтаграмм до максимальной длины составит порядка 190 и 90 % соответственно.

Как и для подавления модуляции междейтаграммных интервалов, необходимы методы, обеспечивающие значительное снижение служебного трафика за счет приемлемого ухудшения уровня защищенности информации. Для обоснования применимости таких методов необходимо исходить из каких-либо общепринятых подходов для задания численных норм защиты. В работе [4] для обоснования уровня защищенности от СК, аналогично техническим каналам утечки информации за счет побочных электромагнитных излучений и наводок (ПЭМИН), предлагается исходить из нормирования отношения сигнал/шум на границе контролируемой зоны защищенной сети. Идея состоит в том, что отношение сигнал/шум может быть пересчитано в эквивалентную пропускную способность СК (C), исходя из которой, в свою очередь, оценивается уровень маскирующей СК помехи.

Для рассматриваемого случая маскирования длин IP-дейтаграмм маскировка СК может обеспечиваться тремя путями:

- включением в исходную *IP*-дейтаграмму случайного числа маскировочных байтов;
- включением в поток передаваемых данных служебных (маскировочных) *IP*-дейтаграмм;
- склеиванием двух или более *IP*-дейтаграмм в одну *IP*-дейтаграмму.

Последний метод предполагает наличие буфера и, соответственно, внесение задержки. При наличии трафика, критичного к временным задержкам (трафика реального времени), задержка, вносимая

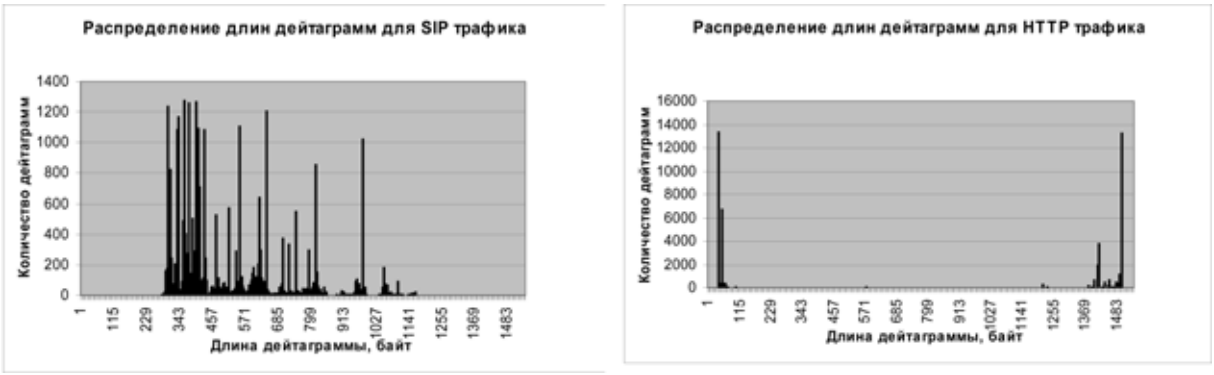


Рис. 2. Гистограммы распределения длин *IP*-дейтаграмм, передающих сообщения протоколов SIP и HTTP

буфером, и, следовательно, его размер должны нормироваться. Переполнение буфера, его заполнение с интервалом времени, превышающим нормы на задержку, либо использование длин *IP*-дейтаграмм со средним размером, превышающим MTU/2, приводят к отсутствию зашумления СК. Использование данного метода без внесения маскировочных байтов или *IP*-дейтаграмм не может обеспечить гарантированную защиту.

Для метода, предполагающего добавление в *IP*-дейтаграмму дополнительных байтов, очевидно, речь идет о внесении аддитивной помехи в канал. Для оценки пропускной способности при наихудшем распределении помехи удобно воспользоваться моделью дискретного канала с бесконечным числом символов и непрерывным временем, имеющего некоррелированное и одинаковое распределение числа пользовательских и маскировочных байтов в *IP*-дейтаграмме.

Для такого канала, согласно работе [5], удельную пропускную способность на один интервал модуляции (в нашем случае на одну *IP*-дейтаграмму) можно оценить следующим образом:

$$C = \frac{1}{2}\log_2\left(1 + \frac{d}{d_N}\right),$$

где d_N и d — дисперсия значений помехи (числа маскировочных байтов) и сигнала (длина *IP*-дейтаграммы). При этом для рассматриваемого СК дисперсия числа маскировочных байтов вычисляется следующим образом:

$$d_N = \frac{d}{2^{2C} - 1},$$

где C — допустимая норма утечки информации по СК на один интервал модуляции. Если формирование маскировочных байтов распределено по закону равномерной плотности на интервале [0; E], параметр E полностью определяет данную функцию, в том числе, через параметр E может быть выражена дисперсия числа маскировочных байтов ($d_N = E^2/12$). Соответственно, для вычисления параметра E используется соотношение

$$E = \sqrt{\frac{12d}{2^{2C} - 1}}.$$

Для второго варианта противодействия (передача маскировочных *IP*-дейтаграмм) в канал вносится мультипликативная помеха, нарушающая синхронизацию отдельных символов СК. *IP*-дейтаграммы, несущие символы СК, окажутся в данном случае перемешанными с маскировочными *IP*-дейтаграммами. Оценка пропускной способности рассматриваемого СК требует иного подхода, нежели использованная выше для канала с аддитивной помехой. Предлагается исходить из следующих допущений.

1. Передаваемые данные кодируются канальными символами $\{C_I\}$, сопоставляемыми с длиной *IP*-дейтаграммы. Для сегмента локальной сети (ЛС) с длиной *IP*-дейтаграммы $L = 1500$ байт можно положить, что на каждую *IP*-дейтаграмму (канальный символ) приходится 10 бит информации.

2. *IP*-дейтаграммы с данными передаются непрерывно. Вероятность вставки в кодирующий кадр канала W и более *IP*-дейтаграмм подряд пренебрежимо мала. При возникновении очередей на интерфейсах первым передается ранее поступившая *IP*-дейтаграмма (FIFO).

3. Для выявления ошибок в передаваемых данных канальные символы группируются в кадры СК, имеющие общую длину N символов, из которых M символов содержат проверочную информацию, сформированную по известному алгоритму обнаружения ошибок.

4. Цикл передачи данных СК сводится к формированию последовательности *IP*-дейтаграмм, содержащих символы кадра (из которых M символов являются проверочные). При передаче по сети *IP*-дейтаграмм, содержащих символы кадра СК, возможна вставка маскировочных *IP*-дейтаграмм, приводящих к искажению последовательности символов кадра.

5. Задано пуассоновское распределение для числа маскировочных *IP*-дейтаграмм с параметром потока A. Для простоты рассматривается случай, когда единицей измерения времени выступает длительность передачи одной *IP*-дейтаграммы. Процесс внесения в поток данных помехи (маскировочных *IP*-дейтаграмм) при этом представляет собой равномерное "размазывание" *IP*-дейтаграмм в пределах окна декодирования размером W. Среднее число маскировочных *IP*-дейтаграмм, обрабатываемых на интервале передачи кадра, пропорционально AW. Время передачи кадра определяется параметрами L, N и пропускной способностью канала связи.

6. Получатель данных СК (находящийся за пределами контролируемой зоны объекта разведки) использует окно декодирования кадра ($W > N$, $W = N$ при отсутствии маскировочных *IP*-дейтаграмм) для исключения из принимаемой последовательности маскировочных *IP*-дейтаграмм. Пример циклограммы передачи кадра данных СК приведен на рис. 3 ($N = 5$).

7. Для исключения из принимаемой последовательности маскировочных *IP*-дейтаграмм используется следующая процедура.

I. Начиная с первого (после предшествующего, целиком обработанного кадра) канального символа, получатель (нарушитель) накапливает в приемном буфере N канальных символов. Если кон-

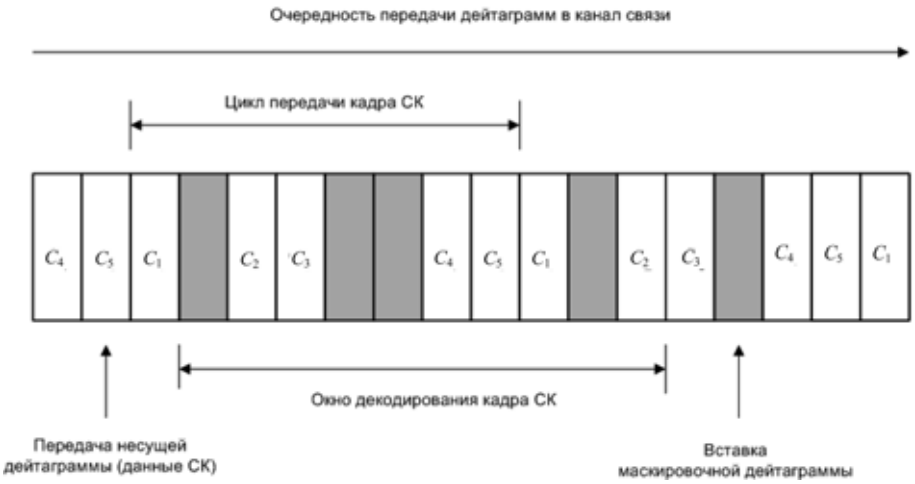


Рис. 3. Иллюстрация работы алгоритма модуляции

трольная информация (M символов) корректна, маскировочные *IP*-дейтаграммы отсутствуют, информация из текущего кадра передается получателю, а принимающий буфер очищается. Осуществляется переход к шагу I.

II. В противном случае принимаются $N + 1$, $N + 2$ и так далее, но не более W *IP*-дейтаграмм. Для каждого из значений $N + I$, сохраненных в принимающем буфере, выполняется поиск N символов, для которых выполняются проверочные соотношения для контрольной информации кода. В первую очередь анализируются *IP*-дейтаграммы, которые первыми помещены в принимающий буфер. Если для $N + I$ *IP*-дейтаграмм в принимающем буфере проверочные соотношения оказываются выполненными, информация из текущего кадра передается получателю, а из принимающего буфера удаляется $N + I$ *IP*-дейтаграмм. Выполняется переход к шагу I.

III. Если для очередной принятой *IP*-дейтаграммы выполнено соотношение $N + I > W$ из буфера приемника удаляется самая старая *IP*-дейтаграмма. Выполняется переход к шагу I.

Выше полагалось, что возникновение ошибки (когда на этапе работы декодера в качестве выходной последовательности кадра выбирается множество *IP*-дейтаграмм, которое содержит одну или более маскировочных *IP*-дейтаграмм, и контрольная сумма при этом является правильной) приводит к автоматическому отбрасыванию всех *IP*-дейтаграмм, интерпретированных, как верно принятый кадр СК. В рамках процедуры декодирования последующих кадров данные *IP*-дейтаграммы не учитываются.

Параметры наилучших схем модуляции

A	W	N	M	C	S
0,025	251	239	54	7,644777	$9{,}99 \cdot 10^{19}$
0,05	168	154	54	6,330497	$9{,}38 \cdot 10^{19}$
0,075	145	130	54	5,34288	$9{,}52 \cdot 10^{19}$
0,1	128	112	54	4,399377	$9{,}33 \cdot 10^{19}$
0,125	115	98	54	3,589246	$8{,}72 \cdot 10^{19}$
0,15	106	88	54	2,900034	$9{,}65 \cdot 10^{19}$
0,175	98	79	54	2,291177	$8{,}66 \cdot 10^{19}$
0,2	92	72	54	1,745331	$8{,}35 \cdot 10^{19}$
0,225	88	67	54	1,281622	$9{,}95 \cdot 10^{19}$
0,25	88	67	54	0,915126	$9{,}95 \cdot 10^{19}$
0,275	84	62	54	0,598704	$9{,}37 \cdot 10^{19}$
0,3	84	62	54	0,391815	$9{,}37 \cdot 10^{19}$
0,325	84	62	54	0,232486	$9{,}37 \cdot 10^{19}$
0,35	84	62	54	0,125861	$9{,}37 \cdot 10^{19}$
0,375	81	58	54	0,070633	$9{,}54 \cdot 10^{19}$

Рассмотренная модель оперирует тремя показателями эффективности, оптимальное сочетание которых как с точки зрения нарушителя, так и с точки зрения системы защиты информации, составляет в реальной системе достаточно сложную задачу. К числу основных характеристик рассмотренной схемы модуляции относятся: пропускная способность C, сложность декодера (суммарная трудоемкость декодера S), вероятность ошибки кадра P. Данные показатели вычисляются по следующим соотношениям:

$$\left.\begin{aligned} C &= \sum_{k=0}^{W-N} \frac{(A \cdot W)^k \mathbf{e}^{-A \cdot W}\left(1 - \frac{M}{N}\right)\log L}{k!}; \\ S &= \frac{W!}{N!(W-N)!}; \\ P &= 1 - [1 - 2^{-M}]^S. \end{aligned}\right\} \quad (1)$$

Для нахождения параметров алгоритма маскировки в качестве основного показателя выберем пропускную способность СК. Будем предполагать, что для рассматриваемого СК допустимым является $C \leqslant 0{,}1$ бит на дейтаграмму.

На остальные показатели наложим следующие ограничения: $P \leqslant 10^{-6}$, $S \leqslant 10^{20}$, длина *IP*-дейтаграмм $L = 1500$ байт (работа в локальной сети). Ограничение $S \leqslant 10^{20}$ предлагается исходя из предельных возможностей нарушителя по прямому подбору ключа криптоалгоритма с 56-битовым ключом (DES).

Расчетные процедуры по оптимизации схемы модуляции, выполненные с использованием формул (1) с учетом заданных ограничений, позволили получить результаты, показанные в таблице.

В приведенной таблице для простоты не указаны значения для показателя P, поскольку для всех схем он принимает очень малые значения.

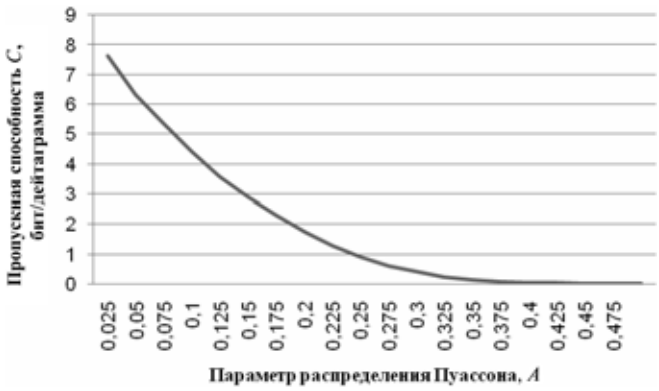


Рис. 4. Зависимость пропускной способности от параметра *A*

Рис. 5. Требуемый запас устойчивости передачи данных по СК

Выборка параметров декодера СК, полученная оптимизацией пропускной способности и с учетом наложенных ограничений, указывает, что параметр M для $S \leqslant 10^{20}$ имеет значение 54. При анализе параметров декодера СК значение $M = 54$ является граничным, при котором ошибка данных становится величиной крайне малой (более низкого порядка, чем 2^{-1000}). Максимальное значение ошибки $P = 8 \cdot 10^{-7}$ достигается для значения параметра потока $A = 0{,}025$.

Зависимость пропускной способности C от параметра потока A для найденных схем модуляции проиллюстрирована графиками на рис. 4.

Разница между размером окна W и числом *IP*-дейтаграмм N, с использованием которых передается кадр СК (необходимый запас устойчивости обмена в условиях флюктуации трафика), увеличивается по мере роста параметра потока. Зависимость приведена на рис. 5.

Для сравнения представленных методов нужно отметить, что средняя избыточность, вносимая добавлением маскировочных байтов в каждую *IP*-дейтаграмму из расчета, что допустимая норма утечки информации по СК на один интервал модуляции $C = 0{,}1$, приблизительно составит 174 байта. Для достижения тех же значений защищенности за счет использования маскировочных *IP*-дейтаграмм необходимо, чтобы число маскировочных *IP*-дейтаграмм X удовлетворяло условию $X = W - N + 1$. При этом W и N берутся для случая, когда пропускная способность становится равной или ниже допустимой нормы утечки, т. е., когда при добавлении маскировочных *IP*-дейтаграмм в общий поток данных произошло превышение размера установленного значения окна.

Как результат, в поток трафика необходимо добавлять 24 (маскировочные) *IP*-дейтаграммы на каждые 84 переданные *IP*-дейтаграммы. Средняя избыточность на *IP*-дейтаграмму составит при этом 214 байт. Применение метода добавления маскировочных байтов в результате дает меньшую среднюю избыточность на *IP*-дейтаграмму в сравнении с методом добавления маскировочных *IP*-дейтаграмм.

В целом, оценка избыточности должна выполняться для определенных видов трафика открытой сети. Для некоторых случаев метод добавления маскировочных *IP*-дейтаграмм может оказаться более рациональным, в том числе и по причине более простой реализации. Кроме того, нужно учитывать, что устранение СК, использующих модуляцию междейтаграммных интервалов (для сетей с жесткими требованиями к задержкам обработки в узлах транспортных сетей), все равно потребует передачи маскировочных *IP*-дейтаграмм согласно общим подходам, описанным в работе [1].

Список литературы

1. **Тарасов И. В., Тарасюк М. В.** Адаптивная маскировка скрытых каналов в открытых системах с многоуровневым доступом // Безопасность информационных технологий. 2004. № 2.

2. **Князев Е. Г.** Скрытая передача данных на основе адаптивного дополнения трафика виртуальной корпоративной сети // Безопасность информационных технологий. 2004. № 4.

3. **Kang M. H., Moskowitz I. S.** A Pump for rapid, reliable, secure communication // Proc. of the 1st ACM Conference on computer and communications security. Fairfax A. ACMPress. 1993. Nov 3—5. P. 119—129.

4. **Тарасов И. В., Тарасюк М. В.** Выбор норм эффективности защиты от утечки информации по скрытым каналам модуляции трафика в средствах сетевого шифрования // Информационные технологии. 2006. № 8. С. 16—19.

5. **Назаров М. В., Кувшинов Б. И., Попов О. В.** Теория передачи сигналов. М.: Связь, 1970.

УДК 001.92:004.89

И. П. Норенков, д-р техн. наук, проф.,
зав. каф. МГТУ им. Н. Э. Баумана,
e-mail: norenkov@wwwcdl.bmstu.ru

Интеллектуальные технологии на базе онтологий

Дан обзор задач и технологий онтологического инжиниринга, включая задачи поиска информации, кластеризации, классификации, создания электронных образовательных ресурсов и других с использованием онтологий. Выделена задача поиска проектных решений на основе семантических сетей паттернов проектирования.

Ключевые слова: *онтология, база знаний, интеллектуальная технология, принятие проектных решений*

Введение

Интеллектуальные системы применяют во многих предметных областях для решения сложных слабоструктурированных проблем. Это системы, оперирующие знаниями. Обычно под знаниями подразумевают представление субъекта о явлениях и закономерностях внешнего, по отношению к субъекту, мира [1]. В информатике знаниями называют информацию, которая в отличие от данных характеризуется внутренней интерпретируемостью, структурированностью, связностью, а к интеллектуальным относят системы, проявляющие способность к целенаправленному поведению [2].

Основные цели применения интеллектуальных систем — сокращение сроков принятия решений или повышение их качества, или и то, и другое. Среди решаемых проблем фигурируют поиск информации, аннотирование документов, подготовка отчетов, формирование учебных пособий, извлечение данных из документов, поддержка принятия решений при проектировании и управлении процессами и др.

Для решения проблем получения, анализа, обработки и представления знаний используют ряд типов интеллектуальных систем — это системы экспертные, информационно-поисковые, поддержки принятия решений, оперативной аналитической обработки данных, управления знаниями и др. Знания представляют в виде таксономий, онтологий, семантических сетей, продукций, хромосом, фреймов, паттернов. Системы, предназначенные для сбора, хранения, поиска и выдачи знаний, называют базами знаний, т. е. база знаний (БЗ) — систематизированное хранилище неструктурированной информации [1].

О создании глобальной базы знаний ("Галактическая сеть") и интерактивных компьютерных вычислений еще на заре компьютерной эры писал Д. Ликлайдер — разработчик сети *Arpanet*, являющейся прообразом Интернет. Далее грандиозный, но также неосуществленный, проект глобальной гипертекстовой сети письменных документов *Xanadu* был предложен Т. Нельсоном [3]. Подобные проекты приобрели реальную почву после разработки Т. Бернерсом-Ли основ "всемирной паутины" *World Wide Web* и семантической сети *Semantic Web*. Семантический Вэб — совокупность идей и средств, направленных на решение проблем семантического поиска информации через Интернет [4].

Технологии Интернет не только существенно расширяют коммуникационные возможности людей. В условиях экспоненциального роста объема накопленной человечеством информации наиболее заметно воздействие глобальной сети на интеллектуальные сферы человеческой деятельности, на технологии накопления и распространения знаний. Создаются системы баз знаний, иначе СОЗ (системы, основанные на знаниях), они включают, кроме собственно данных, также средства управления знаниями, моделирования и оценки ситуаций, логического вывода и поддержки принятия решений [5].

Системы, основанные на знаниях, могут быть мультидисциплинарными или прикладными. В качестве примера работ по созданию мультидисциплинарной системы можно назвать проект *Cyc*, направленный на решение сложных задач из области искусственного интеллекта на основе логического вывода и использования более миллиона утверждений, правил и общеупотребительных идей, занесенных в базы знаний системы [6, 7]. Прикладные системы ориентированы на определенные предметные области. В частности, к ним относятся корпоративные базы знаний (КБЗ) [8], представляющие собой хранилища информации, наполненные проектной документацией, научно-техническими отчетами и статьями, деловой корреспонденцией, бизнес-документами и т. п., сопровождающими деятельность компании. Важное место среди СОЗ занимают интеллектуальные об-

разовательные системы, которые могут быть как мультидисциплинарными, так и предметно-ориентированными.

Объемы информации в КБЗ непрерывно растут и достигают размеров, препятствующих нахождению в них в приемлемые сроки нужных сведений и использованию этих сведений при принятии решений. Поэтому в составе систем автоматизированного проектирования (САПР) и управления предприятиями (ERP) целесообразно иметь средства оперативного поиска требуемой информации в КБЗ и на основе ее обработки выдачи рекомендаций пользователю по принятию решений.

В основе моделей большинства СОЗ лежат онтологии предметных областей. В частности, в онтологиях должны в качестве концептов (понятий) присутствовать если не все, то, по крайней мере, многие из концептов, характеризующих возможные запросы на поиск решений и фигурирующих в метаданных документов КБЗ.

Процесс проектирования и разработки онтологии называют онтологическим инжинирингом. Основы онтологического инжиниринга изложены в работе [9]. Имеются примеры разработки онтологий в сейсмологии [10], медицине [11], природопользовании [12], технике [13, 14] и других областях, применение онтологического подхода при формировании электронных учебников и учебных пособий показано в работе [15].

Однако разработка онтологий вызывает естественные затруднения в силу слабой формализованности задач онтологического инжиниринга. Соответственно, применение СОЗ в настоящее время носит преимущественно эпизодический характер. Вместе с тем, многие сложные проблемы производства, проектирования, образования, число которых увеличивается, не имеют удовлетворительного решения без применения интеллектуальных информационных технологий.

Данная статья посвящена обзору применения онтологий в интеллектуальных технологиях. Отмечены примеры разработки, объединения и применения онтологий для поиска информации, кластеризации, классификации, извлечения информации из текстов, поддержки принятия решений и др. Кроме того, рассмотрен подход к поиску решений в КБЗ на основе семантических сетей патернов проектирования.

Аспекты онтологического инжиниринга

Термин "онтология" впервые появился в XVII веке и обозначал науку о бытии — раздел философии, посвященный проблемам мироустройства. В XX веке он стал использоваться также в искусственном интеллекте, где под онтологией подразумевается система понятий (концептов, сущностей, классов), отношений между ними и правил операций над ними в определенной предметной области или, другими словами, онтология есть спецификация концептуализации предметной области [16, 17]. Более формально онтология определяется как триплет

$$O = \{T, R, F\},$$

где T — множество концептов предметной области, которую описывает онтология; R — множество отношений между концептами; F — функции интерпретации, заданные на сущностях и/или отношениях онтологии [18]. Иерархическая система понятий, соответствующая условию $F = \varnothing$, называется таксономией, если же $F = \varnothing$ и $R = \varnothing$, то онтология преобразуется в простой словарь (глоссарий).

Онтологии классифицируются по ряду признаков [19, 20]. Например, в каждой предметной области могут быть созданы свои прикладные онтологии со специфическими сущностями и отношениями. Такие онтологии называют предметными. В отличие от них метаонтологии характеризуют наиболее общие междисциплинарные понятия и их связи, а задачные онтологии, наоборот, описывают концептуализацию отдельных проблем.

Создание онтологий — процесс итерационный и выполняется человеком с использованием поддерживающих инструментальных программных средств. Общий подход к созданию онтологий включает несколько этапов [18]: 1) определение целей и границ описываемого приложения; 2) выбор основных концептов с определением множеств синонимов (синсетов); 3) определение слотов; 4) определение типов и значений (фасетов) слотов; 5) определение экземпляров концептов; 6) верификация созданной онтологии. Различают нисходящий, восходящий, смешанный стили разработки. При нисходящем стиле выбор концептов начинается с наиболее общих понятий, при восходящем — наоборот, с наиболее конкретных.

Инструментальные средства создания и сопровождения онтологий служат для редактирования, просмотра, документирования онтологий, импорта и экспорта онтологий между системами, могут включать и функции управления онтологиями. Примерами редакторов онтологий могут служить *Ontolingua*, *Protege*, *OntoEdit*, *OilEd*, *WebOnto*, *ODE*, *KADS22* [18, 19, 21, 22]. В ряде систем, кроме функции редактирования онтологий, предусмотрено выполнение таких операций над онтологиями, как выравнивание — установка различного вида соответствий между двумя онтологиями для того, чтобы они могли использовать информацию друг друга; *отображение* (*mapping*) — нахождение семантических связей между подобными элемен-

тами разных онтологий; *объединение* — формирование онтологии, объединяющей информацию двух или более заданных онтологий [18]. Так, объединение онтологий входит в число функций систем *OntoMerge*, *Chimaera*, *OBSERVER*, *PROMPT*, автоматическая разметка *Web*-страниц осуществляется в *SHOE'S Knowledge Annotator*, *MnM* и др.

Известны попытки ввести в процесс разработки онтологий элементы автоматизации. Так, авторы работы [23] предлагают осуществлять построение онтологий на основе извлечения знаний из терминологических словарей с помощью разрабатываемой системы продукций.

Для описания онтологий и управления знаниями, в частности при поиске информации, в СОЗ применяют языки онтологий [24]. К ним относятся как традиционные языки спецификации онтологий (например, *Ontolingua*, *CycL*); языки, основанные на дескриптивных логиках (*LOOM*); языки, основанные на фреймах (*OKBC*, *OCML*, *Flogic*) и языки, основанные на *Web*-стандартах (*XOL*, *SHOE*, *UPML*). Специально для обмена онтологиями через *Web* были созданы языки *RDF* [25], *DAML*, *OIL*, *OWL*.

Удобным способом представления концептов и отношений между ними являются семантические сети. К важным характеристикам семантической сети относятся типы учитываемых отношений. Возможен учет отношений многих типов. Примеры используемых в онтологиях типов отношений приведены, например, в работах [23, 26]. Как правило, среди учитываемых типов фигурируют отношения "целое—часть" и/или "род—вид". В частном случае только родовидовых отношений семантическая сеть есть таксономия, т. е. иерархическая система концептов (таксонов). Если дополнительно учитываются отношения агрегации ("целое—часть"), то семантическая сеть приобретает структуру И-ИЛИ графа.

Применения онтологий

Решение большинства задач в интеллектуальных системах связано в той или иной мере с поиском релевантной информации. В основе поиска лежит сопоставление запроса пользователя с поисковыми образами документов, в результате отбираются релевантные документы, т. е. документы, чьи поисковые образы соответствуют запросу. Релевантность документа определяется степенью его соответствия запросу, а эффективность поиска оценивается отношением числа выданных релевантных документов к числу имеющихся в базе релевантных документов (коэффициент полноты) или к общему числу выданных документов (коэффициент точности) [27].

Поисковые образы, называемые также моделями документов, состоят из атрибутов и/или ключевых слов документов и называются метаданными. Наиболее известной системой метаданных для документов достаточно общего вида является система Дублинского ядра DC (*Dublin Core*) [28]. В набор метаданных DC входят слоты (параметры, атрибуты):

- Title (Заголовок) — название, присвоенное ресурсу создателем или издателем;
- Creator (Автор) — человек или организация, создавшие ресурс;
- Subject (Предмет) — тема ресурса;
- Description (Описание) — текстовое описание содержания ресурса, например реферат;
- Date (Дата) — дата создания ресурса;
- Type (Тип) — категория ресурса (например, учебник, статья, технический отчет);
- Identifier (Идентификатор) — уникальный идентификатор ресурса и некоторые другие (всего 15 слотов).

Другим примером набора метаданных может служить система описания образовательных ресурсов, предложенная в спецификации *Learning Resource Metadata Specification* [29]. В набор входят следующие слоты: автор, название, предметная область, аннотация, владелец ресурса, ключевые слова и др.

Определение метаданных — задача семантического анализа текста документа. Семантический анализ заключается в определении смысловых характеристик имеющихся в документе слов или словосочетаний. В простых системах семантического анализа выполняются лишь морфологический анализ слов, т. е. выделение основ слов, определение свойств слова (часть речи, падеж, число и т. п.), идентификация в множестве слов (словаре). В большинстве случаев метаданные используются для контекстно-свободного поиска документов по запросу, сопоставления множества слов (терминов), указанных в запросе пользователя, с ключевыми словами из текста документа. Более сложные методы анализа в большинстве случаев не задействуются [30].

Метаданные представляются в виде фрейма, часто называемого паттерном или шаблоном. В виде паттерна проектирования с той или иной степенью детализации можно отобразить любое проектное решение. Паттерн проектирования определяется как формализованное описание часто встречающейся задачи проектирования, представление удачного решения задачи, а также рекомендации по применению этого решения в различных ситуациях [31].

Для повышения эффективности поиска применяют контекстный поиск, основанный на использовании ключевых слов в тех или иных соче-

таниях. Для образования словосочетаний используют факты близкого расположения связываемых слов в тексте документа. Так, под расстоянием между словами A и B можно понимать число слов, разделяющих A и B в тексте документа, и если слово A попадает в малую окрестность слова B, то к ключевым словам добавляется словосочетание $\{A, B\}$ [32].

В задачах кластеризации и классификации документов используются модели документов в виде паттернов или множества информативных признаков $\mathbf{X} = (x_1, x_2, ..., x_n)$, где x_i — вес i-го признака. В простейшем варианте взвешивания n — число терминов, вес $x_i = 1$, если термин встречается в документе, иначе $x_i = 0$. Применяют и более совершенные способы взвешивания, например, x_i определяют как частоту встречаемости i-го слова в документе, учитывают тематическую близость и неравномерность использования терминов, длину документов и т. п. [33]. Обзор методов кластеризации и классификации приведен, например, в работе [34].

Одним из важных направлений искусственного интеллекта является развитие *Data Mining*, т. е. путей решения задач "добычи" знаний и порождения новых знаний путем обработки некоторых элементов уже имеющихся знаний. Частный случай *Data Mining*, известный как *TextMining*, — извлечение знаний из текстов [35]. Как отмечено в работе [36], систем автоматической обработки документов с целью извлечения знаний в настоящее время не создано, за исключением отдельных узкоспециализированных систем, примерами которых могут служить системы выделения из текстов ссылок на именованные объекты или обработки опубликованных в Интернете сведений о характеристиках и качестве продукции определенного типа. Такие системы, в частности, используются для решения задач автоматического аннотирования документов.

Корпоративные базы знаний состоят из множества документов, в некоторых из них содержатся те или иные сведения, касающиеся проектных решений. Принятие решения во многих случаях может быть сведено к нахождению релевантных документов в КБЗ.

Принятие решений на основе БЗ включает две фазы: поиск документов, содержащих сведения о предыдущем опыте решения аналогичных задач, и обработка сведений из найденных документов. Вопросы поиска информации с использованием онтологий рассматривались в ряде работ, например, [37]. Возможность использования при поиске окрестностей заданных в запросе концептов была высказана в работе [38]. В то же время, как отмечено выше, обработка сведений, как правило, выполняется вручную.

Если БЗ имеет структуру И-ИЛИ дерева, то поиск решений может выполняться на основе интерактивного справочника, задающего пользователю в вершинах ИЛИ вопросы с предложением нескольких вариантов продолжения поиска [39]. Близким к онтологическому методу поиска решений является метод поиска по подобию, когда в процессе поиска вырабатывают некоторые дополнительные оценки релевантности на основе семантики уже найденных документов.

Ключевая роль в повышении эффективности задач поиска информации, кластеризации, классификации, аннотирования документов, обработки знаний должна быть отведена онтологиям. В моделях документов в качестве параметров x_i могут использоваться те или иные оценки соответствия документа и предметных онтологий. Благодаря онтологиям, появляется возможность использования семантического поиска, при котором релевантность документов оценивается "близостью" метаданных документов и концептов запроса в семантических сетях приложений. Другими словами, релевантность документов определяется не на основе близости слов в тексте, а на основе близости концептов в предметных онтологиях.

Поиск решений на основе семантических сетей паттернов проектирования

Паттерны проектирования в зависимости от целей принятия решения и предметной области могут иметь различную структуру. Структурам паттернов проектирования посвящена работа [40], много внимания уделяется разработке паттернов проектирования программного обеспечения [31, 41].

В слотах паттернов проектирования, в первую очередь, должны быть отражены сведения о предмете (приложении), проблеме (задаче), методах, средствах и результатах решения проблемы.

Пример структуры паттерна проектирования показан на рис. 1. Слот "Прочее" может содержать атрибуты, характерные для таких систем, как например DC.

Значениями слотов паттернов являются концепты из соответствующих онтологий предметных областей и ссылки на документы в КБЗ. Другими словами, множество концептов каждого слота и их отношения образуют семантическую сеть слота (ССС).

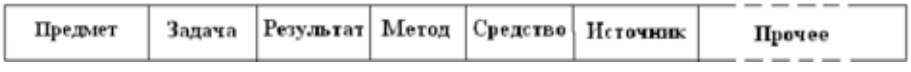


Рис. 1. Пример структуры паттерна проектирования

При решении задач слот может выполнять роль исходных данных, искомого результата или быть нейтральным, соответственно, слот будем называть исходным, искомым или нейтральным.

В запросе к КБЗ при решении задач указываются концепты исходных слотов и помечаются искомые слоты, обычно это "Ссылка на документ в БЗ". Паттерны, перспективные для решения задачи принятия решения, будем называть целевыми.

Обозначим $C_{зi}$ и C_{ki} — концепты, указанные в i-м исходном слоте соответственно запроса и k-го паттерна. Очевидно, что k-й паттерн будет целевым при совпадении $C_{зi}$ и C_{ki} во всех исходных слотах. Но к целевым могут относиться и паттерны других документов, наиболее "близкие" в определенном смысле к запросу, принимаемые как релевантные.

Для оценки релевантности целевых паттернов необходимо ввести меру близости запроса и k-го паттерна R_k как взвешенную сумму расстояний r_{ki} между концептами, указанными в исходных слотах запроса и k-го паттерна:

$$R_k = \sum_{i \in I} w_i r_{ki},$$

где w_i — весовой коэффициент, оценивающий важность i-го слота, I —множество номеров исходных слотов. Дифференциация ключевых слов по их связям с ССС повышает результативность поиска.

В свою очередь, расстояние r_{ki} определяется как минимальная из длин простых цепей, соединяющих вершины концептов $C_{зi}$ и C_{ki} в графе, соответствующем семантической сети i-го слота. При этом ССС есть подсеть общей семантической сети приложения, удовлетворяющая следующим условиям:

1) учитываются только родовидовые и агрегативные отношения;

2) любой из концептов предметной онтологии может быть отражен в семантической сети не более, чем одного слота

$$B_i \cap B_j = \varnothing,$$

где B_i и B_j — множества концептов i-го и j-го слотов соответственно;

3) родовидовые и агрегативные связи между вершинами семантических сетей разных слотов отсутствуют:

$$P_{ij} = \varnothing,$$

где P_{ij} — множество родовидовых и агрегативных связей между B_i и B_j;

4) семантические сети слотов связаны друг с другом через вершины паттернов (рис. 2), но эти связи при определении r_{ki} не учитываются.

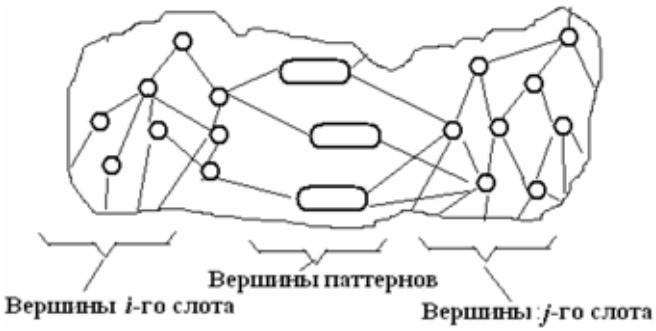


Рис. 2. Фрагмент ССПП

Соблюдение этих условий, в частности, подразумевает необходимость для ряда одинаковых терминов, которые, казалось бы, могли обозначать один и тот же концепт в семантических сетях двух или более слотов, вводить концепты-омонимы. Например, термин "управление" не должен рассматриваться как обозначение одного и того же концепта и в слоте "предмет", и в слоте "задача".

Таким образом, семантическая сеть паттернов проектирования (ССПП) есть объединение семантических сетей разных слотов и вершин паттернов, соответствующих конкретным документам в КБЗ.

Семантические сети слотов постоянно развиваются вместе с развитием общей онтологии приложения, в частности, формирование метаданных новых документов может вызывать появление новых концептов в онтологиях КБЗ.

Построение модели управления знаниями на основе онтологий предметных областей обеспечивает ряд преимуществ.

Во-первых, автоматизируется преобразование текстовых документов в гипертекстовые. Каждое упоминание концепта (основного термина или его синонимов) в тексте документа превращается в гиперссылку и отображается в том слоте паттерна, к ССС которого относится этот концепт. Тем самым частично автоматизируется формирование паттернов проектирования.

Во-вторых, дифференцированный и взвешенный учет роли каждого ключевого слова при поиске релевантных документов повышает эффективность поиска.

В-третьих, имеются возможности регулирования размеров поискового пространства, в том числе его расширения за счет учета окрестностей ключевых концептов в ССС.

Кроме того, появляется возможность предварительной кластеризации концептов в онтологии, что позволяет решать задачи кластеризации и классификации документов на основе сопоставления их метаданных с кластерами концептов.

Алгоритм поиска решения, в основе которого лежит упорядочение документов из КБЗ по зна-

чениям R_k, может быть кратко описан следующим образом:

Обнуление всех $R[k]$;
Циклический процесс по параметру i (i — номер исходного слота)
Циклический процесс по параметру k
{Расчет $R[k] = R[k] + w[i]*r[k][i]$;
Упорядочение документов по значениям $R[k]$;
}
Отбрасывание документов со значением $R[k]$ выше заданного порогового значения $m1$;
Предъявление пользователю верхней части полученного списка документов, ограниченного значением $R[k] <= m1$.

Цикл по i целесообразно начинать с наиболее информативных исходных слотов.

Искомое решение (если оно имеется в документах КБЗ) может содержаться в одном или нескольких документах списка. Кроме того, если в паттернах имеются искомые слоты, значения которых отвечают потребностям пользователя, то решение может быть получено без обращения к собственно документу.

Онтологии в системах информатизации образования

В информатизации образования выделяют группы коммутативных и сущностных проблем. В технологиях коммутативных проблем быстрый прогресс обусловлен развитием Интернет-2 [42], высокими скоростями передачи данных, потоковым видео, реализацией быстрого доступа к распределенным БЗ, агрегированием информации из разных источников с помощью средств RSS [43]. Онтологии применяют в технологиях сущностных проблем, направленных на создание, преобразование, использование контента при обучении.

В свою очередь, сущностные проблемы можно разделить на проблемы проектирования и управления содержанием (контентом) обучения. В задачи проектирования входит разработка учебных планов и обеспечивающих их электронных образовательных ресурсов (ЭОР). Задачи управления решаются непосредственно в процессе обучения и направлены на оперативную корректировку хода обучения в зависимости от результатов промежуточного тестового контроля усвоения обучаемым предлагаемого учебного материала.

Значительную роль в вопросах создания электронных учебных изданий (ЭУИ) и их адаптации к особенностям конкретных обучаемых сыграла концепция модульности контента [44] и ее воплощение в стандарте SCORM [45].

В большинстве инструментальных сред создания гипертекстовых ЭУИ межмодульные связи являются фиксированными (рис. 3, *а*), что существенно ограничивает возможности изменения структуры и состава модулей в ЭУИ, т. е. ограничивает возможности адаптации контента к запросам и уровню предварительной подготовки обучаемых. Поэтому в моделях ЭУИ, соответствующих стандарту SCORM, ради адаптивности ЭУИ произошел отказ от использования межмодульных ссылок (рис. 3, *б*). В отличие от технологии SCORM, в онтологической технологии ТРЕК [15] модули содержат гипертекст с возможными гиперссылками на другие модули, что превращает каждый ЭУИ в средство навигации по различным разделам БЗ (рис. 3, *в*). Гиперссылки осуществляются через посредство концептов онтологии, которая выполняет роль интегрирующей среды, и поэтому при удалении или перемещении модулей в БЗ не требуется корректировка остающихся модулей. Появляется возможность разработки оптимальных траекторий обучения [46] и их обеспечение оперативно создаваемыми индивидуализированными ЭУИ.

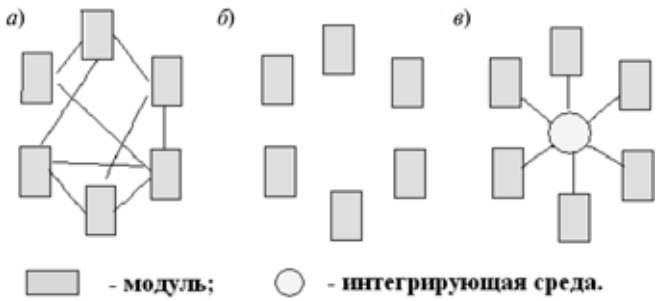


Рис. 3. Варианты структуры ЭУИ:
а — монолитная; *б* — SCORM; *в* — на основе онтологий

Онтологии могут быть полезны также при проектировании тестов, автоматизации оценивания знаний и управления траекториями обучения по результатам тестирования.

Онтологический подход к созданию и применению ЭОР реализован в системе "База и генератор образовательных ресурсов" (БиГОР) [15].

Трудоемкость создания и сопровождения онтологий довольно велика и потому для их развития важно иметь средства объединения онтологий, созданных разными коллективами. Задача объединения онтологий описана в ряде источников, например [21, 47—52], однако автоматизированы лишь отдельные операции композиции онтологий.

Автоматизация объединения онтологий упрощается, если при их создании принята идентичная интерпретация разнонаправленных отношений, например, только "род—вид" с запретом интерпретаций "вид—род", т. е. из двух стилей изложения материала — дедуктивный (нисходящий) или индуктивный (восходящий) — выбран какой-либо один.

Заключение

Рост числа проблем, для решения которых отсутствуют приемлемые формальные методы, обу-

словливает актуальность развития методов искусственного интеллекта. Системы, основанные на знаниях, все шире начинают использоваться в различных областях науки и образования.

В ближайшее время следует ожидать появления новых интеллектуальных технологий и систем поддержки научной и образовательной деятельности, повышенная эффективность которых обусловлена применением онтологий.

Список литературы

1. **Титов В. В., Кузнецов С. В.** Технологии производства баз знаний. URL: http://www.knowbase.ru/
2. **Башмаков А. И., Башмаков И. А.** Интеллектуальные информационные технологии. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2005. — 304 с.
3. **Громов Г. Р.** От гиперкниги к гипермозгу. — М.: Радио и связь, 2004. — 208 с.
4. **W3C** Semantic Web Activity. URL: http://www.w3.org/2001/sw/
5. **Базы** знаний. URL: http://subd2.ru/t9rlpartl.html
6. **Википедия.** Cyc. URL: http://ru.wikipedia.org/wiki/Cyc
7. **OpenCyc.** URL: http://www.opencyc.org/
8. **Гаврилова Т. А., Григорьев Л. Ю.** Разработка корпоративных систем управления знаниями. URL: http://www.big.spb.ru/publications/bigspb/km/create_kms.shtml
9. **Guarino N.** Understanding, Building, And Using Ontologies. URL: http://ksi.cpsc.ucalgary.ca/KAW/KAW96/guarino/guarino.html
10. **McLeod D., Sang Soo Sung.** Ontology-Based Semantic Information Management for Seismology and Geoscience. IMSC Technical Report 05-002. — URL: http://imsc.usc.edu/papers/techreports/pdfs/
11. **Сетевая** мультиагентная модель системы управления здравоохранением региона и система контроля эффективности и качества работы врачей поликлиники / С. В. Батищев, В. А. Виттих, Д. В. Волхонцев и др. — URL: http://www.kg.ru/support/library/polyclinic/
12. **Петухов В. В.** Система интеграции информационных ресурсов при моделировании природно-хозяйственных объектов / Автореферат дис. на соискание уч. степени канд. техн. наук. — СПб.: СПИИА РАН, 2009. — 16 с.
13. **Мартыненко А. А., Шкаберин В. А.** Применение онтологического подхода для реализации системы интеллектуального поиска в области CALS-, CAD-, CAM-, CAE-технологий // Вестник БрГТУ. 2008. № 2. С. 103—110.
14. **Евгенев Г. Б.** Интеллектуальные системы проектирования. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2009. — 334 с.
15. **Норенков И. П., Уваров Ю. М.** База и генератор образовательных ресурсов // Информационные технологии. 2005. № 9. С. 60—65.
16. **Энциклопедия** искусственного интеллекта. Онтология. — URL: http://ai.iii.ru/index.php
17. **Gruber T. R.** A translation approach to portable ontologies // Knowledge Acquisition. 1993. V. 5 (2). P. 199—220.
18. **Гладун А. Я., Рогушина Ю. В.** Онтологии в корпоративных системах // Корпоративные системы. 2006. № 1. С. 41—47.
19. **Авдошин С. М., Шатилов М. П.** Информационные технологии онтологического инжиниринга // Информационные технологии. 2008. № 10. С. 28—37.
20. **Кудрявцев Д.** Технологии применения онтологий. 2006. URL: http://bigspb.ru/theory/km/onto_technologies.php
21. **Овдей Р. М., Проскудина Г. Ю.** Обзор инструментов инженерии онтологий. // Российский научный электронный журнал "Электронные библиотеки". URL: http://www.elbib.ru/index.phtml?page = elbib/rus/journal/2004/part4/op
22. **Guarino N., Welty C.** Evaluating ontological decisions with OntoClean // Communications of the ACM. 2002. N 45 (2). P. 61—65.
23. **Найханова Л. В., Хаптахаева Р. Б., Янсанова Е. Н.** Создание декларативного метода извлечения знаний из терминологических словарей // Информационные технологии. 2008. № 12. С. 2—8.
24. **OWL,** язык веб-онтологий. Руководство. URL: http://sherdim.rsu.ru/pts/semantic_web/REC-owl-guide-20040210_ru.html.
25. **RDF Primer.** W3C Recommendation 10 February 2004. URL: http://www.w3.org/TR/REC-rdf-syntax/
26. **Stumme G., Medche A.** FCA-Merge: Bottom-up merging of ontologies // 7th Int. Conf. on Artificial Intelligence, (IJCAI'01), Seattle, WA. 2001. P. 225—230.
27. **Соколов А. В.** Методика оценки максимально возможных значений показателей эффективности поиска текстовой информации // Информационные технологии. 2009. № 5. С. 18—24.
28. **The Dublin** Core Metadata Initiative. URL: http://dublincore.org/
29. **Learning** Resource Meta-data Specification. V. 1.3. URL: http://www.imsglobal.org/metadata/index.html
30. **Селезнев К.** Обработка текстов на естественном языке // Открытые системы. 2003. № 12.
31. **Дубина О.** Обзор паттернов проектирования. URL: http://www3.citforum.ru/SE/project/pattern/index.shtml
32. **Сулейманов А. Ш.** Метод определения контекстных слов при анализе текста // Информационные технологии. 2009. № 7. С. 46—49.
33. **Толчеев В. О.** Методы выявления информативных признаков в задаче классификации текстовых документов // Информационные технологии. 2005. № 8. С. 14—21.
34. **Технологии** анализа данных: Data Mining, Visiul Mining, OLAP / А. А. Барсегян, М. С. Куприянов, В. В. Степаненко, И. И. Холод. — СПб: БХВ-Петербург, 2007. — 384 с.
35. **Хорошевский В. Ф.** Пространства знаний в сети Интернет и Semantic Web. — URL: http://www.raai.org/ribrary/aidt/aidt2008-1/aidt2008-1.files/2008-1-80-97.pdf
36. **Ермаков А. Е.** Извлечение знаний из текста и их обработка: состояние и перспективы // Информационные технологии. 2009. № 7. С. 50—55.
37. **Боровикова О. И., Загорулько Ю. А.** Организация порталов знаний на основе онтологий. URL: http://www.kmtec.ru/publications/library/select/organiz_pk_na_osn_ontology.shtml
38. **Захарова И. В., Городечный П. П.** Об одном подходе к автоматическому построению онтологии для задач анализа текстов. URL: http://www.dialog-21.ru/dialog2009/materials/html/20.htm
39. **Титов В. В.** Интерактивный справочник. URL: http://serendip.narod.ru/order/sprav/sprav0.htm
40. **Соснин П. И.** Архитектурное моделирование автоматизированных систем. — Ульяновск: УлГТУ, 2002. — 140 с.
41. **Ларман К.** Применение UML и шаблонов проектирования. Введение в объектно-ориентированный анализ и проектирование. — М.: Вильямс, 2004. — 496 с.
42. **Internet-2.** URL: http://www.internet2.edu/
43. **Спецификация** RSS 2.0. — URL: http://beshenov.ru/rss2.html
44. **Норенков И. П.** Концепция модульного учебника // Информационные технологии. 1996. № 2. С. 22—24.
45. **SCORM.** Shareable Content Object Reference Model. 2d Edition. — Advanced Distributed Learning, 2004.
46. **Норенков И. П., Соколов Н. К.** Синтез индивидуальных маршрутов обучения в онтологических обучающих системах // Информационные технологии. 2009. № 3. С. 74—77.
47. **McGuinness P., Fikes R., Rice J., Wilder S.** An environment for merging and testing large ontologies // In Proc. of the Seventh Int. Conf., KR2000. San Francisco: Morgan Kaufmann Publishers. 2000.
48. **Su X.** A Text Categorization Perspective for Ontology Mapping. — URL: http://lvk.cs.msu.su/~bruzz/articles/classification/
49. **Stumme G., Medche A.** FCA-Merge: Bottom-up merging of ontologies // 7th hit. Conf. on Artificial Intelligence (IJCA'01), Seattle, WA. 2001. P. 225—230.
50. **Skvortsov N. A.** Issues of reconciliation of heterogeneous ontological models and ontological contexts.. Ontological Modeling / Ed. by L. A. Kalinichenko // Proc. of the Workshop. — M.: IPI RAN, 2008. P. 149—166.
51. **Щербак С.** Иерархически организованные онтологии + + плагин объединения онтологий. — URL: http://shcherbak.net/2008/02/ierarxicheski-organizovannye-ontologii-plagin-obedineniya-ontologij/
52. **Карпенко А. П., Сухарь Р. С.** Методы отображения онтологий. Обзор // Электронное научно-техническое издание "Наука и образование". 2009. № 1. — URL: http://technomag.edu.ru/doc/115931.html

УДК 519.254

А. С. Титов, аспирант,
МГУ им. М. В. Ломоносова,
e-mail: astitov@s2s.msu.ru

Построение деревьев визуализации с помощью методов гравитационной кластеризации

Представлен обзор существующих подходов к визуализации многомерных данных, описаны их недостатки и достоинства, сформулированы критерии, которым должен удовлетворять искомый визуализатор. Введены понятия визуализации, визуализатора, предложен метод визуализации, использующий алгоритм гравитационной кластеризации.

Ключевые слова: *визуализация, многомерные данные, кластерная иерархия.*

Введение

Развитие сетевых технологий и метасети Интернет привело к тому, что объем доступной для массового пользователя информации в электронном виде перманентно увеличивается. Умение эффективно работать с этой информацией, в первую очередь — обеспечивать поиск нужных документов, определяет в настоящее время успех не только в науке и образовании, но и в области материального производства, в бизнесе и других сферах человеческой деятельности. К активно используемым механизмам оптимизации такого поиска относятся систематизация документов по тематикам, различные способы их индексирования и целый ряд других. Самостоятельное место в таких подходах занимают вопросы визуализации отображения взаимосвязей между текстовыми документами. В настоящее время самым распространенным способом такой визуализации является система иерархий вложенных папок, которая, как правило, строится вручную. Ее цель — упрощение поиска нужного документа. Такая структура хранения документов удобна только при небольшом их числе. При увеличении числа документов появляются различные взаимосвязи между ними и группами документов, которые невозможно представить в виде иерархии. Это обстоятельство требует разработки других способов визуализации.

Постановка задачи

Пусть имеется набор документов, которые представлены точками в некотором пространстве R^m с заданной метрикой d. Необходимо разработать такой математически обоснованный метод для работы с этими данными, использование которого позволяло бы изучать взаимосвязи между документами, например, с позиции их близости по тематике или по смыслу. В настоящей статье представлен обзор существующих подходов к визуализации многомерных данных, описаны их недостатки и достоинства, сформулированы критерии, которым должен удовлетворять искомый визуализатор.

Обзор существующих методов визуализации

Многомерное шкалирование. Все первые разработки в области визуализации многомерных данных относятся к многомерному шкалированию. Пусть **A** — $(n \times n)$-матрица расстояний между n точками. В качестве координат этих n точек на плоскости рассматриваются $\{(x_i, y_i)\}_{i=1}^{n}$, при которых достигается минимум функции S, именуемой стрессом:

$$S(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \sum_{j=1}^{n} \sum_{i=1}^{n} \left(\frac{d(\boldsymbol{x}_i, \boldsymbol{x}_j) - a_{ij}}{a_{ij}} \right)^2,$$

где $\boldsymbol{x}_i = (x_i, y_i)$, а a_{ij} — элементы матрицы **A**.

Следует заметить, что эти координаты определяются неоднозначно, так как поворот всей совокупности точек или смещение их на один и тот же параметр не изменяет расстояния между ними. По этой причине можно до вычисления минимума функции определить произвольным образом значения x_1, y_1, x_2, что задает "направление данных". Задача многомерного шкалирования вследствие минимизации функции трудоемка, и этот факт является серьезным ограничением при практическом применении данного метода.

Методы визуализации иерархий. Существует достаточно много различных методов визуализации иерархий, большая часть из которых ориентирована на визуальное представление файловой структуры. Основной недостаток этих подходов заключается в том, что они не отображают взаимосвязь между данными.

В работе [1] рассмотрен метод TreeMap. Вначале все дерево представляется как прямоугольник, каждое его поддерево — как подпрямоугольник основного прямоугольника. Построение подпрямоугольников подчиняется следующему правилу: сначала прямоугольник разбивается на прямоугольники с помощью проведения вертикальных линий, потом горизонтальных и снова вертикальных. При этом площадь каждого подпрямоугольника пропорциональна числу вершин в нем. Кроме отображения на картинке размера

(веса) вершины дерева используют раскраску каждого подпрямоугольника для отображения некоторых дополнительных свойств вершин дерева, например, для отображения глубины вершины.

В работе [2] предлагается метод визуализации иерархии данных, называемый BotaTree. Результат представляется в виде биологического дерева. Основной аргумент в пользу эффективности данного метода заключается в том, что в природе биологическое дерево имеет много листьев и, несмотря на их количество, мы без труда различаем всю структуру дерева. При построении этого дерева проводится моделирование, в ходе которого выполняется расчет таких параметров, как толщина, отклонение и тяжесть каждой ветви.

Другим, наиболее распространенным методом к визуализации иерархий является Star Tree [3]. Корень данного типа деревьев помещен в центре некоторой окружности, а его дочерние вершины расположены на окружностях меньшего радиуса.

В работе [4] за основу метода Hyperbolic display взят Star Tree с улучшенной навигацией по данным. Основной недостаток данного метода — отображение всех данных, что практически не реализуемо при больших объемах данных.

Метод визуализации графов. В работе [5] предлагается механизм визуализации графа в виде дерева. Пользователь может просматривать дерево, а также получать дополнительную информацию об отдельной вершине в виде других вершин дерева, связанных с ней. Данный подход ориентирован на однократный проход дерева, а именно — на применение в поисковых системах, где в качестве каталогов используется некоторое, заранее построенное, дерево. Если рассматривать данный метод как основу инструментального средства для анализа данных и их структуры, то изменение дерева в ходе его просмотра вызывает трудности в представлении структуры данных.

Требования к визуализации данных

В работе [6] предложены следующие критерии, по которым оценивается качество визуализации иерархий:

- визуализация данных должна быть интуитивно понятной;
- должна обеспечиваться простота в навигации по данным;
- должен присутствовать механизм, поддерживающий просмотр деталей для произвольной вершины или группы вершин;
- пространство должно использоваться эффективно, т. е. визуализация не должна быть громоздкой.

Разрабатываемый инструмент ориентирован на потребности аналитика, основной задачей которого является изучение структуры данных и взаимосвязей между ними. Возможны следующие пути решения этих задач. Первый заключается в анализе структуры данных с использованием статистических методов, таких как кластерный, факторный, дисперсионный, дискриминантный анализы. Результатом анализа являются таблицы и характеристики в виде чисел, что требует их дальнейшего изучения. Корректный результат можно получить только в том случае, если пользователь обладает достаточными знаниями в области статистических методов и имеет некоторый опыт работы с ними. Второй путь в решении задачи заключается в попытке визуализировать данные для удобного и простого просмотра их структуры и взаимосвязей.

Подход	Критерий				
	1	2	3	4	5
Многомерное шкалирование	+	−	+	+	−
TreeMaps	−	+	+	−	−
BotaTree	+	+	+	−	−
Star Tree	+	−	+	−	−
Hyperbolic Display	+	+	+	−	−
Graphs	+	+	+	+	−

С учетом изложенного выше можно сформулировать следующие критерии, которым должен удовлетворять визуализатор, предназначенный для анализа структуры данных и выявления взаимосвязей между ними:

1) интуитивно понятная визуализация;

2) простота в навигация по данным;

3) эффективное использование пространства при визуализации;

4) отображение взаимосвязи между данными;

5) отображение пространственной структуры данных.

В сводной таблице представлены результаты проверки соответствия существующих методов визуализации предложенным критериям.

Визуализация и визуализатор

Введем понятия визуализации и визуализатора многомерных данных. Пусть

$$X = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}$$

— рассматриваемое множество точек, таких, что $\boldsymbol{X}_i \in R^m$.

Определение 1. Визуализация — множество

$$V = \{P_1, \ldots, P_l\},$$

где

$$P_i = \{\boldsymbol{X}_{i_1}, \ldots, \boldsymbol{X}_{i_k}\},\ \boldsymbol{X}_{i_j} \in X.$$

Определение 2. Две визуализации $V_1 = \{P_1, \ldots, P_l\}$ и $V_2 = \{\bar{P}_1, \ldots, \bar{P}_k\}$ совпадают, если $l = k$ и для каждого i найдется j такое, что

$$P_i = \bar{P}_j, \quad 1 \leqslant i \leqslant l, \quad 1 \leqslant j \leqslant k.$$

Определение 3. Визуализатор — множество визуализаций, т. е. $\{V_i\}_{i=1}^{s}$. Число s называется размером визуализатора.

Визуализатор — совокупность комбинаций точек и групп точек, каждая из которых является частью всех данных, которые пользователь может "увидеть".

Утверждение 1. Размер каждого визуализатора, состоящего из различных визуализаций, не превосходит 2^{2n}.

Доказательство. В терминах теории множеств визуализатор — множество подмножеств множества X. Размер визуализатора — мощность визуализатора как множества. По этой причине размер визуализатора не может превосходить мощности множества всех подмножеств множества X, которая равна 2^{2n}.

Построим визуализатор, в основе которого лежит дерево объединения [7]. Рассмотрим некоторое дерево объединения. Каждому листу дерева объединения соответствует некоторая точка $\boldsymbol{X}_i$. Каждой вершине дерева объединения v соответствует множество точек, которые, в свою очередь, соответствуют листьям поддерева с корнем v. Далее, в зависимости от контекста, под v будет подразумеваться вершина дерева или множество точек из X, соответствующих данной вершине.

Пусть v_1 — вершина дерева объединения, а v_2, ..., v_m — ее братья. Каждой вершине дерева объединения v_1 сопоставим визуализацию, состоящую из элементов вершин-кластеров, полученных в результате операции кластеризации по заданному дереву объединения с заранее выделенными вершинами-кластерами v_1, v_2, ..., v_m. При этом вершина v_1 называется *точкой в центре*.

Утверждение 2. Множество визуализаций, соответствующих каждой вершине дерева объединения, образует визуализатор, называемый деревом визуализации.

Утверждение 3. Размер дерева визуализации не превосходит $2n - 1$.

Доказательство. Число визуализаций в визуальном дереве совпадает с числом вершин в соответствующем дереве объединения. Число вершин в дереве с n листьями не больше, чем число вершин в бинарном дереве с n листьями, которое равно $n + \underbrace{1 + \ldots + 1}_{n-1} = 2n - 1$.

Утверждение 4. Пусть v_1 — вершина дерева объединения, а v_2, ..., v_m — ее братья. Тогда визуализации, соответствующие вершинам v_1, ..., v_m, совпадают.

Утверждение 5. Размер дерева визуализации не превосходит n.

Доказательство. Согласно предыдущему утверждению, визуализация для братьев v_1, ..., v_m — общая, поэтому ее можно отождествить с их родителем. Таким образом, число различных визуализаций равно числу вершин в дереве объединения без учета листьев и с учетом визуализации, соответствующей корню дерева. Таким образом, получаем $2n - 1 - n + 1 = n$.

Сопоставление визуализации точек на плоскости

Для работы с визуализатором необходимо каждой визуализации сопоставить некоторый объект, воспринимаемый человеком. В данной статье в качестве объекта выбрано множество кругов на плоскости. Круги в дальнейшем будут называться точками на плоскости.

Сопоставление визуализации точек на плоскости можно осуществить различными способами. В данной статье рассматривается следующая процедура. Визуализации V, порожденной вершиной v, сопоставляются точки на плоскости, которые являются решением задачи многомерного шкалирования со стрессом

$$S(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \sum_{j=1}^{n} \sum_{i=1}^{n} \left(\frac{d(\boldsymbol{x}_i, \boldsymbol{x}_j) - a_{ij}}{a_{ij}} \right)^2,$$

где $\boldsymbol{x}_i \in R^2$, a_{ij} — расстояние между соответствующими вершинами-кластерами.

Следует отметить, что полученное решение не обязательно будет оптимальным, а также оно инвариантно относительно операций поворота и смещения. В качестве расстояния a_{ij} можно рассматривать расстояние между центрами масс вершин-кластеров.

Сопоставление визуализаций

Работа пользователя с визуализатором подразумевает отображение визуализаций и средства для перехода к другим визуализациям. В связи с этим обстоятельством необходимо обеспечить "незаметный" переход от одной визуализации к другой. При описанном подходе сопоставления визуализаций точкам на плоскости визуализации никак не связаны между собой. Они могут иметь разную "направленность", разные смещения, что мешает осознанию структуры данных.

Пусть v_1 — точка в центре, a v_1, ..., v_m — соответствующая ей визуализация. Пусть x_1, ..., x_m — представление визуализации v_1, ..., v_m на плоскости, т. е. каждой вершине v_i соответствует точка x_i.

Для сопоставления визуализаций, построенных по дереву объединения, применяются две операции: смещение и поворот.

Смещение. Операция смещения проводится рекурсивно снизу вверх по дереву объединения.

Пусть v_1 — текущая вершина дерева, которой соответствует визуализация $v_1, ..., v_m$, а w — первый ребенок вершины v_1 с соответствующей ей точкой на плоскости $\boldsymbol{x}_0$ из визуализации с точкой в центре w. Тогда координаты визуализации, соответствующей вершине v_1, определяются как $\boldsymbol{x}_1 + \boldsymbol{t}, ..., \boldsymbol{x}_m + \boldsymbol{t}$, где $\boldsymbol{t} = \boldsymbol{x}_0 - \boldsymbol{x}_1$. Для каждого листа v_1 координаты визуализации определяются как $\boldsymbol{x}_1 + \boldsymbol{t}, ..., \boldsymbol{x}_m + \boldsymbol{t}$, где $\boldsymbol{t} = \boldsymbol{x}_c - \boldsymbol{x}_1$, а $\boldsymbol{x}_c$ — заранее заданная точка, называемая центром экрана.

Утверждение 6. Для каждой визуализации визуального дерева точка в центре имеет координаты $\boldsymbol{x}_c$.

Доказательство. Для листьев визуального дерева данное свойство выполняется, т. е. $\boldsymbol{x}_1 + (\boldsymbol{x}_c - \boldsymbol{x}_1) = \boldsymbol{x}_c$. Данное утверждение является верным в силу того, операция смещения выполняется снизу вверх по визуальному дереву.

Поворот. Операция поворота проводится рекурсивно снизу вверх. Пусть v_1 — текущая вершина дерева, которой соответствует визуализация $v_1, ..., v_m$, (с конфигурацией $\boldsymbol{x}_1, ..., \boldsymbol{x}_m$), а w_1 — первый ребенок вершины v_1, которой соответствует визуализация $w_1, ..., w_k, ..., w_s$, при этом вершины $w_1, ..., w_k$ являются детьми вершины v_1. Обозначим через $\boldsymbol{p}_i$ центр масс всех w_j (точек плоскости), которые принадлежат поддереву с вершиной v_i. Тогда поворот определяется как такой поворот $\alpha_{ij} = \angle \boldsymbol{x}_i \boldsymbol{x}_1 \boldsymbol{z}_j$, при котором минимально значение выражения

$$\sum_{i>1} \min(|\angle \boldsymbol{x}_i \boldsymbol{x}_1 \boldsymbol{y}_i|, |2\pi - \angle \boldsymbol{x}_i \boldsymbol{x}_1 \boldsymbol{y}_i|),$$

где $\boldsymbol{y}_i$ — поворот вокруг $\boldsymbol{x}_1$ точки $\boldsymbol{x}_i$ на угол α_{ij}, $\boldsymbol{z}_j = (\boldsymbol{p}_j + (\boldsymbol{x}_1 - \boldsymbol{p}_1))$.

Если α_{ij} — поворот, на котором достигается минимальное отклонение, то координаты конфигурации точки в центре v_1 примут вид $\boldsymbol{y}_1, ..., \boldsymbol{y}_m$.

Оценка алгоритмической сложности построения визуального дерева

Построение визуального дерева можно разбить на четыре этапа:

- построение дерева объединения;
- вычисление визуализаций визуального дерева;
- вычисление конфигураций визуализаций;
- операции смещения и поворота для визуализаций.

Построение дерева объединения требует $O(n^3 m)$ операций. Вычисление визуализаций визуального дерева требует проведения операции кластеризации по дереву объединения. Таким образом, в связи с тем, что число различных визуализаций не больше чем n, вычисление визуализаций требует $O(n^2)$ операций. Вычисление конфигураций визуализаций основано на многомерном шкалировании. Будем считать, что число операций многомерного шкалирования равно $C(k)$, где k — число точек-переменных в функции стресса. При этом полагаем, если $k < n$, то $C(k) < C(n)$. Тогда вычисление конфигураций визуализаций требует $O(C(n)n)$ операций. Для операции смещения необходим пересчет конфигураций всех визуализаций, что потребует $O(n)$ операций. Операция поворота также требует $O(n)$ операций. В итоге алгоритмическая сложность построения визуального дерева равна $O(n^3 m + C(n)n)$.

Возможно уменьшение алгоритмической сложности построения визуального дерева. В работе [7] описан алгоритм построения дерева объединения за время $O(n^2 m)$. В этом случае алгоритмическая сложность построения визуального дерева равна $O(n^2 m + C(n)n)$.

Методы сокращения ресурсоемкости алгоритма многомерного шкалирования

Существуют следующие способы сокращения вычислительных затрат алгоритма многомерного шкалирования.

1. Использование генетических алгоритмов.

В работе [8] для решения задачи многомерного шкалирования предложено использовать генетические алгоритмы с применением вычислительных кластеров. В работе было получено с достаточной точностью решение задачи поиска минимума функции за меньшее время.

2. Сокращение числа данных.

   a. Знание априорной информации о близких точках.

   Возможно уменьшение сложности решения задачи многомерного шкалирования, если для каждой точки известно, что она гораздо ближе к некоторым точкам, чем к другим. В этом случае можно рассматривать близкие точки как одну, что сокращает число переменных в функции стресса и вычислительные затраты.

   b. Использование методов кластеризации.

   Применение кластеризации данных может быть оправдано, если в них выделяется четкая кластерная структура. После применения некоторого алгоритма кластерного анализа, дающего разбиение данных на кластеры, решается задача многомерного шкалирования, в которой начальными точками являются центры масс найденных кластеров.

3. Сокращение размерности пространства.

   a. Использование другого метода представления данных в многомерном пространстве. В некоторых задачах возможно сокращение размерности задачи, если использовать представление данных в многомерном пространстве с меньшей размерностью пространства.

b. Использование факторного анализа.

В случаях, когда не представляется возможным сокращение размерности пространства, можно использовать методы факторного анализа. Факторный анализ позволяет построить пространство меньшей размерности, но с сохранением представления данных, что достигается за счет построения других осей в пространстве. Классическим примером является представление прямой $y = x$ в одномерном пространстве за счет поворота.

Ниже детально описаны предложенные автором методы сокращения ресурсоемкости алгоритма многомерного шкалирования при построении визуального дерева, к числу которых относятся:

- сокращение числа данных;
- решение упрощенной задачи многомерного шкалирования, например, путем рассмотрения функции стресса как функции от меньшего числа переменных, в предположении, что остальные переменные неизменны.

Сокращение числа данных возможно, если принять предположение, что все документы, принадлежащие одной рубрике, ближе друг к другу, чем к документам других рубрик. Тогда решается задача многомерного шкалирования не для всех документов, а только для рубрик. Перейдем к способу уменьшения числа переменных. Рассмотрим процесс раскрытия точки. Пусть $v_1, ..., v_n$ — некоторые вершины дерева, а их координаты на плоскости $(x_1, y_1), ..., (x_n, y_n)$ соответственно, при этом ни одна из них не является каким-либо ребенком другого. Пусть детьми вершины v_1 являются $v_{1_1}, ..., x_{1_k}$.

Визуализация точек после раскрытия точки v_1 представляется в виде

$$(x_2, y_2), ..., (x_n, y_n), (x_{1_1}, y_{1_1}), ..., (x_{1_k}, y_{1_k}),$$

где значения $(x_{1_1}, y_{1_1}), ..., (x_{1_k}, y_{1_k})$ определяются как значения, на которых функция

$$S(\boldsymbol{x}_{1_k}, \boldsymbol{x}_{1_k}) = \sum_{i=1}^{k} \left(\sum_{j=2}^{k} \left(\frac{d(\boldsymbol{x}_{1_i}, \boldsymbol{x}_j) - a_{1_k j}}{a_{1_k j}} \right)^2 + \right.$$

$$\left. + \sum_{j=1}^{k} \left(\frac{d(\boldsymbol{x}_{1_i}, \boldsymbol{x}_{1_j}) - a_{1_k 1_j}}{a_{1_k 1_j}} \right)^2 \right)$$

достигает минимума.

Таким образом все вершины $v_2, ..., v_n$ фиксированы, и проводится вычисление только координат детей раскрываемой вершины. В отличие от многомерного шкалирования на данном шаге выполняется пересчет не всех координат нарисованных вершин, число которых равно $n + k - 1$, а только новых (k), что снижает вычислительные затраты. Следовательно, число операций сокращается с $C(n + k - 1)$ до $C(k)$.

Заключение

Предложенный в данной статье подход к решению задачи визуализации, в отличие от существующих подходов, удовлетворяет всем предложенным выше требованиям по следующим соображениям.

- Интуитивно понятная визуализация — отображение данных на плоскости и группировка данных, находящихся в "центре", с использованием иерархии.
- Простота в навигации по данным — представление инструментов модификации: раскрытие точки (получение более подробной информации о точке), расположение любой видимой точки в центре экрана (перемещение по данным).
- Эффективное использование пространства при визуализации — отображение небольшого числа точек за счет группировки данных.
- Отображение взаимосвязи между данными определяется расстоянием между ними.
- Отображение пространственной структуры данных с использованием иерархии данных, построенной в ходе работы гравитационного алгоритма кластеризации.

Список литературы

1. **Remi Coulom.** Treemaps for Search-Tree Visualization // The Seventh Computer Olympiad Computer-Games Workshop Proceedings. Maastricht, 2002.
2. **Ernst Kleiberg, Huub van de Wetering, Jarke J. van Wijk.** Botanical Visualization of Huge Hierarchies // Proceedings of the IEEE Symposium on Information Visualization, IEEE Computer Society Washington, DC, USA. 2001.
3. **Lamping J., Rao R.** The Hyperbolic Browser: A Focus + Context technique for Visualizing Large Hierarchies // Journal of Visual Languages & Computing. 1996. Vol. 7. N 1. P. 33—55 (23).
4. **Batagelj V., Pavletic E., Zaversnik M., Korenjak-Cerne S.** Clustering Large Datasets and Visualizations of Large Hierarchies and Pyramids: Symbolic Data Analysis Approach. DIDAY, Edwin (ur.), RODRÍGUEZ, Oldemar // 4th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD'2000, Lyon, France, September 12—16, 2000. Workshop on Symbolic Data Analysis. Lyon: [s.n.], 2000. P. 69—81.
5. **Ming C. Hao, Meichun Hsu, Umesh Dayal, Adrian Krug.** Web-Based Visualization of Large Hierarchical Graphs Using Invisible Links in a Hyperbolic Space // IFIP Conference Proceedings. 2000. Vol. 168. Proceedings of the Fifth Working Conference on Visual Database Systems: Advances in Visual Information Management, 2000. P. 83—94.
6. **Boonthanome Nouanesengsy, Yipeng Li.** Hierarchy Visualizations. Information Visualization, Kluwer Academic Publishing, 2004.
7. **Титов А. С.** Кластеризация на основе гравитационного метода // Информационные технологии и программирование: Межвузовский сборник статей. М.: МГИУ. Вып. 2 (7). 2003.
8. **Julius Zilinskas, Antanas Zilinskas, Audrius Varoneckas.** Parallelization of visualization algorithms based on Multidimensional Scaling, Scientific and Parallel Conputing 06 // Lecture Notes in Computer Science (LNCS), Springer Berlin / Heidelberg. 2006.

УДК 004.912

С. А. Бельков, канд. техн. наук, доц.,
С. Л. Гольдштейн, д-р техн. наук, проф.,
Уральский государственный технический университет, г. Екатеринбург,
e-mail: srgb@mail.ru

Представление материала текстовых и гипертекстовых источников сетью паттернов

Рассмотрена проблема отображения текстов и гипертекстов в более формализованные структуры — сети паттернов. Проанализированы существующие определения паттерна и предложена обобщенная формальная модель сети паттернов, расширяющая арсенал средств, используемых при проектировании структур мультимедийных гипертекстов и модульных систем.

Ключевые слова: *мультимедийный гипертекст, паттерн, сеть паттернов, средство проектирования, модуль знаний*

Введение

В настоящее время накоплено большее количество семантически нагруженных источников (библиографических, текстовых, гипертекстовых, интерактивных, мультимедийных и др.) самой разнообразной тематики. Важным направлением является также разработка гипертекстовых тезаурусов и онтологий [1—6]. Однако существующих эффективных методов их структурной и содержательной формализации явно недостаточно. Целью статьи является рассмотрение возможного инструментария, позволяющего отобразить текст в пространство более формализованных моделей.

Рассуждения будем вести следующим образом. Сначала кратко рассмотрим возможные способы формализованного описания текста. Затем перейдем к обсуждению одной из многообещающих парадигм, связанных с представлением сложных структур посредством простых образцов или шаблонов (паттернов), после чего дадим несколько собственных определений. Для простоты изложения будем рассматривать, например, структуры гипертекстовых страниц, подразумевая, однако, возможность перехода к текстам более широкой природы.

Способы формализованного описания текста

Возникающие здесь задачи связаны с необходимостью отображения произвольного текста (в общем случае гипертекстового, мультимедийного и интерактивного) в более регулярные (формализованные) структуры. Выбор типа регулярных структур может определяться, например, сложностью текста, степенью однородности его составляющих, числом уровней вложенности и т. д.

Для простых текстов вполне применимы отображения, использующие теорию графов. В более сложных случаях можно говорить о сложных структурах представления: предикатах, семантических графах, модулях (взятых, например, из языка UML или концепции SADT), паттернах, и далее возможен переход к иерархиям с блоками опций, семантическим, фреймовым сетям и сетям паттернов.

Таким образом, сложность способов формализованного представления структуры гипертекстов можно разделить на три уровня:

1) простой: графы, опции, алгебра логики, простые модули;

2) средний: иерархии, теги, предикаты, семантические графы, гиперграфы, специализированные модули (подобно тем, что используются в UML и SADT), классы (объекты), паттерны, фреймы;

3) высокий: иерархии с опциями, семантические сети, сети паттернов и фреймов.

Аналоги и проблема их адаптации для текста

Из анализа литературных источников сделан вывод, что с точки зрения способов отображения гипертекстов в более регулярные структуры, одними из наиболее формализованных являются направления, связанные с парадигмами паттернов [7—9]. Традиционно под паттерном понимается некоторый типовой образец или шаблон, применение которого позволяет задать множество типовых элементов, структур, задач, процессов, ситуаций, образов, комплексов ощущений и т. д.

Прикладная теория паттернов

В этой теории используются исследования Гренандера [7] по теории образов. Модели, представленные там, модифицируются таким образом, что все рассматриваемые множества становятся конечными и, таким образом, появляется возможность их практического применения. Дальнейшее развитие теории позволяет говорить о паттернах как о некоторых универсальных модулях [8] и, следовательно, здесь применимы принципы модульного подхода.

Например, с учетом этого направления отдельную гипертекстовую страницу в целом можно рассматривать как некоторый паттерн, имеющий внутреннюю начинку (образующую и содержа-

тельную часть), а также совокупность входных и выходных связей (гипертекстовых страниц или тем, ссылающихся на данную страницу и гипертекстовых страниц, на которые ссылается данная страница или тема). Паттерн является универсальной неделимой единицей, т. е. образующая часть паттерна и его входные и выходные связи рассматриваются всегда совместно и, в этом смысле, паттерн представляет собой некий универсальный и неделимый модуль. Число входов и выходов паттерна может варьироваться, а каждый из них может быть соединен, либо не соединен с выходом или входом другого паттерна.

Определим формализмы прикладной теории паттернов применительно к гипертекстам и с учетом ряда собственных обозначений (более удобных при нашей работе, например, для множества входов и выходов будем использовать переменные x и y).

Сложный текст G образуется множеством простых гипертекстовых страниц (ГС). Структуру каждой ГС можно представить посредством базовой элементарной структуры, которая называется образующей g_i:

$$G = <g_1, g_2, ..., g_n>,$$

где n — число простых гипертекстовых страниц.

Каждая из образующих g_i определяется посредством обобщенного вектора признаков:

$$a(g_i) = <i, t, p_{ih}, x_{im}, y_{ir}>, \quad (1)$$

где i — номер гипертекстовой страницы; t — тип ГС (например, содержит текст или графику); x_{im} — входные связи (ссылки на данную ГС; m — номер входа; для случая, когда из какой-либо ГС имеется несколько ссылок на данную ГС, что нередко случается, появляется дополнительный индекс — x_{iml}, где l — номер копии входа im); y_{ir} — выходные связи (ссылки от данной ГС; r — номер выхода; для случая, когда от данной ГС имеется несколько ссылок на какую-либо конкретную ГС, появляется дополнительный индекс — y_{irk}, где k — номер копии входа ir); p_{ih} — содержит описания внутренних характеристик образующей g_i (h — номер характеристики).

Компонентам p_{ih} и показателям связей x_{im} и y_{ir} соответствуют описывающие их названия и значения домены: D_{ih}, $D_{x,\,im}$, $D_{y,\,ir}$. Если значения всех этих характеристик определены, то имеем *конкретную* образующую. Если значения каких-либо характеристик, но не всех, имеют значение "пусто", то образующая становится *семантической*. Если не определено значение ни для одной из характеристик, то получаем *абстрактную* образующую.

Для образующей, посредством которой мы представляем структуру ГС, имеем, как минимум, следующие типовые конфигурации: линейную (один вход и один выход); анализирующую (один вход и много выходов); синтезирующую (много входов и один выход), начальную (входов нет, выходы есть), терминальную (есть входы, выходов нет).

Структура сети паттернов задается набором ее бинарных связей между входами и выходами: $y_{ir}\ s\ x_{jm}$, где y_{ir} — выход r паттерна i; x_{jm} — вход m паттерна j; s — отношение соединения, которое имеет логическую природу, т. е. может принимать значения "истина" или "ложь".

Паттерны в языках программирования

Сущность парадигмы паттернов, рассматриваемой в программировании, также заключается во введении универсального понятия "паттерн" (образец, шаблон) [9], однако трактуется он здесь иначе. Здесь "паттерн" рассматривается как некоторый аналог объекта или класса и в общем случае объединяет в себе несколько парадигм в представлении данных: графовые, объектно-ориентированные, реляционные, семантические, фреймовые и др.

При таком подходе паттерн, в первом приближении, является *структурно* некоторой надстройкой над объектами или классами и *функционально* — объединяет методы, используемые в объектном программировании, реляционном исчислении, а также в некоторых других языковых парадигмах.

В соответствии с объектным подходом паттерн включает в себя описание его свойств и методов, использует идеи вложенности, инкапсуляции и т. д.

Для случая паттернов можно обобщить, в частности, операции реляционной алгебры (селекция, проекция, объединение, разность, декартово произведение) и теоретико-множественные операции.

Например, представленное в работе [9] определение паттерна (в рамках объединенной парадигмы) может включать следующие части:

- список имен паттернов-классов (метаклассов), к которому принадлежит данный паттерн;
- операторы выборки, формирующие подмножество паттернов, удовлетворяющих некоторым условиям отбора (при этом используются конструкции, подобные запросам языка SQL);
- математические выражения, в состав которых могут входить и логические операторы;
- собственная часть данного паттерна (характеризующая его экземпляр): свойства, методы, определения вложенных паттернов, триггеров и ограничений, а также списка паттернов, для которых данный паттерн выступает как связь;
- функциональная часть, состоящая из алгоритмического модуля (программа или блок в парадигме структурного программирования) и описаний его входных и выходных переменных;

- структурные операторы (например, цикл и условный оператор); рассмотрение структурного оператора как части паттерна предназначено для реализации концепции "элементы программы как объекты и транзакции", что необходимо для динамического изменения программ;
- названия ролей, в которых может участвовать паттерн;
- имя паттерна.

При этом роли использования паттерна могут быть следующими: объект для вызовов свойств и методов; функция или процедура (алгоритмический модуль); запросы на выборку; метакласс; математическое выражение; транзакция (например, для подтверждения или отмены проведенных операций); обозначение структурного оператора для возможности изменения алгоритмического кода.

Для представления посредством паттернов сложных структур, например иерархий, можно использовать свойства наследования или агрегации, рассматриваемые по аналогии с парадигмой объектно-ориентированного программирования.

Специализированные паттерны

Следующий шаг при создании паттернов возникает при переходе от абстрактных моделей к описанию конкретных предметных областей [9].

К специализированным паттернам относятся, например, паттерны из области проектирования, где паттерн описывает содержание некоторой типовой задачи, а также принцип ее решения. Решения здесь также выражаются в терминах объектов и интерфейсов. Назначение специализированного паттерна — предложить решение определенной задачи в заданном контексте. Например, паттерн, используемый для проектирования программных приложений, состоит из четырех основных элементов: *название паттерна* (используя название, можно сразу определить типовую проблему проектирования, способы ее решения и их последствия); *задача* (содержит формулировку задачи и ее контекст); *решения* (описание основных элементов реализации, отношений между ними, функций каждого абстрактного элемента); *результаты* (описания последствий применения паттерна и возникающих при этом компромиссов).

Паттерны проектирования — не очень сложные и типовые предметно-ориентированные решения, применимые в рамках конкретного приложения или подсистемы. Здесь паттерны проектирования используются для описания объектов и классов, а также их взаимодействий, предназначенных для решения задачи проектирования в конкретном контексте.

Критический обзор аналогов

Рассматривая прикладную теорию паттернов [8], можно заключить следующее.

1. В этой теории паттерны предполагается использовать уже на этапе проектирования гипертекста. Однако в связи с огромным количеством уже имеющихся в сети Internet гипертекстов, а также разработок, связанных с созданием гипертекстовых тезаурусов, весьма актуальна и обратная задача: взяв произвольную гипертекстовую страницу, отобразить ее в адекватную сеть паттернов.

2. Рассмотренная формальная модель позволяет создать алгоритм перевода изначально "разношерстного" исходного текста в более формализованную совокупность паттернов, объединенных в сеть. Тем не менее, алгоритм такого перевода авторами прикладной теории паттернов не приводится. Следует добавить, что сложность разработки и реализации такого алгоритма будет зависеть от сложности (число ГС и их связей, степень однородности в типах ГС и т. д.) исходного текста;

3. Поскольку авторы прикладной теории паттернов рассматривали ее приложение только к некоторым простым примерам, не учитывая достаточно серьезную проблему сложности реального гипертекста, для нее, скорее всего, потребуется ввести ряд собственных дополнительных определений.

4. Подход, подразумевающий конкретные и абстрактные описания образующих, позволяет также задуматься о параллелях между паттерном и фреймом. Однако авторы прикладной теории паттернов эти вопросы никак не оговаривают.

Рассматривая паттерновые структуры, используемые в языках программирования [9], в частности в рамках рассмотренного объединенного подхода, заметим:

- используемые в программировании паттерны также подразумевают использование неких формализованных структур (модулей);
- конструкции, представленные в прикладной теории паттернов, можно реализовать и в рамках одной из составляющих (алгоритмического модуля) паттерна, рассматриваемого как класс. Однако при этом входы и выходы модуля необходимо описать не переменными, а более сложными структурами — логическими связями между выходами одного паттерна и входами других;
- к достоинствам следует отнести дополнительное использование объектно-ориентированного подхода, который позволяет отобразить вложенные представления (в прикладной теории паттернов они не были достаточно рассмотрены);
- здесь учитываются элементы других парадигм, например, доступно использование тегов, кроме того, появилась возможность ограничивать рассматриваемые множества посредством опе-

раторов выборки; при выборе вариантов использования паттернов появились роли;
- поскольку объектно-ориентированный подход подразумевает использование методов классов, то появляется возможность определить не только структуры паттернов, но и совокупность операций над ними.

Для паттернов из языков программирования имеем следующие недостатки:
- поскольку прототип, представленный в работе [9], разрабатывался с большей ориентацией на сферу языков программирования, он слабо учитывает особенности семантически нагруженных источников, например текстовых либо гипертекстовых представлений;
- к недостаткам паттернов, рассматриваемых в программировании, следует отнести более сложные процессы формализации.

При рассмотрении специализированных паттернов [9] возникает вопрос о необходимости создания множеств паттернов, адаптированных под конкретную проблемную область (в том числе и для сфер, касающихся гипертекстов), словарей и хранилищ с описаниями специализированных паттернов и т. д.

Таким образом, с точки зрения формализации, например, структуры существующих не очень разветвленных гипертекстовых страниц, прикладная теория паттернов, представленная в работе [8], дает хорошие возможности и для формализации простых гипертекстов достаточно удобна. Более того, структуры, представленные в работе [8], можно в известном смысле назвать атомарными, т. е. имеющими базовый уровень.

Однако в ней отсутствует ряд дополнительных возможностей, использующих парадигмы, взятые из языков программирования [9], в частности, нет возможности для учета большой вложенности структур и описания множества операций, что немаловажно при переходе к более сложным структурам, например к гипертекстовым тезаурусам.

Поэтому за основной прототип нами выбрана парадигма, представленная в работе [8]. Тем не менее, с точки зрения актуальных задач обработки семантически нагруженных источников и разработки гипертекстовых тезаурусов данная модель нуждается в доработке. В частности, во всех рассмотренных аналогах, нет достаточно хорошего определения сети паттернов.

Предлагаемая обобщенная формальная модель сети паттернов

Дополнительно к представленной формальной модели введем несколько более общих определений.

Сеть паттернов определим следующим образом:

$$PN = < C,\ O>,$$

где C — структура сети; O — определенные на сети операции (удаление, включение, сравнение отдельных элементов сети — паттернов, связей);

$$C = <G,\ T,\ H,\ XY,\ V,\ D,\ S>,$$

где G — множество образующих для ГС; H — множество типов характеристик; T — множество типов образующих; XY — множество видов внешних связей, которое разбивается на множество внешних входов X и выходов Y; V — множество видов внутренних связей; D — объединяет соответствующие множествам H, XY, V множества доменов; S — множество видов соединений, которое разобьем, в свою очередь, на внешние и внутренние S_{xy} и S_v, и далее на актуализированные (значение соответствующих логических переменных "истина") и не актуализированные (значение соответствующих логических переменных "ложь").

Например, подмножество видов представления (из множества T типов ГС) можно определить следующим образом: текст, статическая графика, диаграмма, график, формула, анимационный ролик, видео и т. д. Само множество T, как более общее, может учитывать и другие способы типизации.

Множество T видов образующих содержит, в частности, базовый уровень: набор атомарных паттерновых структур. Множество операций над сетью паттернов также включает в себя операции базового уровня, необходимые для работы с отдельными образующими.

Определение отдельного паттерна приобретает следующий вид:

$$G_i = <I,\ ST,\ GO,\ GF,\ GU,\ R>, \qquad (2)$$

где I — имя, идентификатор или номер паттерна; далее имеем соответственно следующие множества определений: ST — структурные; GO — операторные; GF — функциональные (модульные); GU — использования; R — взаимосвязи компонент (для последующих кортежей подразумеваются по умолчанию).

Структурные определения включают:

$$ST = <MC,\ OP>,$$

где MC — множество имен метаклассов; OP — собственные определения конкретного паттерна (списки закрытых и открытых свойств и методов, определений вложенных паттернов, триггеров, ограничений, связанных паттернов).

Операторные определения содержат:

$$GO = <SO,\ MO,\ STO,\ LO,\ U>,$$

где SO — операторы выборки, формирующие подмножества паттернов, удовлетворяющих некоторым условиям; MO — математические и логические операторы (выражения); STO — структурные операторы; LO — операторы, связывающие различные паттерны между собой; U — управляющие

операторы (например, операторы управления транзакциями).

Для множества *STO* полезно также ввести понятие типовой конфигурации и определить сам конфигуратор как некоторую функцию от более или менее устойчивого сочетания определенных компонент. Например, рассмотрим для структуры паттерна зависимость частных (возможны и более общие способы рассмотрения соотношений компонент) типовых конфигураций, определяемых числом ее внешних входов и выходов:

$$C_{xy} = F(X, Y).$$

В данном случае для отдельной образующей также имеем перечисленные выше типовые базовые конфигурации: линейную, анализирующую, синтезирующую, начальную, терминальную.

Множество функциональных (модульных) определений *GF* содержит следующие элементы:

$$GF_i = < i, t, p_{ih}, x_{im}, y_{ir}, v_{ik}, s_{i, xy}, s_{i, v}, o_i >, \quad (3)$$

где i — номер модуля; t — тип модуля; p_{ih} — описания признаков, характеризующих модуль i (h — номер характеристики); x_{im} —внешние входные связи (ссылки на данный модуль; m — номер входа); y_{ir} —внешние выходные связи (ссылки от данного модуля; r — номер выхода); v_{ik} — внутренние связи модуля; $s_{i, xy}$ — актуализированные внешние связи модуля; $s_{i, v}$ — актуализированные внутренние связи; o_i — множество допустимых для модуля операций (частично определяет язык работы с модулем).

Из формулы (3) видно, что она заимствует структуру определения (1) только частично, и, кроме того, становится внутренней частью более общего определения (2) сети паттернов.

Множества определений использования *GU* содержат названия и описания ролей, в которых может участвовать паттерн.

Несоединенные входы, выходы и вектор характеристик паттерна подобны незаполненным фреймовым слотам, конкретизация их соединения может происходить позже. Это позволяет рассматривать паттерн не только как модуль, имеющий внутреннюю структуру, входы и выходы, но и как некий простой типизированный фрейм. Таким образом, сеть паттернов занимает промежуточное положение между семантическими и фреймовыми сетями.

Предложенная обобщенная модель сети паттернов носит универсальный характер. При переходе к отдельным предметным областям появляется необходимость разработки специализированных паттернов, адаптированных для решения задач конкретной предметной области, в частности, возникают следующие подзадачи: формирование совокупности паттернов, выбор основного набора специализированных паттернов; адекватное отображение структуры конкретного гипертекста в сеть паттернов.

Сказанное позволяет сделать заключение, что посредством совокупности паттернов могут быть реализованы и онтологии [10]. В этом случае онтология содержит в себе следующие компоненты:

$$O = < P_{def}, P_{rel}, P_{fun} >,$$

где P_{def} — специализированные паттерны, характеризующие конечное множество понятий и терминов предметной области (ПО); P_{rel} — паттерны, задающие отношения между понятиями и терминами ПО; P_{fun} — паттерны, определяющие конечное множество функций интерпретации терминов или отношений ПО.

Если иметь в виду, что паттерн можно рассматривать как некоторый типизированный фрейм, то же самое определение можно распространить и на более гибкие структуры — фреймы. Достоинством паттернов является то, что они легче поддаются формализации.

В случае, когда отношения и функции не определены, получаем просто словарь паттернов. Он, очевидно, отличается от обычного словаря терминов, так как для каждого специализированного паттерна, кроме всего прочего, должны присутствовать описания их структур, условий применимости и прочие сведения, т. е. уже в самом определении паттерна неявно присутствует некоторая простая онтология.

Такие типовые отношения как "является частью", "содержится в", "является разновидностью", "является членом класса", "есть подобие" и другие также могут быть представлены специализированными паттернами.

Выше рассматривался пример структуры специализированного паттерна для процессов, связанных с областью решения задач проектирования.

Поскольку при описании конкретной ПО полезно иметь несколько уровней абстракции, то формальная модель онтологии, разбиваемая на метаонтологию, предметную онтологию и онтологию задач, также может представляться посредством совокупности паттернов.

Заключение

Таким образом, проанализированы способы формализованного описания семантически нагруженных источников, подробно рассмотрена формализация их посредством паттернов. На основе рассмотренных формальных моделей сформирована обобщенная формальная модель сети паттернов. Сделано заключение, что паттерны являются достаточно универсальным средством

описания, например, посредством паттернов можно представлять компоненты онтологий.

Список литературы

1. **Гаврилова Т. А., Воинов А. В.** Когнитивный подход к созданию онтологий // Научно-техническая информация. Сер. 2. 2007. № 3. С. 19—24.
2. **Гольдштейн С. Л., Инюшкина О. Г., Кармышев В. М.** Развитие системы управления знаниями для разрешения ситуаций в бизнесе. — Екатеринбург: Пироговъ, 2006.
3. **Гольдштейн С. Л**. Системная интеграция бизнеса, интеллекта, компьютера: Кн. 1. Введение в проблематику и постановку задач. — Екатеринбург: Пироговъ, 2006.
4. **Гольдштейн С. Л., Кудрявцев А. Г.** Разрешение проблемных ситуаций при поддержке систем, основанных на знаниях. — Екатеринбург: Пироговъ, 2006.
5. **Тузовский А. Ф.** Онтолого-семантические модели в корпоративных системах управления знаниями: Автореферат дисс. ... уч. ст. д-ра техн. наук. — Томск: ТПУ, 2007.
6. **Бельков С. А., Гольдштейн С. Л.** Гипертекстовый тезаурус системных знаний // Научно-техническая информация. Сер. 2. 1996. № 3. С. 1—11.
7. **Гренандер У**. Лекции по теории образов: В 3-х т. — М.: Мир, 1983.
8. **Шуткин Л. В.** Парадигма модульного мышления в компьютерной науке и практике // Научно-техническая информация. Сер. 2. 2004. № 10. С. 1—14.
9. **Добряк П. В.** Проектирование информационных систем в рамках объектно-реляционного подхода: дис. на соискание канд. техн. наук. — Екатеринбург: УГТУ—УПИ, 2007.
10. **Гаврилова Т. А., Хорошевский В. Ф.** Базы знаний интеллектуальных систем. — СПб.: Питер, 2001.

ПРОГРАММНАЯ ИНЖЕНЕРИЯ

УДК 007; 004.413

В. В. Сафронов, д-р техн. наук, проф.,
ОАО "КБ Электроприбор", г. Саратов,
e-mail: svv@kber.ru
О. Н. Федорец, канд. техн. наук, доц.,
зам. нач. отдела — нач. лаб.,
ФГУ "3 ЦНИИ Минобороны России", г. Москва

Метод построения эффективных моделей разработки программного обеспечения

Рассмотрена классификация моделей разработки программного обеспечения. Введена система критериев, характеризующих эти модели. Поставлена задача выбора эффективных вариантов моделей разработки программного обеспечения, которая сводится к задаче гипервекторного ранжирования. Приведен численный пример.

Ключевые слова: *модель, программное обеспечение, критерии, гипервекторное ранжирование*

Введение

Выбор рациональных моделей разработки программного обеспечения (ПО) представляет собой достаточно сложную задачу для компаний-разработчиков ПО и проектных команд. На практике они не всегда представляют себе все существующие варианты выбора методов, моделей, стандартов и методологий, рассматривают их ограниченный набор, подходят к решению проблемы выбора без учета специфики компании и проекта.

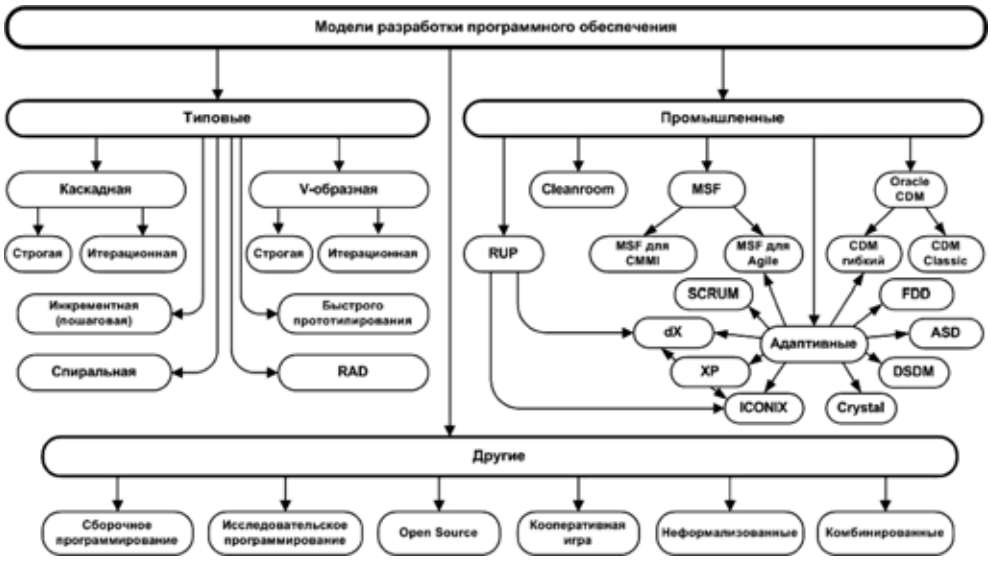
В статье рассмотрены модели разработки программных продуктов, которые нашли широкое применение на практике. На основе анализа научной литературы предложена система критериев, которая позволяет оценить эти модели. С использованием методов системного анализа показан возможный подход к выбору эффективных вариантов моделей разработки программных продуктов.

1. Классификация моделей разработки программного обеспечения

Полезным инструментом для выбора модели разработки ПО является классификация моделей. В настоящее время существует множество подходов к классификации моделей разработки ПО, рассмотрим некоторые из них.

В стандарте [1] упоминаются каскадная, спиральная и эволюционная модели. В новой версии стандарта [2] говорится уже о четырех моделях (добавляется пошаговая модель). В другом стандарте [3] декларируется, что существует множество моделей разработки ПО, но три из них фундаментальные: *каскадная, инкрементная и эволюционная*. Каждая из указанных моделей может быть использована самостоятельно или скомбинирована с другими для создания гибридной модели разработки ПО. Для каждой из них приведены преимущества и недостатки. В работе [4] подробно охарактеризованы шесть основных моделей: *каскадная, V-образная, инкрементная, быстрого прототипирования, спиральная и быстрая разработка приложений (RAD)*.

В работе [5], кроме вышеперечисленных моделей разработки ПО, приводится информация о



Классификация моделей разработки программного обеспечения

Типовые модели — это группа моделей, разработанных, как правило, еще до широкого распространения объектно-ориентированных языков программирования, которые проявили себя в конкретных условиях, имеют определенные преимущества, недостатки и условия применимости. Эта группа моделей устанавливает некоторые основные принципы организации моделей разработки ПО.

К *промышленным моделям* отнесены модели, используемые в различных методологиях программирования. Каждая из таких методологий имеет свои принципы, методы, особенности и, как правило, поддерживается набором CASE-средств, охватывает все этапы жизненного цикла продукта и успешно применяется для решения практических задач.

Существуют также модели разработки программных продуктов, которые трудно отнести к типовым или промышленным, поэтому они вынесены в отдельную группу "*другие*". В эту группу входят следующие модели: сборочное программирование; исследовательское программирование; *Open Source*; кооперативная игра; "кодируем и правим" и комбинированные модели.

2. Формирование совокупности критериев, характеризующих модели разработки программного обеспечения

Каждая из моделей разработки ПО имеет присущие ей преимущества и недостатки, определяющие ее применение для конкретного проекта.

В анализируемой литературе встретилось несколько подходов к сравнению различных моделей разработки ПО. В одном из них ключевыми критериями выбора моделей являются отношение моделей к каскадной или итерационной группе разработки, а также степень формальности разработки рабочих материалов. Примеры использования этих показателей для сравнения методологий можно найти, например, в работе [10].

Другой подход используется в книге Э. Брауда [11]. В ней сравниваются процессы разработки ПО по четырем факторам: легкость контроля документации; возможность взаимодействия с заказчиком; поддержание хорошего проектирования и сбор метрических данных, полученных в хо-

следующих моделях: *каркасных; сборочного программирования; исследовательского программирования*. В работе [6] выделено пять основных подходов к организации процесса создания и использования ПС: *водопадный подход, исследовательское программирование, прототипирование, формальные преобразования и сборочное программирование*. В работе [7] подробно описаны следующие модели: *общепринятая, классическая итерационная, каскадная, объектно-ориентированная и модель Гантера "фазы-функции"*.

Хороший обзор методологий программирования приведен в работе [8], где описаны следующие методологии: *XP; семейство методологий Crystal, Open Source; адаптивная разработка (ASD) по Джиму Хайсмиту; SCRUM; Feature Driven Development и DSDM (Dynamic System Development Method)*. Все их можно отнести к гибким методологиям разработки программного обеспечения (известным также под названием "облегченных" или "адаптивных" методологий). Подробный обзор методологий *RUP, MSF, RAD, XP, Iconix, Crystal, ASD, Scrum, FDD, DSDM* дан в работе [5].

Легко обнаружить, что в соответствующей литературе приводятся разные подходы к классификации моделей разработки ПО, перечень рассматриваемых моделей различен, модели с одинаковыми названиями интерпретируют по-разному.

В целях объединения и систематизации информации о наиболее распространенных моделях разработки ПО предлагается их классификация (см. рисунок), которая получена на основе анализа научной литературы. Модели разработки ПО разделены на следующие три группы: типовые, промышленные и другие [9]. Рассмотрим более подробно каждую из них.

де проекта. Выбор осуществляется только из трех процессов разработки ПО (водопадного, спирального и инкрементного) и сводится к решению вопроса о числе итераций.

Количественное сравнение моделей разработки ПО используется в работе [4]. Специально для выбора модели разработаны 32 критерия, сгруппированных по четырем категориям: это характеристики требований участников команды разработчиков, коллектива пользователей, типа проектов и рисков.

В работе [12] уточняется приведенная выше классификация проектов [4], предназначенная для выбора модели разработки программных средств. В ней для сравнения к имеющимся 32 критериям добавлены еще три: масштабность; стабильность и критичность продукта проекта.

Для выбора оптимальной модели разработки ПО предлагается использовать систему критериев, представленную в табл. 1. Она основана на подходе, приведенном в работе [4]. В ходе анализа оказалось, что 20 скалярных критериев целесообразно объединить в пять векторных компонент.

Первая векторная компонента — *характеристики требований*, классифицирует проекты в зависимости от возможности изменения требований пользователей и заказчика к разрабатываемому продукту.

Вторая векторная компонента — *характеристики команды разработчика*, определяет проекты с учетом особенностей коллектива разработчиков и учитывает тот факт, что от данных особенностей зависит успех реализации проекта.

Третья векторная компонента — *характеристики пользователей и заказчика*, позволяет учесть степень участия пользователей (заказчиков) в процессе разработки.

Четвертая векторная компонента — *характеристики типов проектов и рисков*, отражает сложность и масштаб проекта, достаточность ресурсов для его исполнения и т. д.

Пятая векторная компонента — *характеристики процесса разработки*, отражает особенности процесса разработки, например, учет требований российских стандартов серий 19.XXX и 34.XXX, степень формализации разработки рабочих материалов и др.

С использованием методов теории принятия решений [13—17] предлагаемая система критериев позволяет проводить оценку множества различных моделей разработки ПО. Для такой оценки широкое применение нашли различные методы ранжирования [14, 16, 18].

Как показывает анализ системы критериев, используемых для сравнения различных моделей разработки ПО, задача принятия решений сводится в данном случае к задаче гипервекторного ранжирования [19—21]. Рассмотрим постановку и метод решения такой задачи.

3. Постановка и метод решения задачи гипервекторного ранжирования

Введем необходимые в дальнейшем обозначения:

$S = \{S_\alpha, \alpha = \overline{1, n}\}$ — множество моделей разработки ПО (моделей);

$K(S_\alpha) = \{K_\varepsilon(S_\alpha), \varepsilon = \overline{1, E}\}$ — множество многовекторных компонент $K_\varepsilon(S_\alpha)$, характеризующих модель $S_\alpha \in S$;

$K_\varepsilon(S_\alpha) = \{K_{\varepsilon j}(S_\alpha), j = \overline{1, r_\varepsilon}\}$ — множество векторных компонент, характеризующих модель S_α;

$K_{\varepsilon j}(S_\alpha) = \{K_{\varepsilon ji}(S_\alpha), i = \overline{1, r_{\varepsilon j}}\}$ — множество скалярных компонент, характеризующих модель S_α;

$A = \{a_\varepsilon, \varepsilon = \overline{1, E}\}$ — множество коэффициентов важности многовекторных компонент, причем $\sum\limits_{\varepsilon = 1}^{E} a_\varepsilon = 1$;

$A_\varepsilon = \{a_{\varepsilon j}, j = \overline{1, r_\varepsilon}\}$ — множество коэффициентов важности векторных компонент, причем $\sum\limits_{j = 1}^{r_\varepsilon} a_{\varepsilon j} = 1$, $\varepsilon = \overline{1, E}$;

$A_{\varepsilon j} = \{a_{\varepsilon ji}, i = \overline{1, r_{\varepsilon j}}\}$ — множество коэффициентов важности скалярных компонент, причем $\sum\limits_{i = 1}^{r_{\varepsilon j}} a_{\varepsilon ji} = 1$, $j = \overline{1, r_\varepsilon}$, $\varepsilon = \overline{1, E}$;

S^π — множество эффективных (Парето-оптимальных) моделей с числом элементов n^π, $S^\pi \in S$;

$S_p^0 \in S^\pi$ — модели, которые входят в множество эффективных решений, $p \in \{\overline{1, n}\}$;

$D(S_\alpha) = \{D_i(S_\alpha), i = \overline{1, M}\}$ — множество ограничений, накладываемых на модели, причем должно выполняться условие

$$D_i(S_\alpha) \leqslant D_i^0, \quad i = \overline{1, M}, \tag{1}$$

где D_i^0 — допустимое значение i-го ограничения, M — число ограничений;

$P = \{S_{k_1}^0, S_{k_2}^0, ..., S_{k_{n^\pi}}^0\}$ — упорядоченное множество эффективных моделей (кортеж Парето); элементы кортежа ранжированы в соответствии с решающими правилами так, что выполняется условие

$$S_{k_1}^0 > S_{k_2}^0 > ... > S_{k_i}^0 > ... > S_{k^\pi}^0,$$

где ">" — знак отношения доминирования, $k_i \in \{1, 2, ..., n\}$. Длина кортежа равна n^{π}.

Допустим, известны множества A, A_{ε}, $A_{\varepsilon j}$, S, $K_{\varepsilon j}(S_{\alpha})$, $D(S_{\alpha})$, ($\alpha = \overline{1, n}$; $\varepsilon = \overline{1, E}$; $j = \overline{1, r_{\varepsilon}}$), решающие правила. Требуется найти кортеж Парето P, для элементов которого справедливо равенство

$$K(S_p^0) = \min_{S_{\alpha} \in S} K(S_{\alpha}),\ S_p^0 \in P, \qquad (2)$$

и выполняется условие (1).

Пусть нам известен метод ранжирования систем по совокупности скалярных компонент $K_{\varepsilon j}(S_{\alpha\varepsilon})$, $\varepsilon = \overline{1, E}$, $j = \overline{1, r_{\varepsilon}}$, например метод "же-

Таблица 1

Система критериев, используемая для сравнения различных моделей разработки ПО

Критерии и категории, характеризующие модели разработки ПО	Обозначение	Примечание
Характеристики требований к проекту	K_1	
Могут ли все требования к процессу не задаваться сразу в начале ЖЦ (до начала проектирования)?	K_{11}	0 — нет; 1 — да
Могут ли требования изменяться в процессе итерации?	K_{12}	0 — нет; 1 — возможно; 2 — да
Приветствуется ли изменение требований даже на поздних этапах разработки ради достижения заказчиком конкурентного преимущества?	K_{13}	0 — нет; 1 — да
Необходимо ли демонстрировать требования в целях их определения и/или для проверки концепции?	K_{14}	0 — нет; 1 — да
Характеристики команды разработчиков	K_2	
Являются ли проблемы предметной области проекта новыми для большинства разработчиков?	K_{21}	0 — нет; 1 — да
Являются ли инструменты, используемые в проекте, новыми для большинства разработчиков?	K_{22}	0 — да; 1 — нет
Является ли команда разработчиков самоорганизующейся?	K_{23}	0 — нет; 1 — да
Могут ли меняться роли участников проекта?	K_{24}	0 — нет; 1 — да
Характеристики пользователей и заказчика	K_3	
Частота отслеживания заказчиком хода выполнения проекта	K_{31}	Задан интервалами значений. 0 — реже 1 раза в 2 месяца; 1 — раз в 1—2 месяца; 2 — раз в 1— 2 недели; 3 — ежедневно/раз в несколько дней.
Вовлечены ли пользователи в фазы жизненного цикла?	K_{32}	0 — нет; 1 — частично; 2 — на всех этапах
Характеристики типов проектов и рисков	K_4	
Будет ли проект являться расширением существующей системы?	K_{41}	0 — нет; 1 — да
Будет ли финансирование проекта стабильным на всем протяжении жизненного цикла?	K_{42}	0 — нет; 1 — да
Ожидается ли длительная эксплуатация продукта?	K_{43}	0 — нет; 1 — возможно; 2 — да
Будет ли продукт изменяться (возможно, с применением непредвиденных методов), на этапе сопровождения (стабильность продукта)?	K_{44}	Задан интервалами значений. 0 — маловероятно; 1 — возможно; 2 — постоянно
Масштаб разрабатываемого продукта	K_{45}	Задан интервалами значений. 0 — малый; 1 — средний; 2 — большой
Критичность разрабатываемого продукта, влияющая на уровень безопасности ПО	K_{46}	Задан интервалами значений. 0 — потеря комфорта; 1 — потеря денег; 2 — потеря больших денег и бизнеса; 3 — потеря жизни
Характеристики процесса разработки	K_5	
Будет ли проектирование и конструирование отдельно каждого свойства системы/пошаговая детализация?	K_{51}	0 — нет; 1 — возможно; 2 — да
Будут ли учитываться требования ГОСТов серий 19 и/или 34?	K_{52}	0 — нет; 1 — да
Степень формализации разработки рабочих материалов	K_{53}	0 — низкая (работающее ПО выше всеобъемлющей документации); 1 — высокая (детальная документация, большое число документов, формальные процедуры рецензирования); 2 — средняя (документация в основном сконцентрирована на конечном продукте)
Необходимо ли как можно быстрее получить первые версии работающей программы?	K_{54}	0 — нет; 1 — да

сткого" ранжирования [19—21]. После его применения будут построены частные кортежи Парето, которые позволят однозначно определить расположение модели разработки ПО S_α относительно других моделей по каждой векторной компоненте. Причем выявляются как доминирующие (доминируемые), так и эквивалентные модели. Это позволяет придать всем векторным компонентам некоторые числа, значения которых зависят от расположения моделей: для доминируемых моделей эти значения больше, чем для доминирующих, а для эквивалентных моделей разработки ПО эти значения будут равными. Назовем такие числа *псевдозначениями* (рангами) векторных компонент. Введение таких псевдозначений позволяет вновь применить метод "жесткого" ранжирования (число обращений к методу будет равно числу многовекторных компонент, т. е. E раз) и построить частные кортежи Парето. В результате решения задачи получаем расположение моделей по совокупности многовекторных компонент $K_\varepsilon(S_\alpha)$, $\varepsilon = \overline{1, E}$. Это, в свою очередь, позволяет обоснованно ввести псевдозначения многовекторных компонент и вновь обратиться к методу "жесткого" ранжирования. В итоге и будет построен искомый кортеж Парето.

Более подробно методы ранжирования рассмотрены в работах [19—21].

Методика решения задачи гипервекторного ранжирования

1. Провести анализ исходной информации, формирование критериев оценок моделей, определить коэффициенты важности критериев.
2. Провести ранжирование моделей разработки ПО по множеству скалярных компонент каждой векторной компоненты.
3. Определить псевдозначения векторных компонент.
4. Провести ранжирование моделей по множеству векторных компонент (построить частные кортежи Парето).
5. Определить псевдозначения многовекторных компонент.
6. Построить кортеж Парето.
7. Провести анализ результатов решения.
8. В случае необходимости уточнить исходные данные. Перейти к шагу 2. В противоположном случае перейти к шагу 9.
9. Конец решения.

4. Численный пример

С использованием метода гипервекторного ранжирования найдем решение прикладной задачи. Допустим, необходимо построить упорядоченное множество эффективных моделей разработки ПО (кортеж Парето) для проекта, предполагающего автоматизировать процесс некоторой гипотетической системы управления коммуникациями в организации.

Таблица 2

Значения критериев, характерные для моделей разработки ПО

Критерии	Модели разработки ПО									
	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
Характеристики требований к проекту (K_1)										
K_{11}	0	0	1	1	1	0	1	0	1	1
K_{12}	0	0	1	0	2	0	1	0	0	1
K_{13}	0	0	0	0	1	0	0	0	0	1
K_{14}	0	0	1	0	1	1	0	0	0	1
Характеристики команды разработчиков (K_2)										
K_{21}	0	0	1	0	1	0	1	1	1	1
K_{22}	1	1	1	0	0	0	1	1	0	0
K_{23}	0	0	0	0	1	1	0	0	0	1
K_{24}	0	0	1	1	1	0	0	0	1	1
Характеристики пользователей и заказчика (K_3)										
K_{31}	0—1	0—1	0—1	1—2	2—3	2—3	1—2	0—1	1—2	1—3
K_{32}	0	0	0	0	1	1	1	0	0	2
Характеристики типов проектов и рисков (K_4)										
K_{41}	1	1	0	1	0	1	0	0	1	1
K_{42}	1	1	0	0	1	1	1	1	0	0
K_{43}	2	2	2	2	0	1	2	2	2	1
K_{44}	0—1	0—1	0—2	0—1	0—1	0—1	0—1	0—1	0—1	1—2
K_{45}	0—2	1—2	1—2	1—2	1—2	0—1	1—2	1—2	0—1	0—1
K_{46}	0—2	2—3	2—3	1—2	0—1	0—1	2—3	1—2	2—3	0—3
Характеристики процесса разработки (K_5)										
K_{51}	0	0	0	2	1	0	0	0	2	1
K_{52}	1	1	0	0	0	0	1	0	0	0
K_{53}	1	1	1	2	2	2	1	1	2	0
K_{54}	0	0	0	0	1	1	1	0	0	1

Значения критериев для 10 возможных моделей разработки ПО приведены в табл. 2.

В табл. 2 приняты следующие обозначения: S_1 — каскадная модель; S_2 — *V*-образная модель; S_3 — спиральная модель, модель *Microsoft Solutions Framework* (MSF); S_4 — инкрементная (пошаговая) модель; S_5 — модель быстрого прототипирования; S_6 — модель *Rapid Application Development* (RAD); S_7 — модель *Ration Unified Processing (RUP)*; S_8 — модель *Oracle Custom Development Method (CDM)* классический; S_9 — модель *Cleanroom Software Engineering*; S_{10} — модели гибких методологий.

Модели из группы "другие" не были включены в табл. 2 по причине их неприменимости для разработки большинства проектов.

Решение

1. Найдем коэффициенты важности критериев.

С этой целью воспользуемся модифицированным методом анализа иерархий Т. Саати [16]. Результаты решения приведены в табл. 3.

Таблица 3

Коэффициенты важности критериев

j	a_{1j}	a_{2j}	a_{3j}	a_{4j}	a_{5j}	a_j
1	0,536	0,154	0,333	0,465	0,233	0,132
2	0,268	0,051	0,667	0,232	0,699	0,132
3	0,107	0,026		0,052	0,039	0,033
4	0,089	0,769		0,116	0,029	0,044
5				0,077		0,659
6				0,058		

2. *Решим задачу гипервекторного ранжирования*. В результате получим следующий кортеж Парето $P = <S_7, S_5, S_3, S_{10}, S_2>$. Модели S_1, S_4, S_6, S_8, S_9 оказались неэффективными. Предпочтение следует отдать модели *Ration Unified Processing*, хотя не исключено, что лицо, принимающее решение, может сделать выбор в пользу модели быстрого прототипирования или модели *Microsoft Solutions Framework*, которые входят в первую тройку.

После выбора модели необходима ее адаптация к конкретному проекту с учетом особенностей организации, типа выполняемых работ, навыков и используемых инструментальных средств.

Заключение

В статье рассмотрена возможная прикладная задача обоснованного выбора моделей разработки программного обеспечения. Проведенный анализ современной научной литературы позволил: провести классификацию таких моделей; предложить систему критериев, с помощью которой можно было бы осуществлять выбор варианта модели разработки ПО для его практического внедрения. Рассмотрены три группы моделей: типовые, промышленные и другие. Критерии целесообразно представить в виде пяти векторных компонент. Особенностью ряда скалярных критериев является то, что они заданы в интервальном виде. Задача выбора эффективных вариантов моделей разработки ПО (построения кортежей Парето) сводится в этом случае к задаче гипервекторного ранжирования. Использование предлагаемого подхода к сравнению моделей разработки ПО и выбору оптимального варианта для конкретного проекта позволяет заказчикам, системным аналитикам и ведущим специалистам, связанным с разработкой программных продуктов, быть более объективными в выборе соответствующих моделей при решении практических задач.

Список литературы

1. **ГОСТ Р ИСО/МЭК 12207—1999.** "Информационная технология. Процессы жизненного цикла программных средств". М.: Изд-во стандартов, 1999.

2. **Проект ГОСТ Р ИСО/МЭК 12207.** "Информационная технология. Процессы жизненного цикла программных средств". М., 2009.

3. **ГОСТ Р ИСО/МЭК ТО 15271—2002**. "Информационная технология. Руководство по применению ГОСТ Р ИСО/МЭК 12207". М.: Стандартинформ, 2002.

4. **Фатрелл Р. Т., Шафер Д. Ф., Шафер Л. И**. Управление программными проектами: достижение оптимального качества при минимуме затрат / Пер. с англ. М.: Издательский дом "Вильямс", 2003. 1136 с.

5. **Принципы** разработки программных продуктов" системы дистанционного обучения сети ВИТ-центров. Учебный курс. М.: Изд. ФГУ ГНИИ ИТТ "Информика". URL: http://elearnin.informika.ru/content/public/met.

6. **Зубкова Т. М.** Технология разработки программного обеспечения: учебное пособие. Оренбург: Изд. ГОУ "Оренбургский государственный университет", 2004. 101 с.

7. **Скопин И. Н.** Модели жизненного цикла программного обеспечения // Сб. Новосибирская школа программирования. Перекличка времен. Новосибирск: Изд. Института систем информатики им А. П. Ершова СО РАН, 2004. С. 120—173.

8. **Фаулер М.** Новые методологии программирования. URL: http://www.maxkir.com/sd/newmethRUS.html.

9. **Федорец О. Н.** Модели разработки программного обеспечения // Доклады Академии военных наук. 2008. № 5 (34). С. 147—152.

10. **Кролл П., Крачтен Ф**. Rational Unified Process — это легко: руководство по RUP для практиков / Пер. с англ. Кудиц — Образ, 2004. 432 с.

11. **Брауде Э.** Технология разработки программного обеспечения. СПб.: Питер, 2004. 655 с.

12. **Бахтизин В. В., Глухова Л. А.** Выбор модели жизненного цикла разработки программных средств и систем на основе сводной таблицы критериев классификации проекта // Доклады Белорусского государственного университета информатики и радиоэлектроники. 2005. № 1. С. 110— 113.

13. **Дубов Ю. А., Травкин С. И., Якимец В. Н.** Многокритериальные модели формирования и выбора вариантов систем. М.: Наука, 1986. 296 с.

14. **Ларичев О. И.** Теория и методы принятия решений, а также хроника событий в волшебных странах: учебник. Изд. второе, перераб. и доп. М.: Логос, 2003. 392 с.

15. **Моисеев Н. Н.** Математические задачи системного анализа. — М.: Наука. Гл. ред. физ.-мат. лит., 1981. 488 с.

16. **Ногин В. Д.** Упрощенный вариант метода анализа иерархий на основе нелинейной свертки критериев // Журнал вычислительной математики и математической физики. 2004. Т. 44. № 7. С. 1259—1268.

17. **Подиновский В. В., Ногин В. Д.** Парето-оптимальные решения многокритериальных задач. 2-е изд., испр. и доп. М.: Физматлит, 2007. 256 с.

18. **Саати Т. Л.** Принятие решений. Метод анализа иерархий / Пер. с англ. М.: Радио и связь, 1993. 320 с.

19. **Сафронов В. В.** Проблемы проектирования сложных технических систем и некоторые пути их решения // Доклады Академии военных наук. 1999. № 1. С. 84—95.

20. **Сафронов В. В.** Гипервекторное ранжирование сложных систем // Информационные технологии. 2003. № 5. С. 23—26.

21. **Сафронов В. В.** Основы системного анализа: методы многокритериального ранжирования. Энгельс: Ред.-изд. центр ПКИ, 2007. 185 с.

УДК 681.124.2

С. Е. Суховерхов, аспирант,
e-mail: ssukhoverkhov@gmail.com,
О. Б. Штейнберг, ассистент,
Южный федеральный университет,
e-mail: olegsteinb@gmail.com

Автоматическое распараллеливание рекуррентных циклов с проверкой устойчивости

Рассматривается алгоритм автоматического распараллеливания рекуррентных циклов с линейной рекуррентной зависимостью. Доказывается устойчивость рассматриваемого распараллеливающего преобразования. Алгоритм программно реализован в "Открытой распараллеливающей системе" www.ops.rsu.ru.

Ключевые слова: *параллельные вычисления, рекуррентные циклы, теплицевы матрицы.*

Введение

В данной работе предложен алгоритм автоматического распараллеливания рекуррентных циклов и получены условия его устойчивости.

Рассматривается задача распараллеливания цикла:

$$
\begin{aligned}
&x[1] = h[1];\\
&x[2] = h[2];\\
&\ldots\\
&x[m] = h[m];\\
&for(i = m + 1;\ i < m + 1 + L;\ i = i + 1)\\
&\{\\
&x[i] = a_1x[i - 1] + a_2x[i - 2] + \ldots + a_mx[i - m];\\
&\}
\end{aligned}
\qquad (1)
$$

где $a_m \neq 0$.

Распараллеливание выполняется в расчете на вычислительную модель, состоящую из 2^k, $k \in N$, вычислительных устройств (ВУ). Представленный алгоритм можно использовать в случае архитектуры с общей памятью, распределенной памятью, а также и на специализированных параллельных вычислительных системах. От типа памяти, как и от других элементов архитектуры, зависит эффективность, но не зависит корректность и устойчивость алгоритма.

Задача о распараллеливании рекуррентных циклов рассматривалась в работах [1]—[4]. В статье [1] рассматривается алгоритм распараллеливания рекуррентных циклов с опережающим вычислением коэффициентов и указано, что описанное преобразование сохраняет устойчивость вычислений. В данной работе предлагается другой метод параллельного вычисления рекуррентных циклов. Основной идеей метода является сводимость данной задачи к решению системы линейных уравнений с теплицевой матрицей, которая может быть преобразована к блочной теплицевой двухдиагональной матрице. Доказано, что если исходный последовательный алгоритм вычислений был устойчив, то устойчивым будет и результирующий параллельный алгоритм. В работе [3] получены результаты более общие, чем в работе [4]. Предлагаемый в данной работе метод может быть скомбинирован с методом работы [3]. В работе [2] рассмотрен более узкий класс распараллеливаемых рекуррентных циклов (циклы, вычисляющие сумму элементов массива, и подобные им).

Алгоритм, описанный в этой статье, программно реализован в рамках проекта "Открытая распараллеливающая система" (ОРС) [5].

1. Алгоритм параллельного вычисления рекуррентного цикла

Не уменьшая общности, будем считать, что L кратно m. Если это не так, то применим преобразование "расщепление цикла", заменяя фрагмент (1) следующим фрагментом:

$$
\begin{aligned}
&x[1] = h[1];\\
&x[2] = h[2];\\
&\ldots\\
&x[m] = h[m];\\
&for(i = m + 1;\ i < m + 1 + L - (L\ div\ m);\ i = i + 1)\\
&\{\\
&\quad x[i] = a_1x[i - 1] + a_2x[i - 2] + \ldots + a_mx[i - m];\}\\
&\}\\
&for(j = m + 1 + L - (L\ div\ m);\ j < m + 1 + L;\\
&j = j + 1)\\
&\{\\
&\quad x[j] = a_1x[j - 1] + a_2x[j - 2] + \ldots + a_mx[j - m];\\
&\}
\end{aligned}
\qquad (2)
$$

Число итераций в первом цикле фрагмента (2) равно $L - (L\ div\ m)$, а во втором цикле, соответственно, — $L\ div\ m$. После этого преобразования к первому циклу можно применить рассматриваемый алгоритм. Второй цикл можно не распараллеливать, поскольку он содержит мало итераций.

Для описания предлагаемого в данной работе алгоритма приведем следующие рассуждения.

Применяя к циклу фрагмента (1) преобразование "раскрутка цикла", получим

$$
\begin{aligned}
&x[1] = h[1];\\
&x[2] = h[2];\\
&\ldots
\end{aligned}
$$

$x[m] = h[m];$
$x[m+1] = a_m x[1] + a_{m-1} x[2] + \ldots + a_1 x[m];$
$x[m+2] = a_m x[2] + a_{m-1} x[3] + \ldots + a_1 x[m+1];$
...
$x[m+L] = a_m x[L] + a_{m-1} x[L+1] + \ldots +$
$+ a_1 x[L+m-1].$

В результате получим код, который по сути описывает систему линейных алгебраических уравнений (СЛАУ). Данной СЛАУ соответствует следующая расширенная матрица:

$$\left(\begin{array}{cccccccccc|c}
1 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & h[1] \\
0 & 1 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & h[2] \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
0 & 0 & \ldots & 1 & 0 & 0 & \ldots & 0 & 0 & 0 & h[m] \\
-a_m & -a_{m-1} & \ldots & -a_1 & 1 & 0 & \ldots & 0 & 0 & 0 & 0 \\
0 & -a_m & \ldots & -a_2 & -a_1 & 1 & \ldots & 0 & 0 & 0 & 0 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
0 & 0 & \ldots & -a_m & -a_{m-1} & -a_{m-2} & \ldots & 1 & 0 & 0 & 0 \\
0 & 0 & \ldots & 0 & -a_m & -a_{m-1} & \ldots & -a_1 & 1 & 0 & 0 \\
0 & 0 & \ldots & 0 & 0 & -a_m & \ldots & -a_2 & -a_1 & 1 & 0
\end{array}\right)$$

Эту матрицу, в свою очередь, можно представить в виде блочной двухдиагональной матрицы:

$$\left(\begin{array}{cccccccc|c}
I & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & \bar{h} \\
-B & A & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\
0 & -B & A & 0 & \ldots & 0 & 0 & 0 & 0 \\
0 & 0 & -B & A & \ldots & 0 & 0 & 0 & 0 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
0 & 0 & 0 & 0 & \ldots & A & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \ldots & -B & A & 0 & 0 \\
0 & 0 & 0 & 0 & \ldots & 0 & -B & A & 0
\end{array}\right),$$

где $I = \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 & 0 \\ 0 & 0 & \ldots & 0 & 1 \end{pmatrix};$

$$\bar{h} = \begin{pmatrix} h[1] \\ h[2] \\ \ldots \\ h[m-1] \\ h[m] \end{pmatrix}; \quad A = \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 \\ -a_1 & 1 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ -a_{m-2} & -a_{m-3} & \ldots & 1 & 0 \\ -a_{m-1} & -a_{m-2} & \ldots & -a_1 & 1 \end{pmatrix};$$

$$B = \begin{pmatrix} a_m & a_{m-1} & \ldots & a_2 & a_1 \\ 0 & a_m & \ldots & a_3 & a_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & a_m & a_{m-1} \\ 0 & 0 & \ldots & 0 & a_m \end{pmatrix}.$$

Матрица A является нижнетреугольной с главной диагональю, состоящей из единиц, а значит — обратимой. В блочной матрице умножим слева все строки, кроме первой, на A^{-1}. Получим

$$\left(\begin{array}{cccccccc|c}
I & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & \bar{h} \\
-B^{(1)} & I & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\
0 & -B^{(1)} & I & 0 & \ldots & 0 & 0 & 0 & 0 \\
0 & 0 & -B^{(1)} & I & \ldots & 0 & 0 & 0 & 0 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
0 & 0 & 0 & 0 & \ldots & I & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \ldots & -B^{(1)} & I & 0 & 0 \\
0 & 0 & 0 & 0 & \ldots & 0 & -B^{(1)} & I & 0
\end{array}\right), \qquad (3)$$

где $B^{(1)} = A^{-1}B.$

Таким образом, видим, что задача решения СЛАУ (3) эквивалентна задаче вычисления исходного цикла (1). Кроме того, эта СЛАУ может быть решена с использованием следующего фрагмента кода:

```
x[1] = h[1];
...
x[m] = h[m];
for(i = m + 1; i < m + 1 + L; i = i + m)
{
x[i] = b(1)1,1 x[i − m] + b(1)1,2 x[i − m + 1] + ... +
+ b(1)1,m x[i − 1];                                          (4)
x[i + m − 1] = b(1)m,1 x[i − m] + b(1)m,2 x[i − m + 1] +
+ ... + b(1)m,m x[i − 1];
}
```

Теперь вычтем изо всех строк, кроме первой, предыдущую строку, умноженную слева на $B^{(1)}$. Получим

$$\left(\begin{array}{cccccccc|c}
I & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & \bar{h} \\
0 & I & 0 & 0 & \ldots & 0 & 0 & 0 & \overline{h_2} \\
-B^{(2)} & 0 & I & 0 & \ldots & 0 & 0 & 0 & 0 \\
0 & -B^{(2)} & 0 & I & \ldots & 0 & 0 & 0 & 0 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
0 & 0 & 0 & 0 & \ldots & I & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \ldots & 0 & I & 0 & 0 \\
0 & 0 & 0 & 0 & \ldots & 0 & -B^{(2)} & I & 0
\end{array}\right), \qquad (5)$$

где $B^{(2)} = -(B^{(1)})^2$, $\overline{h_2} = -B^{(1)}\overline{h}$.

Решение полученной СЛАУ (5) равносильно вычислению цикла фрагмента (1). Кроме того, эта СЛАУ может быть решена с использованием следующего фрагмента кода:

$x[1] = h[1];$
...
$x[m] = h[m];$
$x[m+1] = h_2[1];$
...
$x[2m] = h_2[m];$
$for(i = 2m + 1;\ i < 2m + 1 + L;\ i = i + 2m)$
{
$x[i] = b^{(2)}_{1,1}\, x[i - 2m] + b^{(2)}_{1,2}\, x[i - 2m + 1] + ... +$
$+ b^{(2)}_{1,m}\, x[i - m - 1];$
...
$x[i + m - 1] = b^{(2)}_{m,1}\, x[i - 2m] + b^{(2)}_{m,2}\, x[i - 2m + 1] +$
$+ ... + b^{(2)}_{m,m}\, x[i - m - 1];$
}
$for(j = 2m + 1;\ j < 2m + 1 + L;\ j = j + 2m)$
{
$x[j + m] = b^{(2)}_{1,1}\, x[j - m] + b^{(2)}_{1,2}\, x[j - m + 1] + ... +$
$+ b^{(2)}_{1,m}\, x[j - 1];$
...
$x[j + 2m - 1] = b^{(2)}_{m,1}\, x[j - m] + b^{(2)}_{m,2}\, x[j - m + 1] +$
$+ ... + b^{(2)}_{m,m}\, x[j - 1];$
}

(6)

Первый цикл этой программы работает с нечетными строчками блочной матрицы, а второй цикл — с четными строчками. Эта программа может быть параллельно выполнена на двух ВУ (каждое ВУ вычисляет по одному циклу).

Таким же образом существует возможность получить фрагмент кода, эквивалентный фрагменту (1), который, в свою очередь, может быть распараллелен на $P = 2^k$ ВУ:

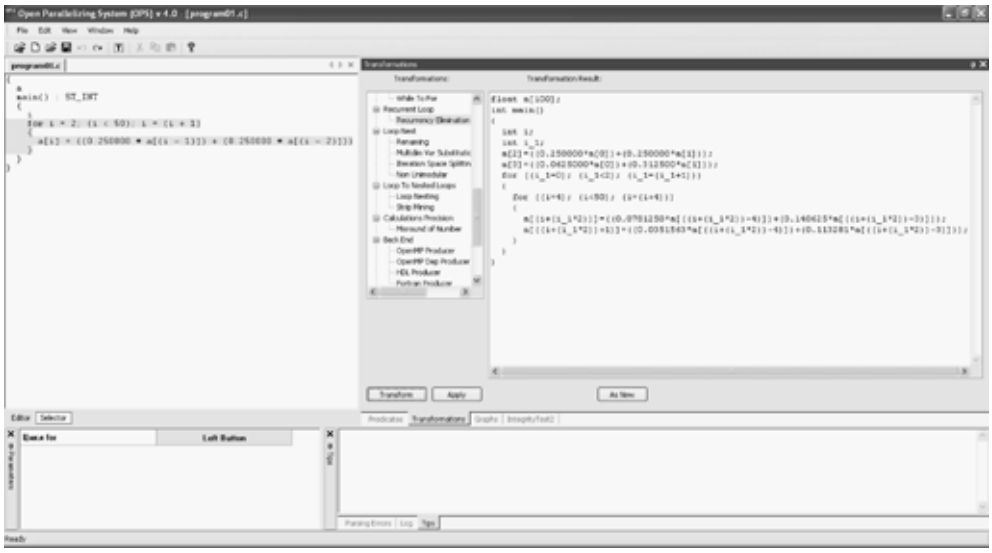
$x[1] = h[1];$
...
$x[m] = h[m];$
$x[m + 1] = h_2[1];$
...
$x[2m] = h_2[m];$
...
$x[P - 1)m + 1] = h_P[1];$
...
$x[Pm] = h_P[m];$
$for(i_1 = Pm + 1;\ i_1 < Pm + 1 + L;\ i_1 = i_1 + Pm)$
{
$x[i_1] = b^{(P)}_{1,1}\, x[i_1 - Pm] + b^{(P)}_{1,2}\, x[i_1 - Pm + 1] + ... +$
$+ b^{(P)}_{1,m}\, x[i_1 - (P-1)m - 1];$
...
$x[i_1 + m - 1] = b^{(P)}_{m,1}\, x[i_1 - Pm] + b^{(P)}_{m,2}\, x[i_1 - Pm +$
$+ 1] + ... + b^{(P)}_{m,m}\, x[i_1 - (P-1)m - 1];$
}
$for(i_2 = Pm + 1;\ i_2 < Pm + 1 + L;\ i_2 = i_2 + Pm)$
{
$x[i_2] = b^{(P)}_{1,1}\, x[i_2 - (P-1)m] + b^{(P)}_{1,2}\, x[i_2 - (P-1)m +$
$+ 1] + ... + b^{(P)}_{1,m}\, x[i_2 - (P-2)m - 1];$
...
$x[i_2 + m - 1] = b^{(P)}_{m,1}\, x[i_2 - (P-1)m] + b^{(P)}_{m,2}\, x[i_2 -$
$- (P-1)m + 1] + ... + b^{(P)}_{m,m}\, x[i_2 - (P-2)m - 1];$
}
...
$for(i_P = Pm + 1;\ i_P < Pm + 1 + L;\ i_P = i_P + Pm)$
{
$x[i_P] = b^{(P)}_{1,1}\, x[i_P - m] + b^{(P)}_{1,2}\, x[i_P - m + 1] + ... +$
$+ b^{(P)}_{1,m}\, x[i_P - 1];$
...
$x[i_P + m - 1] = b^{(P)}_{m,1}\, x[i_P - m] + b^{(P)}_{m,2}\, x[i_P - m + 1] +$
$+ ... + b^{(P)}_{m,m}\, x[i_P - 1];$
}

(7)

Итак, получен алгоритм, выполняющий замену фрагмента (1) фрагментом программы, циклы которого могут вычисляться параллельно.

Представленный в данной работе алгоритм вместе с проверкой условия устойчивости вычисления программно реализован в проекте OPS4.

Ниже на рисунке представлен скриншот программной реализации алгоритма в OPS4. Слева показан исходный рекуррентный цикл, справа — фрагмент программы, полученный в результате применения алгоритма. Далее к полученному фрагменту можно применить преобразование "разрезание цикла" (*Loop Distribution*), после чего получатся два цикла, которые можно выполнять параллельно на различных ВУ.



Скриншот программной реализации алгоритма в OPS4

2. Устойчивость рекуррентных циклов

Рассмотрим следующий рекуррентный цикл:

$$
\begin{aligned}
&x[1] = h_1;\\
&\ldots\\
&x[m] = h_m;\\
&for(i = m + 1;\ i < L - r + 1;\ i = i + r)\\
&\{\\
&x[i + 1] = f_{11}x[i - 1] + \ldots + f_{1m}x[i - m];\\
&\ldots\\
&x[i + r] = f_{r1}x[i - 1] + \ldots + f_{rm}x[i - m];\\
&\}
\end{aligned}
\qquad (8)
$$

Начальными данными в задаче вычисления цикла (8) являются компоненты вектора $h = (h_1, h_2, \ldots, h_m) \in R^m$, выходными данными — вектор $x = (x[1], x[2], \ldots, x[L]) \in R^L$. Если малое возмущение входных данных приводит к малому возмущению выходных, то цикл будем называть устойчивым. Дадим более строгое определение.

Определение. Будем считать, что вычисление цикла вида (8) устойчиво, если существует такое число $M > 0$, что для любого вектора возмущений $\delta h \in R^m$ справедливо неравенство:

$$\sup_{L \in N} \|x - \tilde{x}\|_L < M\|\delta h\|_m,$$

где x, $\tilde{x}$ — результаты вычислений цикла с исходными и возмущенными начальными данными соответственно.

Здесь $\|x\|_n = \max\limits_{i = 1 \ldots n} |x_i|$ обозначает максимум-норму вектора $x \in R^n$.

Несложно заметить, что результат вычислений x цикла (8) линейно зависит от вектора входных данных h. Тогда можно дать более простое определение устойчивости.

Определение. Будем считать, что цикл (8) устойчив, если существует такое число $M > 0$, что для любого вектора начальных данных h справедлива оценка

$$\sup_{L \in N} \|x\|_L < M\|h\|_m. \quad (9)$$

Частными случаями цикла (8) являются: цикл (1) при $r = 1$ и цикл (4) при $r = m$. Сформулируем критерии устойчивости этих рекуррентных циклов.

Определение. Характеристическим полиномом цикла (1) будем называть:

$$p(z) = z^m - a_1 z^{m-1} - a_2 z^{m-2} - \ldots - a_m.$$

Теорема 1. Вычисление цикла (1) устойчиво в том и только в том случае, когда все корни характеристического многочлена $p(z)$ принадлежат замкнутому единичному кругу комплексной плоскости, причем корни, принадлежащие границе единичного круга, просты.

Доказательство: сводится к доказательству устойчивости разностного уравнения ([6] гл. 13, § 3).

Рассмотрим векторную форму записи цикла (4):

$$
\begin{aligned}
&X^{(0)} = h;\\
&for(k = 1;\ k < L + m + 1;\ k + +)\\
&\{\\
&X^{(k)} = B^{(1)}X^{(k-1)};\\
&\}
\end{aligned}
$$

где $X^{(k)} = (x[(k-1)m + 1], \ldots, x[km]) \in R^m$. Отсюда видно, что $X^{(K)} = B^{(1)}X^{(k-1)} = (B^{(1)})^k h$.

Определение. Матрицу $B^{(1)}$ будем называть характеристической матрицей цикла (4).

Определение. Обозначим через $\|B\|_{n,\,n} = \sup\limits_{Y \neq 0} \dfrac{\|BY\|_n}{\|Y\|_n}$ норму линейного оператора, действующего в R^n.

Теорема 2. Следующие условия равносильны:

а) цикл (4) устойчив;

б) $\sup\limits_{k \in N} \|(B^{(1)})^k\|_{m,\,m} < \infty$; (10)

в) все собственные числа характеристической матрицы лежат в замкнутом единичном круге комплексной плоскости, причем собственным

числам, лежащим на единичной окружности, в жордановой нормальной форме соответствуют только жордановы клетки размера 1.

Доказательство. Докажем эквивалентность а) и б). Предположим, что условие б) не выполнено, т. е. $\sup\limits_{k \in N} \|(B^{(1)})^k\|_{m,\,m} = \infty$. Тогда в соответствии с принципом равномерной ограниченности для линейных операторов [8, стр. 123] существует такой вектор $\tilde{h} \in R^m$, что $\lim\limits_{k \to \infty} \|(B^{(1)})^k \tilde{h}\|_m = \infty$. Это означает, что цикл (4) неустойчив.

Пусть теперь $M = \sup\limits_{k \in N} \|(B^{(1)})^k\|_{m,\,m} < \infty$. Тогда при любом векторе начальных данных $h \|X^{(k)}\|_m = \|(B^{(1)})^k h\|_m \leqslant M\|h\|_m$.

Эквивалентность б) и в) следует из того, что условие (10) выполняется тогда и только тогда, когда $\sup\limits_{k \in N} \|(\text{ЖНФ}(B^{(1)}))^k\|_{m,\,m} < \infty$, где ЖНФ($B^{(1)}$) — жорданова нормальная форма матрицы $B^{(1)}$ (о жордановых нормальных формах см., например [7], гл. 3) и формулы возведения в степень жордановых клеток.

3. Устойчивость преобразований фрагментов программ, содержащих рекуррентные циклы

Определение. Будем считать, что фрагмент программы, не содержащий циклических участков, кроме циклов вида (8), устойчив, если устойчив каждый цикл, содержащийся в этом фрагменте.

В разделе 1 настоящей работы описан алгоритм, преобразующий фрагмент программы (1) в эквивалентные ему фрагменты (4) и (7). Соответствующие преобразования будем обозначать соответственно (1) → (4) и (1) → (7).

Определение. Будем считать, что преобразование фрагментов программ устойчиво, если из устойчивости исходного фрагмента программы следует устойчивость фрагмента программы, полученного в результате преобразования.

Теорема 3. Преобразование (1) → (4) устойчиво.

Доказательство. Доказательство опирается на следующую лемму.

Лемма 1. Пусть $q_1, q_2, ..., q_r$ — все различные корни полинома $p(z)$ кратностей $K_1, K_2, ..., K_r$ соответственно. Тогда $q_1^m, q_2^m, ..., q_r^m$ — все различные собственные числа матрицы $B^{(1)}$ кратностей $K_1, K_2, ..., K_r$, причем каждому собственному числу кратности K_i в жордановой нормальной форме матрицы соответствует единственная жорданова клетка размера K_i.

Доказательство леммы. Рассмотрим сопровождающую матрицу (матрицу Фробениуса) характеристического многочлена $p(z)$:

$$S = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ a_1 & a_2 & a_3 & \dots & a_{m-1} & a_m \end{pmatrix}.$$

Непосредственным умножением несложно убедиться, что $(SX^{(k)})_j = x[(k-1)m + j + 1]$ для всех $1 \leqslant j \leqslant m - 1$. Аналогично получаем, что

$$(SX^{(k)})_m = a_m x[km] + \dots + a_1 x[(k-1)m + 1] = x[km + 1].$$

Тогда $X^{(k)} = S^m X^{(k-1)}$. Но в то же время $X^{(k)} = B^{(1)} X^{(k-1)}$. Поскольку матрицы $B^{(1)}$ и S^m обратимы, а формулы справедливы для любого вектора начальных данных, это означает, что $B^{(1)} = S^m$. Характеристический многочлен матрицы S совпадает с ее минимальным аннулирующим многочленом и равен $p(z)$ (см. [7] гл. 3). С учетом изложенного все корни $p(z)$ и только они являются собственными числами матрицы S, причем каждому числу в жордановой нормальной форме матрицы соответствует единственная жорданова клетка. Лемма доказана.

Устойчивость преобразования (1) → (4) теперь очевидным образом следует из теоремы 1, леммы 1 и теоремы 2.

Теорема 4. Преобразование (1) → (7) устойчиво.

Доказательство. Заметим, что каждый из $P = 2^k$ циклов фрагмента (7) имеет вид (4) с характеристической матрицей $(B^{(1)})^k$. Очевидно, что условие (10) гарантирует устойчивость каждого из этих циклов. Это условие выполнено в силу устойчивости цикла (1) и теоремы 3. Тогда преобразование (1) → (7) устойчиво.

Список литературы

1. **Пархи К.** Методы преобразования алгоритмов для параллельных процессоров // ТИИЭР. 1989. Т. 77. № 12. С. 96—114.
2. **Pottenger W. M.** Theory, techniques and experiments in solving recurrences in computer programs // Thesis PhD in Computer Science, University of Illinois at Urbana-Champaign. 1997. May.
3. **Штейнберг О. Б.** Распараллеливание рекуррентных циклов // Высокие информационные технологии в науке и производстве: Тезисы докладов. Ростов-на-Дону: Изд-во Ростовского университета, 2006. С. 80.
4. **Штейнберг Б. Я.** Распараллеливание программных рекуррентных циклов // Информационные технологии. 2004. № 4. С. 16—23.
5. **Открытая** распараллеливающая система. — URL: www.ops.rsu.nu.
6. **Крылов В. И.** Приближенное вычисление интегралов. М.: Физматлит, 1959. 327 с.
7. **Хорн Р., Джонсон Ч.** Матричный анализ. М.: Мир, 1989. 656 с.
8. **Треногин В. А.** Функциональный анализ. М.: Физматлит, 2002. 488 с.

УДК 004.4’234; 004.514

А. С. Зуев, канд. техн. наук, доц.,
e-mail: Zuev_Andrey@mail.ru,
Петров Ю. И., лаборант,
Московский государственный университет
приборостроения и информатики

Описание модернизации дерева папок проводника Windows Explorer*

Представлено описание модернизированного дерева папок проводника операционной системы Windows, приведены результаты сравнения с аналогичными элементами интерфейса других приложений, обоснованы эргономичность, эффективность и конкурентоспособность данной разработки.

Ключевые слова: *человеко-компьютерное взаимодействие, графический пользовательский интерфейс, эргономика программного обеспечения, проектирование графических интерфейсов, оптимизация графических интерфейсов, модификация дерева папок, дерево папок, навигация по логическим дискам и каталогам, навигация в сети Интернет.*

Введение

Неотъемлемой частью программного обеспечения (ПО), используемого в интерактивном режиме, является графический интерфейс (ГИ), позволяющий управлять работой программ. Широкое применение интерактивного ПО привело к необходимости улучшения и оптимизации графических интерфейсов. Качественный интерфейс может обеспечить увеличение эффективности применения программы, снижение стоимости ее сопровождения, уменьшение длительности обучения пользователей и т. п. Пользователи оценивают сложность эксплуатации программ на основе представления их функциональных возможностей в ГИ, который приобретает все большее значение как важное конкурентное преимущество ПО.

Графический интерфейс формирует условия работы пользователей и должен разрабатываться в целях достижения максимальной эффективности выполнения определенных задач управления работой программы. Заметим, что оценки сложности эксплуатации компьютерных программ (КП) пользователями могут быть субъективны, поэтому целесообразно оценивать эффективность ее ГИ. В качестве параметра (критерия), характеризующего эффективность интерфейса, могут рассматриваться суммарные затраты времени, требующиеся пользователю для выполнения конкретного набора операций. Данные затраты времени обусловлены необходимостью визуального выбора (поиска) требующихся элементов интерфейса на экране, а также выполнением воздействий на них с помощью некоторого средства манипулирования, например мыши.

В связи с ростом функциональных возможностей программ и информационных нагрузок на пользователей наряду с задачами проектирования и оптимизации интерфейса может рассматриваться задача его модернизации (совершенствования) [1]. Целью модернизации ГИ должна являться минимизация затрат времени пользователя на выполнение конкретных наборов операций, которая может приводить также к сокращению числа ошибок и повышению устойчивости внимания в процессе работы с КП.

Рассматриваемая в статье модель TreeExplorerA является модернизацией стандартного дерева папок проводника Windows Explorer — реализованы опции, повышающие эргономичность окон операционной системы (ОС), сокращающие сложность работы и затраты времени пользователя, а также снижающие необходимость одновременного использования нескольких окон ОС. В статье описаны основные функциональные возможности модели и ее настройки, приведены результаты сравнения с аналогами и подходы к оценке эффективности.

Представляемая модель не является дополнением к ОС Windows, но позволяет оценить удобства работы, обеспеченные предложенными техническими решениями, и сделать выводы о целесообразности реализации ее функциональных возможностей в окнах ОС. Модель зарегистрирована в отраслевом фонде алгоритмов и программ (свидетельство № 12083), а ее подробное описание — в Национальном фонде неопубликованных документов (номер государственной регистрации 50200900043).

Основными преимуществами модели TreeExplorerA являются:

- улучшенный интерфейс и дополнительные опции, повышающие эффективность работы с деревом папок;
- система закладок, позволяющая переходить к наиболее востребованным папкам без непосредственного выбора их адресов в дереве;
- возможность навигации в сети Интернет по FTP-серверам и Web-сайтам без использования дополнительного ПО;

* Работа выполнена при финансовой поддержке грантов Президента РФ молодым российским ученым-кандидатам наук (грант № МК-948.2008.9)

- реализация поиска папок и файлов на персональном и удаленных компьютерах непосредственно из дерева папок.

Описание функциональных возможностей модели

Интерфейс TreeExplorerA представлен на рис. 1. Ключевым элементом, реализующим основные функциональные возможности, является *Панель папок*.

Заголовок окна, панель навигации, адресная панель, файловый браузер и строка состояния соответствуют типовым элементам интерфейса окон ОС Windows. Панель папок содержит опции, являющиеся новшеством, преимуществом и отличительной особенностью модели TreeExplorerA. В таблице приведено краткое описание назначения элементов интерфейса панели папок.

Особенностью модели TreeExplorerA является объединение основных элементов дерева папок в выпадающий список, обеспечивающий:

- хранение наименований локальных и съемных дисков, системных и пользовательских закладок, а также Интернет-ссылок (*FTP-закладки* — папки на удаленных компьютерах, *HTTP-закладки* — страницы Web-сайтов) (рис. 2);
- отображение дерева только для выбранного элемента списка (рис. 3);
- переход между дисками, закладками, удаленными компьютерами и Web-сайтами с минимальными затратами времени и минимальным числом манипуляций мышью и клавиатурой.

Нажатием правой клавиши мыши на дереве папок вызывается контекстное меню опций. На рис. 4 представлено меню работы с папками персонального компьютера, меню работы с Web-закладками рассмотрено ниже.

Представленное на рис. 4 меню реализует следующие опции модели:

- Добавить закладку — вызывает окно добавления закладки с адресом, соответствующим адресу текущей папки, выбранной в дереве.
- Дерево — содержит опции развертывания и сворачивания как всего дерева, так и содержания текущей выбранной в нем папки.
- Открыть — обеспечивает вызов отдельного окна ОС Windows, отображающего содержание текущей выбранной папки.
- Менеджер закладок — вызывает окно *Менеджер закладок*.
- Настройки — вызывает окно настроек программы.
- Поиск — включает режим поиска папок и файлов.

Работа с моделью TreeExplorerA может заключаться в выборе адреса папки в дереве папок, навигации в сети Интернет, в работе с закладками, организации поиска, а также в изменении настро-

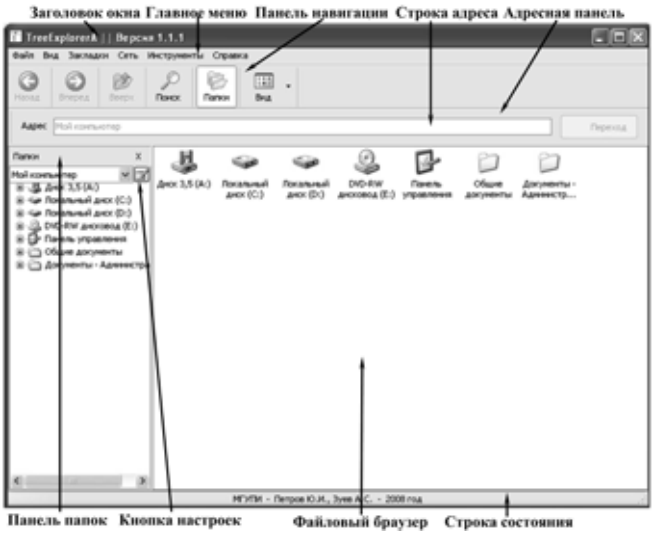


Рис. 1. Интерфейс модели TreeExplorerA

Элементы интерфейса панели папок

Элемент	Описание
Мой компьютер	Выпадающий список, содержащий наименования локальных и съемных дисков, закладок и Интернет-ссылок (Web-закладок)
	Кнопка вызова окна настроек (правая клавиша мыши) или менеджеры закладок (левая клавиша мыши)
Диск 3,5 (A:) Локальный диск (C:) Локальный диск (D:) PETROV (E:) CD-дисковод (K:) Панель управления	Дерево папок, отображающее пути к папкам в файловых системах персонального и удаленных компьютеров, а также к страницам Web-сайтов

Рис. 2. Выпадающий список панели папок

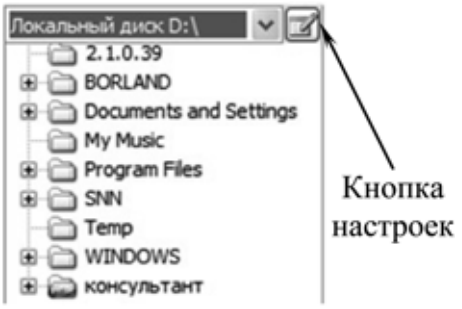


Рис. 3. Модернизированное дерево папок

ек. Функциональность модели ориентирована на повышение эргономичности и эффективности использования панели папок, при этом возможна работа с клавиатурой и мышью, в том числе сочетание данных средств манипулирования.

Для выбора адреса папки с помощью клавиатуры требуется:

1. Открыть выпадающий список нажатием клавиши *F11*.

2. Нажатием клавиш-стрелок *Вниз*, *Вверх* или *Вправо*, *Влево* выбрать требующийся элемент списка.

3. Перейти к содержанию выбранного элемента нажатием клавиши *Enter* или закрыть выпадающий список с помощью клавиши *Esc*.

4. Перейти на дерево папок нажатием клавиши *Tab* и продолжить выбор адреса требующейся папки с помощью клавиш-стрелок и клавиши *Enter*.

Для выбора адреса папки с помощью мыши требуется:

1. Выбрать элемент выпадающего списка нажатием ее левой клавиши.

2. Аналогично продолжить выбор адреса в дереве папок.

В модели TreeExplorerA реализована система закладок, позволяющая переходить к наиболее востребованным папкам без непосредственного выбора их адресов в дереве. По умолчанию предусмотрено пять системных закладок ОС Windows: *Мой компьютер, Рабочий стол, Сетевое окружение, Панель управления* и *Корзина*. Данные закладки отображаются в выпадающем списке панели папок после наименований дисков (см. рис. 2), для них не доступны опции переименования, удаления и изменения порядка перечисления.

Для управления пользовательскими закладками предназначено окно *Менеджер закладок* (рис. 5), вызвать которое можно следующими способами:

- выбрать пункт главного меню *Закладки — Управление закладками*;
- нажать горячую клавишу *F9*;
- нажать левую клавишу мыши на кнопке настроек (см. рис. 3);
- с помощью контекстного меню дерева папок (см. рис. 4).

Окно *Менеджер закладок* содержит таблицу с именами и адресами пользовательских закладок и предоставляет следующие опции управления ими:

- добавление, редактирование и удаление;
- изменение порядка перечисления в выпадающем списке — изменение очередности закладок в таблице с помощью кнопок *Вниз* и *Вверх*.

Система пользовательских закладок имеет следующие ограничения:

- запрещено добавлять закладки с одинаковыми именами или адресами;
- длина имени закладки ограничена 20 символами.

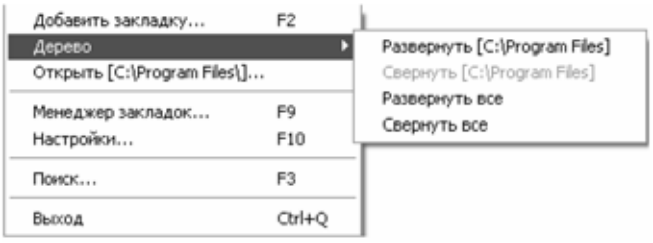


Рис. 4. Контекстное меню работы с папками персонального компьютера

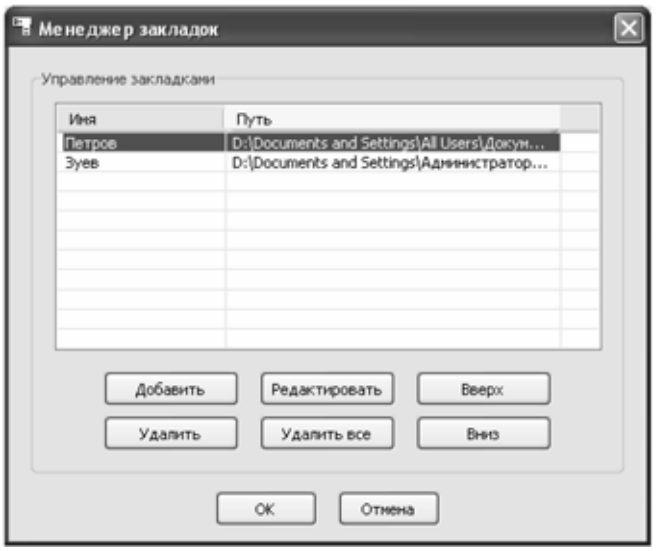


Рис. 5. Окно *Менеджер закладок*

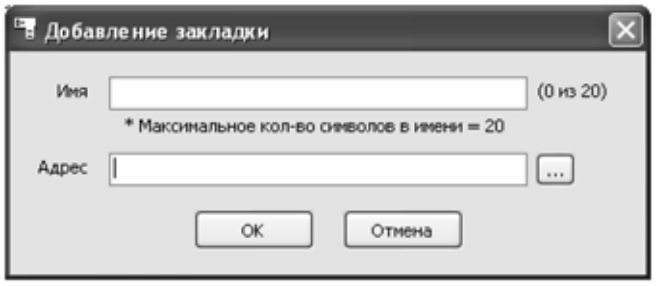


Рис. 6. Окно *Добавление закладки*

Для добавления пользовательской закладки с помощью менеджера закладок требуется выполнить следующие действия.

1. Нажатием кнопки *Добавить* вызвать окно *Добавление закладки* (рис. 6).

2. Ввести имя закладки и адрес соответствующей папки. Нажав кнопку "...", можно выбрать адрес с помощью стандартного диалогового окна.

3. При успешном добавлении закладки обновится таблица в менеджере закладок, а в случае ошибки будет выведено сообщение с указанием причины.

Для добавления пользовательской закладки, адрес которой выбран в дереве папок, требуется выполнить следующие действия:

1. Нажать горячую клавишу *F2* либо вызвать контекстное меню дерева папок и выбрать в нем пункт *Добавить закладку* (см. рис. 4).

2. В окне *Добавление закладки* ввести имя (Адрес задан автоматически).

3. При успешном добавлении закладки обновится содержание выпадающего списка панели папок.

В модели TreeExplorerA предусмотрена работа с Web-закладками, что позволяет осуществлять навигацию в сети Интернет по страницам Web-сайтов (HTTP-закладки) и FTP-серверам (FTP-закладки), адреса которых внесены в базу данных приложения. За настройку HTTP- и FTP-закладок отвечает окно *Менеджер сети* (рис. 7), вызвать которое можно следующими способами:

- выбрать пункт главного меню *Сеть — Менеджер сети*;
- нажать горячую клавишу *F8*;
- с помощью контекстного меню дерева папок (см. рис. 4).

Принципы работы с окном *Менеджер сети* аналогичны рассмотренным выше принципам работы с *Менеджером закладок*.

Для получения доступа к страницам Web-сайтов в выпадающем списке панели папок требуется выбрать элемент *HTTP-закладки*. В результате будет отображено дерево адресов главных и дочерних страниц Web-сайтов в одном из следующих форматов, определяемых настройками приложения.

1. [Имя HTTP-закладки] (Адрес HTTP-закладки) (рис. 8, *а*).

2. Имя HTTP-закладки (рис. 8, *б*).

Добавление в базу данных приложения адреса страницы Web-сайта выполняется аналогично рассмотренному выше добавлению пользовательской закладки. Соответствующее окно может быть вызвано из *Менеджера сети*, из раздела *Сеть* главного меню или с помощью контекстного меню дерева папок (рис. 9). Работа с папками на удаленных компьютерах (FTP-серверах) осуществляется аналогично работе со страницами Web-сайтов, в выпадающем списке панели папок требуется выбрать элемент *FTP-закладки*.

В программе TreeExplorerA предусмотрена возможность поиска папок и файлов на персональном и удаленных компьютерах непосредственно из дерева папок. Режим поиска (рис. 10) включается нажатием соответствующей кнопки на панели навигации, а также с помощью контекстного меню дерева папок.

Для выполнения поиска необходимо задать *текст*, содержащийся в искомом объекте, указать *тип* данного объекта, а также *место поиска*. Заме-

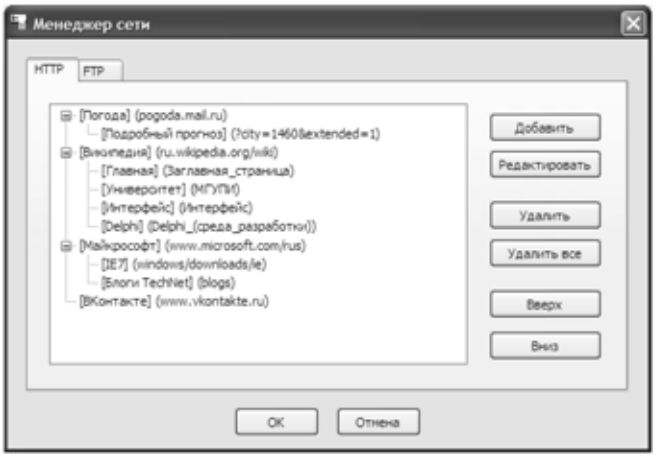


Рис. 7. Окно *Менеджер сети*

Рис. 8. HTTP-закладки:
а — имена и адреса;
б — только имена

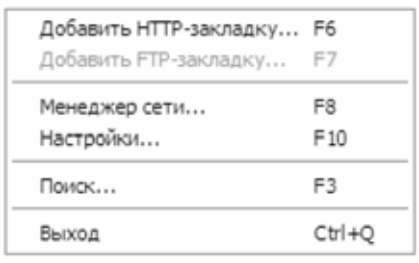


Рис. 9. Контекстное меню работы с Web-закладками

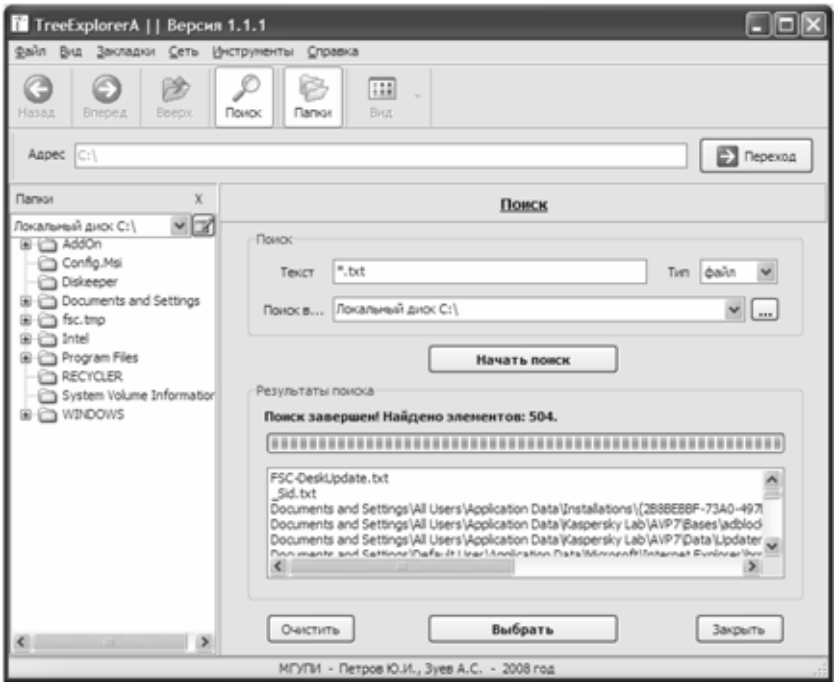


Рис. 10. Режим поиска

тим, что при реализации режима поиска в прикладном ПО число используемых параметров может быть увеличено.

Текст может состоять из любых символов, разрешенных ОС Windows для имени папки, а также специальных символов "*" и "?," обозначающих любое число и один произвольный символ. *Тип* искомого объекта определяет направление поиска — поиск папок, файлов или текста в файлах. *Место поиска* представляет собой путь в файловой системе компьютера и при включении рассматриваемого режима автоматически определяется адресом папки, выбранной в дереве. Изменить место поиска можно с помощью типового диалогового окна, вызываемого нажатием кнопки "...".

Результаты поиска сохраняются до момента выхода из данного режима или до запуска следующего поиска. Выбрать элемент из списка результатов можно воздействием левой клавиши мыши, а открыть соответствующий объект можно либо в

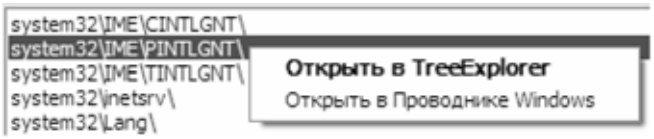


Рис. 11. Открытие объекта

Рис. 12. Панель навигации

окне ОС Windows, либо в браузере TreeExplorerA (рис. 11).

Если объект был открыт в браузере TreeExplorerA, то вернуться к результатам поиска можно нажатием кнопки *Поиск* на панели навигации. При наличии сохраненных результатов она имеет вид, представленный на рис. 12.

Выход из режима поиска с потерей результатов осуществляется нажатием кнопки *Закрыть* (см. рис. 10) или повторным нажатием кнопки Поиск.

Настройки программы

Настройки программы — это опции управления ее функциональными возможностями. Вызвать окно настроек можно одним из следующих способов:

- выбрать пункт главного меню *Инструменты — Настройки*;
- нажать правую клавишу мыши на кнопке настроек (см. рис. 3);
- нажать горячую клавишу *F10*;
- выбрать пункт *Настройки* контекстного меню дерева папок (см. рис. 4).

Настройки программы подразделяются на общие, связанные с деревом папок, закладками, работой в сети Интернет и поиском.

Общие настройки:

1. *Число позиций истории браузера* (от 3 до 31) определяет число возможных переходов к предыдущим адресам дерева папок.

2. *Шрифт* надписей на элементах интерфейса может быть оформлен различными стилями и кеглем.

Настройки дерева папок:

1. *Запретить изменение размеров* — запрещает изменение размеров панели папок с помощью курсора.

2. *Сворачивать пути при перемещении по истории* — при навигации по позициям истории браузера реализует сворачивание всех ветвей дерева, кроме адреса текущей выбранной папки.

3. *Сохранять путь дерева* — при изменении элемента, выбранного в выпадающем списке панели папок, в истории браузера сохраняется одна ветвь дерева, соответствующая адресу текущей папки.

4. *Сохранять позиции дерева* — при изменении элемента, выбранного в выпадающем списке панели папок, в истории браузера сохраняется все дерево.

5. *Число элементов в списке* — устанавливает число элементов, отображаемых в выпадающем списке панели папок без полосы прокрутки.

Настройки закладок:

1. *Переходить на закладку при добавлении* — определяет автоматический выбор добавляемой закладки в выпадающем списке, т. е. отображение вложенных папок и файлов в браузере, а также вложенных папок в дереве.

2. Может быть запрещено *добавление, удаление, редактирование* и *изменение порядка* перечисления закладок в выпадающем списке.

Настройки работы в сети Интернет:

1. *Разворачивать Web-закладки* — при выборе в выпадающем списке элементов FTP-закладки и HTTP-закладки обеспечивает автоматическое отображение в дереве всех папок на удаленных компьютерах и дочерних страниц Web-сайтов, адреса которых хранятся в базе данных приложения.

2. *Отображать только имена закладок* — позволяет отображать в дереве папок имена HTTP- и FTP-закладок без указания их адресов.

3. Может быть запрещено *добавление, удаление, редактирование* и *изменение порядка* перечисления адресов страниц Web-сайтов, удаленных компьютеров и папок в их файловых системах.

Настройки поиска:

Сортировать результаты поиска:

- *по порядку в дереве* — иерархически по отношению к корневому каталогу;
- *по алфавиту* — в алфавитном порядке по именам объектов;
- *по уровню вложенности* — по "глубине" вложенности в папки.

Сравнение с аналогами

В данном разделе представлены результаты сравнения функциональных возможностей дерева папок модели TreeExplorerA с аналогичными элементами интерфейса Windows Explorer, TotalCommander и UltraExplorer. Как показывает представленный ниже материал, TreeExplorerA обладает бóльшими функциональностью, эффективностью и эргономичностью навигации по файловым системам компьютеров и страницам Web-сайтов. Фактически TreeExplorerA не только предоставляет оптимальное сочетание опций и функциональных возможностей аналогичных элементов интерфейса перечисленных выше приложений, но и обеспечивает оригинальные опции использования закладок, поиска и работы в сети Интернет.

Windows Explorer. История операционной системы Windows началась в 1995 г. с выпуска Windows95 и на данный момент заканчивается выпуском в феврале 2008 г. ОС Windows Server 2008. Основой окон данных операционных систем является проводник Windows Explorer, обеспечивающий графический интерфейс доступа к папкам и файлам и фактически реализующий графическую оболочку файловой системы ОС Windows. В данном проводнике реализовано стандартное дерево папок (рис. 13), функциональные возможности и принципы использования которого оставались неизменными во всех версиях рассматриваемой операционной системы.

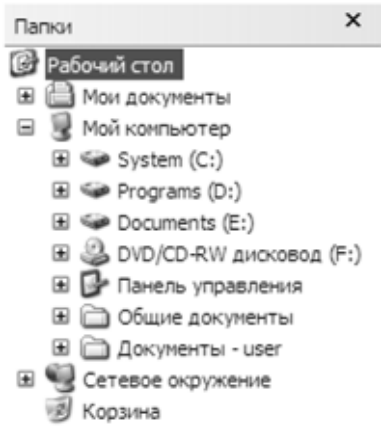


Рис. 13. Стандартное дерево папок проводника Windows Explorer

Главным недостатком дерева папок проводника Windows Explorer является его перегруженность информацией, а именно:

- корневым элементом всегда является папка *Рабочий стол*, хотя при работе с файловой системой она используется достаточно редко. В дереве отображаются также все папки, расположенные на *Рабочем столе*;
- в дереве отображаются наименования всех локальных и съемных дисков, а также системных папок, это требует действий по его реорганизации (например, при работе с несколькими дисками), рассеивает внимание пользователя и обуславливает применение полосы прокрутки.

Результаты сравнения с моделью TreeExplorerA:

- наименования локальных и съемных дисков, а также системных папок вынесены в выпадающий список, это минимизирует информационную нагрузку на дерево и требующиеся от пользователя действия по его реорганизации;
- выпадающий список может содержать пользовательские закладки, а дерево папок — адреса сети Интернет, это повышает эргономичность и функциональность приложения;
- работа с выпадающим списком сокращает необходимость одновременного использования нескольких окон ОС.

TotalCommander — условно-бесплатный файловый менеджер, в интерфейсе которого реализованы следующие технические решения:

- отказ от дерева папок как средства навигации по файловой системе;
- наличие выпадающего списка и панели дисков, позволяющих быстро переходить между ними (рис. 14);

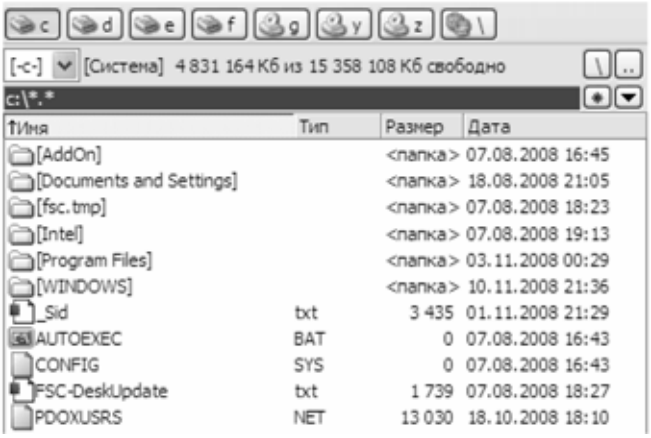


Рис. 14. Интерфейс файлового менеджера TotalCommander

Рис. 15. Меню закладок TotalCommander

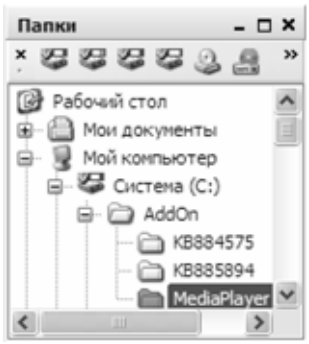


Рис. 16. Дерево папок UltraExplorer

- наличие меню закладок, вызываемого нажатием кнопки "*", расположенной рядом с полем отображения адреса (рис. 15).

TotalCommander позволяет эффективно работать с файловой системой, однако является отдельным приложением, интерфейс которого не может быть эффективно интегрирован в современные операционные системы. Заметим, что в данном приложении не предусмотрена возможность работы в сети Интернет.

UltraExplorer — бесплатный файловый менеджер; стандартное дерево папок дополнено панелью логических и съемных дисков (рис. 16).

Выделим недостатки приложения UltraExplorer, ограничивающие его применение и отсутствующие в модели TreeExplorerA:

- реализованное дерево папок не исключает недостатков, свойственных проводнику Windows Explorer, — корневым элементом является папка *Рабочий стол*, при этом в дереве отображаются расположенные на нем папки, а также все диски и системные папки;
- на панели дисков отсутствуют их текстовые наименования;
- возможность работы в сети Интернет не предусмотрена.

Оценка эффективности модернизированного дерева папок

Описывать процесс работы пользователя с графическим интерфейсом (ГИ) позволяет метод GOMS [2] (Goals, Operators, Methods, Selection Rules — Цели, Операторы, Методы и Правила выбора). Данный метод подразумевает декомпозицию задач пользователя на действия, требующиеся для их решения с помощью ГИ программы. Время выполнения задачи определяется затратами времени на визуальный выбор элементов интерфейса и выполнение воздействий на них, например, мышью. Графы перемещения курсора мыши между элементами дерева папок очень сложны, поэтому эффективность модернизированного и стандартного дерева целесообразно оценивать и сравнивать на основании их информационной нагрузки и затрат времени на визуальный выбор элементов в каждом из них.

В стандартном дереве отображаются наименования всех локальных и съемных дисков, а также системных папок и папок, расположенных на *Рабочем столе*. В результате дерево перегружено информацией, для представления которой применяется полоса прокрутки. При этом наблюдается следующее противоречие — при работе с одним диском (рис. 17, *а*) дерево папок практически не используется, а при работе с несколькими (двумя и более) — оно перегружено информацией (рис. 18, *а*). В модели TreeExplorerA наименования дисков и системных папок вынесены в выпадающий список, поэтому модернизированное дерево не перегружено информацией (рис. 17, *б*, рис. 18, *б*) и более эффективно при работе как с одним, так и с несколькими дисками.

Рассмотрим подход к оценке информационной нагрузки на стандартное и модернизированное дерево папок. Пусть к узлам нулевого уровня

Рис. 17. Информационная нагрузка при работе с одним диском: *а* — стандартное дерево папок; *б* — модернизированное дерево папок

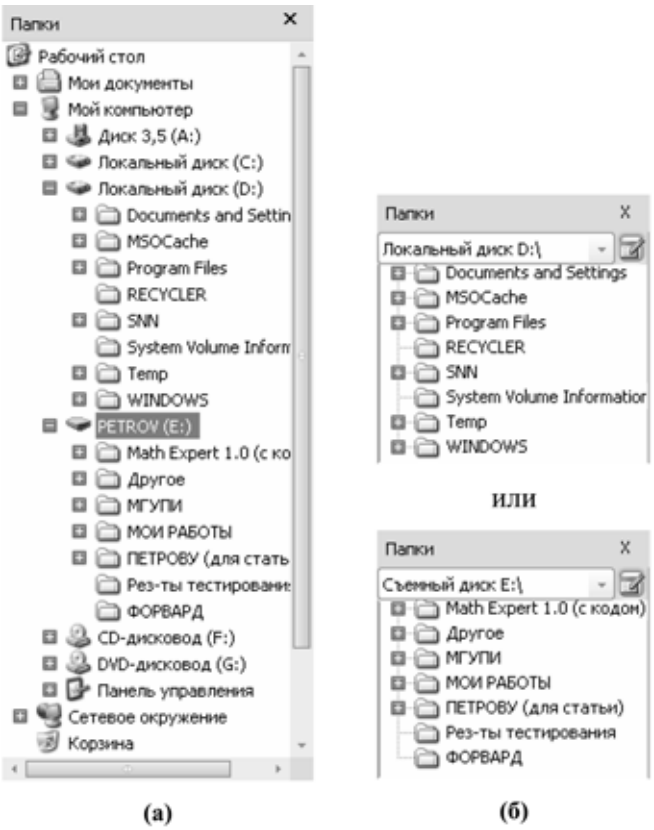


Рис. 18. Информационная нагрузка при работе с двумя дисками:
а — стандартное дерево папок; *б* — модернизированное дерево папок

(корням) относятся диски и папки, постоянно отображенные в дереве. В стандартном дереве имеются минимум пять корней (Рабочий стол, Мои документы, Мой компьютер, Сетевое окружение, Корзина), а в модернизированном — один (выбранный элемент выпадающего списка). Пусть уровни остальных узлов (папок) определяются очередностью их вложенности друг в друга. Предками по отношению к данному узлу будем считать те узлы, которые имеют общий с ним корень и принадлежат к меньшим уровням.

Пусть для стандартного и модернизированного дерева папок известны:

- узлы $v^i_{k,j}$, где $i = \overline{0, n}$, — их уровни, а j и k — порядковые номера данного узла и его предка (для корней k и j равны 0);
- M — множество порядковых номеров узлов;
- $N_i \in M$, $i = \overline{0, n}$, — порядковые номера узлов уровня i, $\bigcup_{i=0}^{n} N_i = M$;
- m — число узлов, отображаемых в дереве без полосы прокрутки;
- $F \geqslant m$ — приемлемая информационная нагрузка на дерево;
- A — число корней.

Для сравниваемых стандартного и модернизированного дерева папок требуется выполнить представленные ниже исследования.

Для каждого узла $v^i_{k,j}$, $k \in N_{i-1}, j \in N_i$, $i = \overline{1, n}$, определим информационную нагрузку D_j — число узлов, отображаемых в дереве папок при доступе к нему. Общего предка с $v^i_{k,j}$ имеют E_j узлов $v^i_{k_c,j_c}$, $k_c \in N_{i-1}, j_c \in N_i$, $k_c = k$, $j_c \neq j$. Обозначим $v^i_{k,j}$ через $v^i_{k_i,j_i}$, его предками являются узлы $v^{i-r}_{k_{i-r},j_{i-r}}$, $k_{i-r} \in N_{i-r-1}, j_{i-r} \in N_{i-r}$, $r = \overline{1, (i-1)}$, для которых $j_{i-r} = k_{i-r+1}$, пусть их B_j. Для каждого из выявленных $v^{i-r}_{k_{i-r},j_{i-r}}$ определим узлы, имеющие с ним общего предка: $v^{i-r}_{k1_{i-r},j1_{i-r}}$, $k1_{i-r} \in$ $\in N_{i-r-1}, j1_{i-r} \in N_{i-r}$, где $k1_{i-r} = k_{i-r}, j1_{i-r} \neq$ $\neq j_{i-r}$, пусть их C_j. В результате $D_j = A + B_j +$ $+ C_j + E_i, j \in N_i$, $i = \overline{1, n}$; определим все $D_j > F$ и обозначим их через D^*_j, $j \in M^*$. Выявим суммарную перегрузку дерева папок информацией $S = \sum_{j \in M^*} (D^*_j - F)$.

Если для стандартного и модернизированного дерева папок были получены значения S_1 и S_2 суммарной перегрузки информацией, то коэффициент $K = S_1/S_2$ является оценкой эффективности представления информации в модернизированном дереве папок по сравнению со стандартным.

Рассмотрим подход к определению затрат времени пользователя на визуальный выбор узлов в стандартном и модернизированном дереве папок. Значения, получаемые для различных исходных данных (числа узлов), могут рассматриваться как оценки эффективности.

В законе Хика [2] утверждается, что при визуальном выборе одного из N равновероятных вариантов (в данном случае — узлов дерева), затраты времени t (в миллисекундах) пропорциональны логарифму по основанию 2 от $(N + 1)$:

$$t = a + b\log_2(N + 1), \qquad (1)$$

где a и b — коэффициенты, зависящие от навыков пользователя (для приближенных вычислений $a = 50$, $b = 150$).

Перед тем как переместить курсор, пользователь должен визуально выбрать требующийся элемент интерфейса, поэтому чем больше вариантов доступно, тем больше времени требуется для выбора. Если варианты не равновероятны, то для определения затрат времени применяется формула

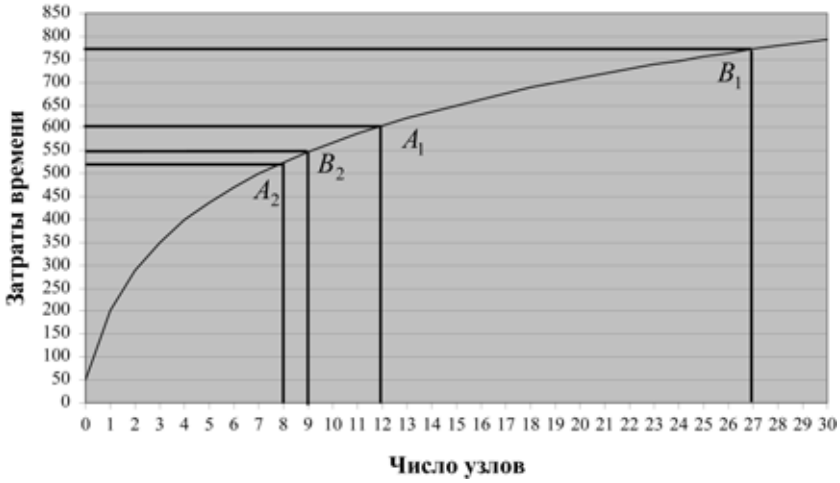


Рис. 19. Зависимость затрат времени пользователя от числа узлов в дереве папок

$$t = a + b \sum_{i \in N} p_i \log_2\left(\frac{1}{p_i} + 1\right),$$

где i — порядковый номер варианта с вероятностью p_i.

Представленные формулы позволяют сделать вывод, что при увеличении числа N отображаемых узлов дерева папок затраты времени пользователя на визуальный выбор одного из них существенно возрастают. Задача выявления значений p_i является достаточно сложной, и разработчик графического интерфейса может не обладать информацией, необходимой для ее решения. Поэтому для иллюстрации зависимости затрат времени пользователя на визуальный выбор узлов в дереве папок от их числа воспользуемся формулой (1), приняв коэффициенты a и b равными соответственно 50 и 150 (рис. 19).

На графике, приведенном на рис. 19, отмечены следующие точки, соответствующие стандартному и модернизированному дереву папок:

- A_1 и A_2 — для случая работы с одним диском (см. рис. 17, *а* и *б*);
- B_1 и B_2 — для случая работы с двумя дисками (см. рис. 18, *а* и *б*).

На основании представленных формул и приведенного на рис. 19 графика можно утверждать, что модернизированное дерево папок более эффективно, так как в нем отображается минимальное число узлов.

Следует подчеркнуть, что получаемые оценки информационной нагрузки и затрат времени на визуальный выбор узлов в модернизированном и стандартном дереве папок зависят от заданных исходных данных. Поэтому целесообразно проводить независимые исследования и формировать исходные данные в соответствии с конкретными особенностями задач пользователя.

Заключение

Модернизированное дерево папок модели TreeExplorerA является конкурентоспособной оригинальной разработкой, соответствующей передовому уровню организации человеко-компьютерного взаимодействия. Модель максимально адаптирована к работе с мышью и клавиатурой, что является отдельным преимуществом и позволяет внедрять ее в различное прикладное ПО, требующее навигации по иерархически структурированным данным. Дерево папок TreeExplorerA расширяет доступное пользователю информационное пространство за счет предоставления возможности навигации в сети Интернет, а также реализует возможность поиска папок и файлов на персональном и удаленных компьютерах. Некоторые разработки, положенные в основу функциональных возможностей модели TreeExplorerA, опубликованы в [3], а подобным разработкам посвящены работы [4, 5].

Представленная модель является прототипом, позволяющим оценить предоставляемые функциональные возможности и реализованные технические решения, в том числе — с точки зрения оценки целесообразности их реализации в различном прикладном программном обеспечении.

Список литературы

1. **Зуев А. С.** Математическое и программное обеспечение средств проектирования и совершенствования интерактивных графических человеко-машинных интерфейсов [Текст]: дис. ... канд. техн. наук / А. С. Зуев. М.: Московский гос. ун-т приборостроения и информатики. 2006. 125 с.

2. **Раскин Д.** Интерфейс: новые направления в проектировании компьютерных систем: Пер. с англ. СПб.: Символ-Плюс, 2005. 272 с.

3. **Зуев А. С.** Подход к разработке и модернизации структур интерфейсов компьютерных программ // Информационные технологии. 2007. № 1. С. 55—62.

4. **Зуев А. С., Петров Ю. И.** Описание модификации строки адреса проводника Windows Explorer // Информационные технологии. 2009. № 3. С. 11—19.

5. **Зуев А. С., Кучеров О. Б.** Модификация принципов работы с дочерними окнами программ, панелями инструментов, главными и контекстными меню // Информационные технологии. 2009. № 4. С. 50—55.

УДК 004.728.5 + 004.451.87 + 004.624

В. А. Пономарев, ст. преподаватель,
О. Ю. Богоявленская, канд. техн. наук, доц.,
Ю. А. Богоявленский, канд. техн. наук, зав. каф.,
Петрозаводский государственный университет,
e-mail: vadim@cs.karelia.ru

Конфигурируемая модульная система мониторинга поведения транспортного протокола на уровне ядра операционной системы

Описывается новая версия системы мониторинга GetTCP. Ключевое отличие системы GetTCP от общеизвестных средств — возможность получения данных, находящихся в адресном пространстве ядра ОС Linux, а также малое влияние на временные характеристики процесса передачи данных. Приведен анализ существующих механизмов получения данных, описана разработанная методика измерения времени выполнения малых участков кода, находящихся в ядре ОС Linux. Описаны архитектура, особенности реализации и временные характеристики системы GetTCP. Приведен пример практического использования системы GetTCP для учета трафика.

Ключевые слова: *TCP, ядро ОС Linux, система мониторинга трафика, GetTCP.*

Введение

Протокол *Transmission Control Protocol* (TCP) [1, 2] играет ключевую роль в управлении трафиком сети Интернет. Алгоритмы протокола определяют качественные и количественные характеристики потоков данных большинства приложений Интернет. Поэтому поведение реализаций протокола TCP оказывает определяющее влияние на устойчивость, надежность и производительность современных сетей, что делает задачу исследования свойств TCP весьма актуальной.

Протокол TCP в части, определяющей управление потоком, является частным случаем протокола передачи данных уровня "точка—точка" с оконным управлением [3]. Для таких протоколов одной из важнейших метрик является величина скользящего окна. Этот параметр определяет верхнюю границу числа единиц данных, которые отправитель уже передал в сеть, но еще не получил подтверждения об их приеме от получателя. Фактически размер окна задает максимальную пропускную способность логического соединения "точка—точка". Значение размера окна для каждого логического соединения TCP хранится в служебной структуре данных, находящейся в адресном пространстве ядра операционной системы (ОС).

К сожалению, общеизвестные средства мониторинга и анализа поведения TCP [4, 5] не позволяют получить доступ к данным, находящимся в адресном пространстве ядра ОС. Таким образом, отсутствует возможность получить точное значение размера окна, которое использовалось при отправке сегмента TCP. Его можно только оценивать, используя различные косвенные характеристики. Аналогичная проблема возникает и для других параметров, определяющих поведение реализации алгоритмов TCP (состояние алгоритма на данный момент времени, различные временные характеристики соединения и т. п.).

Еще одна проблема при реализации средств мониторинга возникает в связи с требованиями ко времени, необходимому для обработки каждого кадра данных в современных высокоскоростных сетях передачи данных.

Пусть пропускная способность физического канала связи известна и постоянна, размер кадра известен. Тогда можно оценить время передачи этого кадра. Чтобы полностью использовать пропускную способность физического канала связи, ОС должна успеть за это время подготовить к отправке следующий кадр. Какая-то часть этого времени расходуется сетевой подсистемой ОС на задачи, не связанные с мониторингом (работа алгоритмов TCP, маршрутизация, формирование кадра и т. п.). Таким образом, если необходимо, чтобы работа системы мониторинга не приводила к снижению пропускной способности, то на время, требуемое системе мониторинга на обработку каждого кадра, накладывается достаточно жесткое временное ограничение: это время не должно превышать разности между временем передачи кадра и временем, затрачиваемым на подготовку кадра к отправке операционной системой (30 360 тактов процессора для нашей тестовой платформы). К сожалению, существующие средства мониторинга часто оказываются не в состоянии уложиться в описанные временные ограничения, что на практике приводит к потерям в учете трафика,

а также к искажениям временных характеристик процессов передачи данных.

Далее описывается система мониторинга GetTCP [6, 7, 8], при разработке которой нами была предпринята попытка решить перечисленные проблемы.

Анализ существующих механизмов получения данных из ядра ОС

Из существующих средств наиболее близки к решению описанных проблем система Systemtap [9] и цели (targets) ULOG и NFLOG системы фильтрации пакетов Netfilter [10]. Система Systemtap в основном предназначена для диагностики проблем и отладки компонентов ядра ОС, применение этой системы для перехвата большого объема данных (например, в системах учета трафика или при накоплении большого количества экспериментальных данных) нецелесообразно. Цели ULOG и NFLOG системы Netfilter лучше подходят для такого рода задач. К сожалению, они не позволяют получить, например, действующую в данный момент времени фазу алгоритма контроля потока TCP. Кроме того, как показывает опыт практического использования цели ULOG, создаваемая вычислительная нагрузка, хотя и является приемлемой при низких скоростях передачи данных, при достижении скоростей порядка сотен Мбит/с становится чрезмерно большой и приводит к потерям данных.

Еще одно направление в данной области — перехват и последующая обработка данных непосредственно в ядре ОС, без передачи перехваченных данных на пользовательский уровень. Примером такого подхода являются "цель" (target) ipt-netflow [11] системы фильтрации пакетов Netfilter, а также псевдоустройство pflow [12] ОС OpenBSD. В указанных примерах агрегация трафика в потоки Netflow осуществляется в ядре ОС, при этом отсутствуют накладные расходы на передачу данных из адресного пространства ядра в пользовательское адресное пространство. Однако, по мнению авторов, такой подход, хотя и является тактически допустимым, ошибочен с точки зрения минимизации объема вычислений, выполняемых в ядре ОС, приводит к уменьшению надежности ядра и потенциальным проблемам (например, потере данных), если код, реализующий обработку данных в ядре, окажется не в состоянии справиться с поступающим трафиком (потоком заявок) в реальном времени.

Таким образом, ни одно из известных авторам существующих средств не позволяет без значительных модификаций решить описанные проблемы, что и послужило причиной создания собственной системы мониторинга.

Очевидно, что для решения первой описанной во введении проблемы (отсутствие у существующих средств мониторинга доступа к внутренним структурам данных ядра ОС) часть системы должна действовать в адресном пространстве ядра. Для эффективной передачи больших объемов данных из адресного пространства ядра в пользовательское адресное пространство была выбрана файловая система RelayFS [13]. Архитектура первой версии системы GetTCP описана в [6].

При разработке второй версии системы GetTCP была предпринята попытка избежать необходимости модификации исходного кода ядра ОС. Для этого была изучена возможность применения технологии Kprobes [14]. Эта технология позволяет устанавливать динамические "контрольные точки" (breakpoints) в исполняемом коде ядра ОС без модификации исходного кода ядра и при достижении такой "контрольной точки" получать управление процессором, а также доступ к аргументам функции, в которой находится достигнутая "контрольная точка". При рассмотрении вопроса о возможности использования технологии Kprobes было исследовано время, затрачиваемое на передачу и возврат управления при достижении "контрольной точки". Для проведения таких исследований оказалось необходимым разработать и реализовать систему измерения времени выполнения участков кода в ядре ОС, решив при этом ряд возникших проблем.

Одна из этих проблем — необходимость измерения малых временных интервалов (время выполнения десятков команд процессора) —сравнительно просто решается аппаратными средствами, в процессорах Intel, начиная с Pentium, доступен 64-битный регистр TSC (time-stamp counter), содержимое которого увеличивается на единицу с каждым тактом процессора. Несмотря на ограниченную применимость (процессор Intel или совместимый, постоянная тактовая частота процессора или процессор, поддерживающий постоянную частоту обновления регистра TSC), для задачи сравнения различных технологий на доступном авторам оборудовании оказалось удобным использовать указанный счетчик тактов процессора.

Другая проблема — влияние на измерения посторонних факторов (например, попадание или непопадание исполняемого кода, время работы которого измеряется, в кэш процессора), в связи с чем время выполнения одной и той же последовательности команд может случайным образом меняться от измерения к измерению. Известным методом в таких случаях является многократное повторение измерений и использование наименьшего полученного значения. Однако при большом числе повторений суммарное время всех измерений может оказаться достаточно большим,

что при работе в режиме ядра ОС приводит к нарушению работы всей вычислительной системы. Чтобы избежать этого, измерительная система выполняет в режиме ядра ОС только одно измерение и затем передает измеренное время с помощью файловой системы debugfs [15] на пользовательский уровень, где работает вторая часть измерительной системы, осуществляющая накопление и обработку экспериментальных данных.

С помощью описанной выше методики были проведены измерения времени, затрачиваемого на получение управления с помощью технологии Kprobes. Аппаратная часть тестовой платформы (ПЭВМ на основе процессора Celeron с тактовой частотой 3 ГГц) не изменилась по сравнению с [6], изменения в основном коснулись программной части, использовалась более новая версия ядра ОС Linux (ядро 2.6.25.11, дистрибутив openSUSE 11.0). Выяснилось, что получение управления занимает минимум 3185 тактов процессора, что составляет около 1 мкс. Учитывая то, что в первой версии системы GetTCP общие накладные расходы на обработку сегмента в 94,02 % случаев не превышали 0,3 мкс [6], было принято решение отказаться от использования KProbes.

Другой механизм — kernel markers [16], он был включен в "официальное" ядро ОС Linux в версии 2.6.24, которая вышла 24 января 2008 г. Kernel markers представляют собой статические "маркеры", заранее включенные в исходный код ядра ОС Linux (в отличие от динамических "контрольных точек" механизма KProbes, которые устанавливаются во время выполнения). Такой подход имеет свои преимущества и области применения. Статические "маркеры" сопровождаются вместе с исходным кодом ядра ОС Linux и, следовательно, всегда согласованы с ним. Кроме того, разработчиками kernel markers были приняты специальные меры по уменьшению накладных расходов при использовании этого механизма.

По описанной выше методике на той же тестовой платформе были проведены измерения времени, затрачиваемого при получении управления с помощью механизма kernel markers. Выяснилось, что для получения управления требуется 138 тактов процессора, т. е. около 0,05 мкс. Учитывая указанные выше обстоятельства, в целях уменьшения накладных расходов при работе системы перехвата было принято решение об использовании для новой версии GetTCP механизма kernel markers.

Еще одна серия измерений была проведена в целях экспериментального подтверждения правильности выбора механизма передачи данных из ядра на пользовательский уровень RelayFS (позднее relay API и debugfs). Сравнивались накладные расходы, возникающие при использовании механизма relay API, с накладными расходами, возникающими при использовании механизма Netlink [17]. Механизм Netlink предоставляет возможность групповой рассылки (multicast) для передачи данных между ядром и пользовательскими процессами и применяется, например, в упомянутых выше целях ULOG и NFLOG системы фильтрации пакетов Netfilter.

Выяснилось, что при использовании механизма Netlink накладные расходы складываются из следующих составляющих:

- выделение памяти для SKB (socket buffer [18]), которое требует минимум 552 такта процессора;
- получение области в выделенном буфере, что требует 241 такт процессора;
- "передача заполненного буфера с помощью групповой рассылки, что требует 563 такта процессора.

При этом в цели ULOG существует ограничение на максимальное число порции данных в буфере, которое составляет всего 50 (константа ULOG_MAX_QLEN). Таким образом, для выделения памяти, заполнения и последующей передачи одного буфера с использованием механизма Netlink требуется $552 + 50 \cdot 241 + 563 = 13165$ тактов процессора, т. е. около 263 тактов на порцию данных.

При использовании механизма relay API накладные расходы складываются из следующих составляющих: получение области в буфере, что требует 92 такта процессора, и переход к следующему буферу, что требует 598 тактов процессора. Размер буфера может быть значительно больше, чем в случае механизма Netlink. Пусть размер буфера relay 128 Кбайт, размер каждой порции данных 232 байт, тогда для заполнения одного буфера потребуется передача 564 порций данных, что потребует $92 \cdot 564 + 598 = 52486$ тактов процессора, т. е. около 93 тактов на порцию данных. Оказалось, что на тестовой платформе на передачу порции данных с использованием интерфейса relay API затрачивается более чем в 2 раза меньше тактов процессора, чем на передачу порции данных с использованием механизма Netlink. Это связано как с меньшими накладными расходами на наиболее частую операцию, так и с возможностью использования в механизме relay API буфера большего размера, чем для механизма Netlink.

Также за счет возможности применения буфера достаточно больших размеров (в тестируемой в данный момент конфигурации используется 4 Мбайт на каждый процессор) смягчаются требования к времени обработки каждого пакета. При этом, даже если процесс на пользовательском уровне не будет успевать обрабатывать поступающие данные, вычислительная система в целом продолжит нормально функционировать (в отли-

чие от упомянутых выше подходов с обработкой данных непосредственно в ядре ОС).

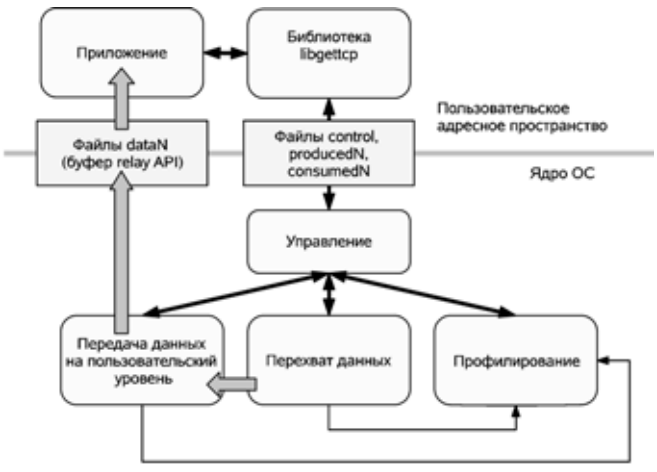
Таким образом, из известных авторам механизмов передачи больших объемов данных из ядра на пользовательский уровень наиболее эффективным действительно является механизм relay API, который и использовался в системе GetTCP, начиная с первых версий.

Для обеспечения интерфейса между частью системы GetTCP, работающей в ядре ОС, и частью, работающей на пользовательском уровне, используется файловая система debugfs.

Архитектура и реализация

Архитектура системы на момент написания статьи представлена на рисунке. Основное отличие от архитектуры первой версии системы GetTCP, описанной в [6] — оформившееся деление части системы, действующей в ядре ОС, на подсистемы (управления, перехвата, передачи данных и профилирования). Подсистемы, в свою очередь, действуют на двух уровнях: управляющем (подсистема управления) и функциональном (подсистемы перехвата, передачи данных, профилирования).

Подсистема управления осуществляет получение команд от пользовательского уровня и их обработку. При загрузке модуля ядра системы GetTCP подсистемой управления в файловой системе debugfs создается управляющий файл control, через который затем происходит передача команд из пользовательского пространства в часть системы GetTCP, действующую в ядре ОС. Команды в новой версии системы GetTCP представляют собой строки, состоящие из обязательной команды и одного или нескольких параметров, разделенных пробельными символами (см. пример ниже). Информация о результате выполнения команды возвращается в виде кода возврата системного вызова write, с помощью которого подается команда. Ошибка при выполнении команды выглядит как ошибка записи в управляющий файл, отсутствие ошибки выглядит как успешная запись в управляющий файл. Такой подход обеспечивает простое взаимодействие с системой GetTCP из командного интерпретатора и из программ на языках среднего и высокого уровня (C, Perl) и, в то же время, наиболее просто и с минимальным размером кода реализуется в ядре ОС. При обработке команд учитывается состояние, в котором находится система GetTCP в момент поступления команды. Например, невозможно включить перехват данных, если не задана конфигурация перехвата или не создан канал передачи данных на пользовательский уровень.



Архитектура и реализация системы GetTCP

Подсистема передачи данных на пользовательский уровень по сравнению с предыдущей версией GetTCP изменилась незначительно. В новой версии для передачи на пользовательский уровень счетчиков числа заполненных и обработанных подбуферов (содержимое файлов consumedN и producedN файловой системы DebugFS) используются строки, а не двоичные значения. Также были внесены незначительные изменения, связанные с тем, что за время, прошедшее с публикации [6], механизм эффективной передачи данных из ядра на пользовательский уровень из файловой системы RelayFS трансформировался в более общий программный интерфейс relay API, а для организации файлового интерфейса стала использоваться файловая система debugfs [15].

Подсистема перехвата данных была модифицирована, в новой версии для получения управления используются kernel markers [16]. В новой версии системы GetTCP появилась возможность динамического управления набором данных, которые необходимо перехватывать (в первой версии этот набор был фиксирован, его изменение требовало модификации исходного кода системы). Также в подсистему перехвата была добавлена реализация цели (target) системы фильтрации пакетов Netfilter [10]. Параметры этой цели аналогичны параметрам цели ULOG, что позволяет легко модифицировать системы учета трафика, основанные на цели ULOG и утилите fprobe-ulog для использования системы GetTCP. При использовании перехвата средствами системы Netfilter недоступно получение таких данных, как, например, номера строки исходного кода, из которой была вызвана передача сегмента TCP. Вместе с тем, становится возможным использование ядра из дистрибутива ОС (при обновлении ядра достаточно перекомпилировать загружаемый модуль ipt_RELAY), а также модификация системы учета трафика сводится к тривиальной замене цели

ULOG на RELAY и запуску модифицированной утилиты fprobe вместо fprobe-ulog.

Подсистема измерения накладных расходов (профилирования) также была реализована с помощью механизма kernel markers ("маркеры" находятся в коде подсистемы получения данных). Основное отличие от предыдущей версии — возможность динамического управления конфигурацией (в новой версии можно указывать ожидаемые диапазоны измерений, число участков гистограммы распределения времени, затрачиваемого на перехват и т. д.).

Значительно изменилась по сравнению с первой версией часть системы, действующая на пользовательском уровне. Ранее эта часть фактически являлась прототипом и состояла из одной программы, сохраняющей перехваченные данные в файл для последующей обработки. В новой версии GetTCP реализована библиотека libgettcp, предоставляющая функции для создания и уничтожения канала передачи данных relay API, управления конфигурацией перехвата, запуска и остановки измерения накладных расходов, запуска и остановки перехвата, получения перехваченных данных. Указанную библиотеку можно применять как в оригинальных программах, нуждающихся в возможностях, которые предоставляет только система GetTCP, так и в уже существующих утилитах перехвата и обработки сетевого трафика. При разработке библиотеки были приняты меры по снижению накладных расходов.

Для проверки возможности применения библиотеки libgettcp в уже существующих программах, а также для тестирования работы библиотеки была использована утилита fprobe-ulog [19], служащая для перехвата сетевого трафика с последующим экспортом в формате NetFlow [20]. Указанная утилита была модифицирована для использования библиотеки libgettcp вместо libipulog. Объем модификаций оказался незначительным (размер полученного diff-файла составил около 6 кбайт). Проведенные эксперименты показали работоспособность полученной модификации утилиты fprobe, а также отсутствие потерь данных при перехвате и формировании потоков NetFlow.

В дополнение к описанным выше изменениям в новой версии в код всех частей системы GetTCP были внесены исправления для работы на 64-битных платформах, новая версия GetTCP успешно компилируется и работает на многоядерных платформах (AMD Athlon X2, Intel Core 2 Duo, Intel Xeon).

Эксперимент по определению реактивности

Для оценки задержки, вносимой новой версией GetTCP в передачу сегмента TCP, был проведен эксперимент, аналогичный описанному в [6]. С помощью утилиты Iperf [21] передавался максимально возможный объем данных за фиксированный интервал времени (120 с) при фиксированном размере сегмента. Для имитации наиболее неблагоприятных условий работы (минимальное время между отправками сегментов данных) был выбран размер сегмента 50 байт.

Полученные результаты представлены в таблице. Для большинства сегментов (94 %) задержка не превышает 1 мкс.

Следует учесть, что измерения проводились для модуля перехвата с возможностью динамического изменения конфигурации перехвата, что, к сожалению, увеличило накладные расходы на перехват. Эксперимент для подсистемы перехвата с фиксированной конфигурацией показал результаты, аналогичные первой версии системы GetTCP.

Заключение

В статье представлена новая версия системы GetTCP, обладающая рядом преимуществ по сравнению с предыдущей версией:

- улучшены архитектура системы и интерфейс между частью системы GetTCP, работающей в ядре ОС, и пользовательским уровнем;
- добавлена возможность динамического изменения различных параметров системы (размера и числа подбуферов relay API, набора перехватываемых данных, ожидаемых диапазонов для подсистемы профилирования);
- реализована библиотека libgettcp, предоставляющая доступ к возможностям системы GetTCP приложениям, работающим на пользовательском уровне;
- добавлена возможность работы как на платформе IA32, так и на платформе AMD64.

При этом сохранились ключевые особенности системы GetTCP (возможность получения данных, недоступных при использовании других существующих механизмов и минимальные накладные расходы). При исследовании существующих инструментов и технологий разработана и опробована методика измерения времени выполнения малых участков кода, находящегося в ядре ОС.

В целях проверки возможностей, а также для расширения области применения системы GetTCR

Задержки, вносимые GetTCP в передачу сегмента TCP

Интервал, мкс	(0; 0,2]	(0,2; 0,3]	(0,3; 0,4]	(0,4; 0,5]	(0,5; 0,6]	(0,6; 0,7]	(0,7; 0,8]	(0,8; 0,9]	(0,9; 1]	>1
Число сегментов, %	2,5	65,2	11,6	8,4	3,4	1,8	0,7	0,3	0,1	<6,1

реализована цель RELAY системы фильтрации пакетов Netfilter, утилита fprobe-ulog модифицирована для использования библиотеки libgettcp и цели RELAY вместо библиотеки libipulog и цели ULOG. Полученный комплекс запущен в тестовую эксплуатацию и выполняет учет трафика на одном из популярных серверов г. Петрозаводска (среднесуточный исходящий трафик около 300 Мбит/с, максимальный около 600 Мбит/с).

Изучаются способы уменьшения накладных расходов на перехват при сохранении возможности динамического управления конфигурацией перехвата. Ведется работа по трансформации системы в свободный программный продукт.

Список литературы

1. **Postel J.** Transmission Control Protocol/STD7, RFC793, September 1981.
2. **Allman M., Paxon V., Stevens W**. TCP Congestion Control // RFC2581. April. 1999.
3. **Бертсекас Д., Галлагер Р.** Сети передачи данных. М.: Мир, 1989. 490 с.
4. **tcpdump,** URL: http://www.tcpdump.org.
5. **tcptrace,** URL: http://masaka.cs.ohiou.edu/software/tcptrace.
6. **Пономарев В. А., Богоявленская О. Ю., Богоявленский Ю. А.** Система мониторинга поведения транспортного протокола на уровне ядра операционной системы // Матер. Второй междунар. научной конф. по проблемам безопасности и противодействия терроризму. М.: Изд. МЦНМО. 2007. С. 349—357.
7. **Пономарев В. А., Ковалев В. Н., Богоявленская О. Ю., Богоявленский Ю. А.** Расширение функций системы мониторинга поведения транспортного протокола на уровне ядра ОС Linux // Тр. XIV Всероссийской научно-метод. конф. "Телематика 2007". СПб.: Изд. СПбГУ ИТМО. 2007. Т. 1. С. 101—102.
8. **Пономарев В. А., Богоявленская О. Ю., Богоявленский Ю. А.** Проверка адекватности и модификация модели случайного потока, генерируемого транспортным протоколом TCP в сети передачи данных // Тр. Междунар. семинара "Распределенные компьютерные и телекоммуникационные сети: теория и приложения (DCCN 2007)". М.: Изд. ИППИ РАН. 2007. Т. 2. С. 59—64.
9. **Systemtap,** URL: http://sourceware.org/systemtap/
10. **Netfilter** — packet filtering framework, URL: http://www.netfilter.org/
11. **Netflow** exporting module for Linux kernel, URL: http://sourceforge.net/projects/ipt-netflow/
12. **pflow** — kernel interface for pflow data export, URL: http://www.openbsd.org/cgi-bin/man.cgi?query = pflow
13. **relay** (formerly relayfs), URL: http://relayfs.sourceforge.net/
14. **An introduction** to Kprobes, URL: http://lwn.net/Articles/132196/
15. **debugfs**, URL: http://lwn.net/Articles/115405/
16. **Kernel** markers, URL: http://lwn.net/Articles/245671/
17. **Kernel Korner** — Why and How to Use Netlink Socket, URL: http://www.linuxjournal.com/article/7356
18. **How** SKBs work, http://vger.kernel.org/~davem/skb.html
19. **NetFlow** probes: fprobe and fprobe-ulog, URL: http://fprobe.sourceforge.net/
20. **Cisco** IOS NetFlow Introduction, URL: http://www.cisco.com/go/netflow
21. **The TCP/UDP** Bandwidth Measurement Tool (iperf), URL: http://dast.nlanr.net/Projects/Iperf

---

УДК 004.7

Н. В. Медведев, канд. техн. наук, зав. каф., **И. И. Троицкий,** канд. техн. наук, доц.,
П. М. Квасов, зав. лаб.,
МГТУ им. Н. Э. Баумана, e-mail: medvedev@bmstu.ru

Аналитическая модель телекоммуникационной сети нового поколения с приоритетным обслуживанием информационных сообщений

В настоящее время разнообразие способов реализации телекоммуникационных и вычислительных сетей определяет актуальность задач как выбора их архитектуры и технологии построения, так и оценки их эффективности как меры соответствия предъявляемым требованиям.

Для объективной оценки сервисов, предоставляемых сетью, пользователи и поставщики услуг используют специализированные характеристики, позволяющие оценить тот или иной аспект качества сети. Проведен анализ сетевых характеристик и особенностей архитектур построения сетей, использующих различные технологии и существующие методы моделирования архитектур.

Ключевые слова: *абонентская станция, узел сети, локальная вычислительная сеть, телекоммуникационная сеть, метод доступа, протокол межсетевого взаимодействия, многопротокольная коммутация по меткам, качество обслуживания, инжиниринг трафика, виртуальные частные сети*

Обозначения и сокращения:

АС — абонентская станция, узел сети;
ЛВС — локальная вычислительная сеть;
ТКС — телекоммуникационная сеть;
ALOXA — метод случайного доступа с разрешением конфликтных ситуаций;
ATM (*asynchronous transfer mode*) — технология асинхронного режима переноса/ передачи данных;
CSMA (*carrier sensitive multiple access*) — множественный доступ с контролем носителя;
CSMA CD (*carrier sensitive multiple access with collision detection*) — множественный доступ с контролем носителя и обнаружением конфликтов;

FRR (*Fast ReRoute*) — механизм быстрой перемаршрутизации;
IP (*Internet Protocol*) — протокол межсетевого взаимодействия, основа транспортных средств стека протоколов TCP/IP;
MPLS (*Multiprotocol Label Switching*) — многопротокольная коммутация по меткам;
QoS (*quality of service*) — качество обслуживания;
TE (*traffic engineering*) — инжиниринг трафика;
VPN (*virtual private network*) — виртуальные частные сети.

Введение

Среди технологий реализации ТКС в настоящее время лидирует MPLS. Изначально задумывавшаяся как средство для упрощения сопряжения сетей IP и ATM, а также для снижения нагрузки на маршрутизаторы, технология MPLS достигла высокой распространенности, благодаря реализованным на ее основе приложениям, таким как инжиниринг трафика TE, виртуальные частные сети (VPN), Fast ReRoute (FRR), обеспечение качества обслуживания QoS. Более того, именно реализация QoS, возможности MPLS VPN и TE вывели ее на лидирующие позиции [1].

Задача исследования такой сети с приоритетным обслуживанием сообщений заключается в оптимизации обобщенного показателя эффективности по числу приоритетов сообщений при условиях безотказной работы сети и отсутствия дополнительных шин для передачи сообщений (предполагается, что оптимальный метод доступа определен).

Исследование сети с приоритетным обслуживанием сообщений

Содержательная постановка частной задачи сводится к следующему. Известны:

- множество $\{T\}$ технических параметров сети (длина такта — h);
- множество $\{S\}$ структурных параметров сети (N — число АС в сети ВН; топология G — "шина"; r^* — оптимальный метод доступа);
- множество $\{H\}$ нагрузочных параметров сети (q_i — вероятность прихода сообщений в выходной буферный накопитель I-го приоритета за время такта h, $i = \overline{1, m}$; L — длина сообщения, распределенная по гипергеометрическому закону с математическим ожиданием $M[L] = \frac{1}{b}$, функция распределения F, относящиеся к оптимальному методу доступа r^*);
- множество $\{C\}$ стоимостных параметров сети ($C_{\text{АС}}$ — стоимость АС; $C_{\text{КС}}$ — стоимость канала; $C_{\text{дос } r^*}$ — стоимость аппаратных и программных средств, необходимых для реализации оптимального метода доступа; $C_{\text{дос}j}$ — стоимость аппаратных и программных средств, необходимых для реализации системы приоритетов сообщений, состоящей из j приоритетов, $1 \leqslant j \leqslant m$).

Необходимо найти оптимальное число приоритетов сообщений i^*, $1 < i^* \leqslant m$, при котором обобщенный показатель эффективности $\Phi = \frac{T_{\text{дос}\,i}}{T_{\text{зад}\,i}} \cdot \frac{C}{C_{\text{зад}}}$ принимал бы минимальное значение при следующих ограничениях:

- среднее время доставки сообщений i-го приоритета $T_{\text{дос}\,i}$ не превышает заданного значения $T_{\text{зад}\,i}$, $i = \overline{1, m}$;
- стоимость разработки сети C не превышает заданного значения $C_{\text{зад}}$;
- число приоритетов сообщений не более m;
- емкость выходных буферных накопителей каждой АС $V^- = 1$;
- емкость входных буферных накопителей каждой АС $V^+ = \infty$.

Математическая постановка рассматриваемой задачи имеет следующий вид: заданы $\{T\}$, $\{S\}$, $\{H\}$, $\{C\}$, необходимо найти систему приоритетов j^*, обеспечивающую

$$\min_j \Phi = T^1_{\text{дос}}\, C = \Phi(j^*) \qquad (1)$$

при значениях $T_{\text{дос}i} \leqslant T_{\text{зад}i}$, $C \leqslant C_{\text{зад}}$, $1 < i \leqslant m$, $V^- = 1$, $V^+ = \infty$, $i = \overline{1, m_j}$, где m_j — число приоритетов j-й системы, $r \leqslant m_j \leqslant m$.

Нахождение оптимального решения задачи (1) является очень сложным, поэтому с помощью графоматричного метода [2] определим только ее допустимое решение.

В рамках решения рассматриваемой задачи (1) проводится также оптимизация среднего времени доставки сообщений i-го приоритета $T_{\text{дос}i}$ по параметрам N, b, q и параметрам функций распределения, которые относятся к оптимальному методу доступа r^*.

Будем считать, что классов приоритетов m_j ($m_j > 1$), причем мощность i-го класса приоритетов равна N_i, где $\sum_{i=1}^{m_j} N_i = N$.

Предположим, что каждая АС может передавать сообщения только одного класса приоритетов [3].

Приоритетное обслуживание сообщений состоит в том, что сначала передаются сообщения, имеющие первый (высший) приоритет, затем передаются сообщения второго (низшего) приоритета и т. д. до m_j-го приоритета.

В данной задаче рассматривается относительный приоритет сообщений: если сеть осуществляет передачу сообщения, имеющего низший при-

оритет, а в это время в АС приходят сообщения, имеющие высший приоритет по сравнению с приоритетом сообщения, которое передается, то сеть заканчивает передачу сообщения низшего приоритета, а затем переходит к передаче сообщений наиболее высокого приоритета.

Под состоянием сети в данном случае будем понимать вектор $(K_1, K_2, ..., K_{m_j}, S)$, где K_1 — число занятых АС, имеющих для передачи сообщения первого приоритета; K_i — число занятых АС, имеющих для передачи сообщения i-го приоритета; K_{m_j} — число занятых АС, имеющих для передачи сообщения m_j-го приоритета; S — номер состояния канала в данном такте.

Состояния канала:

$S = 0$ — канал свободен;

$S = I$ — состояние успешной передачи сообщения, имеющего i-й приоритет, $i = \overline{1, m_j}$;

$S = m_j + 1$ — состояние конфликта (для случайных методов доступа) или передача маркера (для детерминированного доступа);

$S = m_j + 2$ — состояние искаженной передачи информации по каналу (это состояние только для случайных методов доступа);

$K = m_j + 3$ — состояние прерывания успешной передачи информации по каналу (это состояние только для случайных методов доступа).

Тогда основные показатели эффективности функционирования сети определяются следующим образом.

1. Среднее время доставки сообщений l-го приоритета, $l = \overline{1, m_j}$

$$T_{\text{дос}}(l) = \frac{1}{b}\left(\frac{1 - P_0^{(l)}}{E_l}\right) \text{ (в единицах такта } h), \quad (2)$$

где $E_l = \sum_{i_1=0}^{N_1} \sum_{i_l=0}^{N_l} ... \sum_{i_{m_j}=0}^{N_{m_j}} \pi(i_1, ..., i_l, ..., i_{m_j} l)$, $l = \overline{1, m_j}$;

$$P_0^{(l)} = \sum_{\substack{S=0 \\ S \neq l}}^{m_j+3} \sum_{i_1=0}^{N_1} ... \sum_{i_{l-1}=0}^{N_{l-1}} \sum_{i_{l+1}=0}^{N_{l+1}} ... \sum_{i_{m_j}=0}^{N_{m_j}} \pi(i_1, i_2, ..., i_{l-1}, 0, i_{l+1}, ..., i_{m_j}, S),$$

$\pi(i_1, ..., i_{m_j}, l)$ — финальная вероятность нахождения сети в состоянии $(i_1, ..., i_{m_j}, l)$.

2. Стоимость разработки сети

$$C = NC_{\text{АС}} + C_{\text{КС}} + C_{\text{дос.}r^*} + C_{\text{сис.пр.}m_j} \quad (3)$$

где $C_{\text{АС}}$ — стоимость АС; $C_{\text{КС}}$ — стоимость КС; $C_{\text{дос.}r^*}$ — стоимость аппаратных и программных средств, необходимых для реализации оптимального метода доступа r^*; $C_{\text{сис.пр.}m_j}$ — стоимость аппаратных и программных средств, необходимых для реализации системы приоритетов сообщений, состоящей из m_j приоритетов, $1 \leqslant m_j \leqslant m$.

Необходимо отметить, что формула (2) несправедлива для случайных методов доступа, которые позволяют прерывать успешную передачу данных [4].

Можно показать, что среднее время доставки сообщений l-го приоритета для случайных методов доступа ALOXA и n-CSMA определяется следующим образом:

$$T_{\text{дос}}^{(l)} = \frac{1 - P_0^{(l)}}{b\left(E_l - \eta\left(\frac{\alpha_l - \alpha_l^{L+1}}{1 - \alpha_l}\right)\right)}, \quad (4)$$

где $\alpha_l = q^{-N_l - 1}$, $\eta = \sum_{i_1=0}^{N_1} \sum_{i_2=0}^{N_2} ... \sum_{i_m=0}^{N_{m_j}} \pi(i_1, ..., i_{m_j} m_j + 3)/m_j$ (на каждом наборе хотя бы один индекс должен быть $\geqslant 2$).

Поскольку оптимальным методом доступа r^* может быть любой из возможных методов доступа, то для решения задачи исследования сети с приоритетным обслуживанием сообщений необходимо рассмотреть все допустимые методы доступа.

Случайные методы доступа с приоритетным обслуживанием

При исследовании в сети с приоритетным обслуживанием сообщений случайных методов доступа при одновременной передаче сообщений, имеющих одинаковый приоритет, возникает конфликт. Для разрешения конфликта используется процедура разрешения конфликта.

Для рассматриваемых случайных методов доступа: ALOXA, синхронная ALOXA, n-CSMA и p-CSMA-CD алгоритм определения финальных вероятностей является однотипным:

- определение вероятностей перехода из одного состояния сети в другие;
- решение системы линейных уравнений, приведенных в отчете за первый этап, в целях определения финальных вероятностей состояний сети.

На основе финальных вероятностей определяется математическая зависимость для вычисления значений среднего времени доставки сообщений для случайных методов доступа при наличии системы приоритетов сообщений.

Детерминированный метод доступа с приоритетным обслуживанием сообщений

Исследуем маркерный метод доступа. Дисциплина обслуживания классов приоритетов сообщений следующая: сначала маркер передается по

АС, имеющим для передачи сообщения первого (высшего) приоритета, в случае отсутствия сообщений с первым приоритетом, маркер передается по АС, имеющим сообщения для передачи второго приоритета и т. д. до m_j-го приоритета[5].

Поскольку вероятности прихода сообщений одного приоритета одинаковы, то будем считать, что внутри каждого класса приоритетов сообщений маркер передается равновероятно от одной АС к другой с вероятностью $\frac{1}{N_l}$, где $l = \overline{1, m_j}$.

Для маркерного метода доступа алгоритм определения финальных вероятностей полностью совпадает с алгоритмом определения финальных вероятностей для случайных методов доступа [6].

На основе финальных вероятностей определяется математическая зависимость для вычисления значений среднего времени доставки сообщений для маркерного метода доступа при наличии системы приоритетов сообщений.

Выбор системы приоритетов

Для нахождения допустимого решения рассматриваемой задачи для оптимального метода доступа r^* необходимо выполнить следующую последовательность действий:

1. Выбрать начальные значения числа приоритетов M_0 и N_i^0, где $i = \overline{1, m_0}$ и определить финальные вероятности состояний сети.

2. По формулам (2)—(4) вычислить основные показатели эффективности и сравнить их с заданными значениями.

3. Если все ограничения на основные показатели эффективности выполняются, то допустимым решением частной задачи являются начальные значения числа приоритетов M_0 и N_i^0, где $i = \overline{1, m_0}$. В противном случае анализируются полученные результаты и выбираются новые значения числа приоритетов M_1 и N_i^1, где $i = \overline{1, m_1}$, и т. д.

4. В случае отсутствия допустимого решения рассматриваемой задачи целесообразно перейти к решению задачи по выбору оптимального числа дополнительных шин для передачи, необходимых для передачи данных.

Сравнительный анализ методов доступа

Сравнительный анализ полученных результатов показывает, что методы доступа по минимальному значению среднего времени доставки сообщений высшего приоритета (без учета ограничений на средние времена доставки сообщений других приоритетов и на стоимость разработки сети) ранжируются следующим образом [7]:

- маркерный метод доступа;
- p-CSMA-CD (при условии оптимизации по параметрам p, t);
- n-CSMA (при условии оптимизации по параметру t);
- синхронная ALOXA (при условии оптимизации по параметру t);
- ALOXA (при условии оптимизации по параметру t).

В случае учета ограничений на средние времена доставки сообщений низших приоритетов и на стоимость разработки сети сделать однозначный вывод о преимуществе того или иного метода доступа не представляется возможным [8].

Выводы

1. Разработаны аналитические модели зависимости среднего времени доставки сообщений от значений показателей безотказности, надежности при различных условиях функционирования сети и варьировании техническими, структурными, нагрузочными параметрами с учетом стоимостных ограничений.

2. Разработаны рекуррентные алгоритмы определения финальных вероятностей, с помощью которых определяется среднее время доставки сообщений с учетом показателей надежности сети.

3. Анализ полученных результатов показывает, что при увеличении числа абонентов, имеющих для передачи сообщения высшего приоритета, и числа приоритетов сообщений резко увеличивается среднее время доставки сообщений низших приоритетов.

4. Сравнительный анализ показывает, что для нагруженного режима функционирования сети оптимальным является маркерный метод доступа. Для ненагруженного режима все рассматриваемые методы доступа имеют приблизительно одинаковые значения среднего времени доставки сообщений. Это связано с тем, что вероятность конфликта сравнительно мала.

Список литературы

1. **Олвейн В.** Структура и реализация современной технологии MPLS. М.: Вильямс, 2004. 474 с.
2. **Спотак М.** и др. Компьютерные сети и сетевые технологии. Киев: ТИД ДС, 2002. 650 с.
3. **Романов А. И.** Основы теории телекоммуникационных сетей. Киев: Киевский политехнический ин-т, 2002.
4. **Башлы П. Н.** Современные сетевые технологии. М.: Горячая линия —Телеком, 2006. 334 с.
5. **Олифер В. Г., Олифер Н. А.** Компьютерные сети. С.-Пб.: Питер, 2007. 958 с.
6. **Бертсекас Д., Галлагер Р.** Сети передачи данных. М.: Мир, 1989. 544 с.
7. **Шиндер Д. Л.** Основы компьютерных сетей. М.: Вильямс, 2003. 651 с.
8. **Дансмор Б., Скандьер Т.** Справочник по телекоммуникационным технологиям. М.: Вильямс, 2004. 628 с.
9. **Резников Б. А.** Анализ и оптимизация сложных систем. М.: Военное издательство Минобороны РФ, 2007.

УДК 004.75

В. А. Васенин, д-р физ.-мат. наук, проф.,
МГУ им. М. В. Ломоносова,
А. В. Инюхин, науч. сотр.,
НИИмеханики МГУ им. М. В. Ломоносова,
e-mail: shurick@sectorb.msk.ru,
М. В. Шевелев, аспирант,
МГУ им. М. В. Ломоносова

Фрагмент территориально-распределенной информационно-вычислительной среды на основе методологии Grid

Представлены результаты исследований, направленных на создание средств поддержки приложений для проведения вычислений на территориально-распределенных Grid-комплексах. Рассматриваются вопросы, связанные с организацией работы подобных комплексов и обсуждаются механизмы их использования при решении задач различного типа.

Ключевые слова: *распределенные вычислительные системы, Grid-системы, экспериментальный полигон, прототип программного средства, математическая модель.*

В последние годы широкое распространение получает подход, основанный на использовании распределенных вычислений, в целях консолидации ресурсов для выполнения крупномасштабных вычислительно емких приложений. Одной из наиболее популярных на этом направлении в настоящее время является методология Grid (англ. — решетка [1, 2]) с характерными для нее инфраструктурными, архитектурными и технологическими решениями. Под инфраструктурой Grid понимаются программно-аппаратная среда и организационно-техническое обеспечение, которые объединяют ресурсы отдельных вычислительных установок, находящихся в различных административных доменах телекоммуникационной сети. Такая среда позволяет в режиме дистанционного управления эффективно использовать ресурсы этих установок — процессорные мощности, оперативную и постоянную память, программы и данные.

Grid-архитектура принадлежит к классу архитектур распределенного компьютинга с коллективной формой обслуживания пользователей. В качестве его ближайшего аналога можно указать широко используемые на практике кластерные вычислительные системы. Однако Grid нельзя рассматривать как "кластер кластеров". Его инфраструктура гораздо богаче. Как следствие, в случае использования Grid возникает ряд административно-технических и технологических проблем, которые не имеют аналогов и решений в традиционных кластерных системах. Наиболее важными отличительными свойствами Grid-инфраструктур являются следующие.

Глобальная распределенность. Входящие в состав Grid-систем вычислительные установки, как правило, территориально и даже географически удалены друг от друга. По этой причине коммуникационные "издержки" могут быть достаточно велики, и это обстоятельство в числе других предъявляет повышенные требования к отказоустойчивости и безопасности подобных систем.

Автономный характер использования ресурсов. Ресурсная база Grid-инфраструктур формируется объединением средств и систем, независимых друг от друга вычислительных центров или отдельных компьютеров. Ресурсы имеют разных владельцев, которые, обслуживая и администрируя свою часть ресурсов, имеют право проводить собственную политику организации доступа к ним пользователей.

Разделение ресурсов. Ресурсы Grid-инфраструктуры используются коллективно. Этот факт означает, что должны решаться вопросы гибкого, скоординированного их распределения между пользователями, каждый из которых решает свою задачу на общей ресурсной базе.

Динамичность среды. Состав участников и состав ресурсов может меняться с течением времени. Причем для Grid-систем это свойство является скорее правилом, чем исключением.

Таким образом, программное обеспечение, используемое для поддержки функционирования Grid-систем, должно иметь средства обеспечения безопасности, надежности ресурсов, мониторинга состояния системы на предмет работоспособности, наличия аномальных ситуаций, вызванных деструктивными воздействиями, а также запуска заданий и последующего управления ими.

Несмотря на повышенную сложность в организации Grid общая тенденция в развитии данной методологии состоит в обеспечении качественно нового уровня интеграции распределенных информационно-вычислительных ресурсов. Систему, созданную на основе Grid-подхода, в перспективе можно рассматривать как некую единую операционную среду обработки запросов (заданий), работа с которой должна быть не сложнее, чем работа с современными многопроцессорными и кластерными системами.

Осознавая актуальность данной методологии, уже в 2003 г. на базе высокопроизводительных вычислительных систем НИИ механики МГУ им. М. В. Ломоносова (Москва), ИПС РАН (Пере-

славль), ОИПИ НАН РБ (Минск) с использованием телекоммуникационных ресурсов национальной сети для науки и образования был создан первый в России вычислительный Grid-полигон. Его основные функциональные свойства были продемонстрированы на приемосдаточных испытаниях продуктов, разрабатываемых в рамках российско-белорусской программы "СКИФ", которые проводились в Минске. В дальнейшем экспериментальный полигон развивался, и на настоящий момент его можно рассматривать как полноценную Grid-систему экспериментального назначения (пилотную зону) с довольно развитой инфраструктурой, число и состав участников которой постоянно меняются.

В качестве базового программного обеспечения на полигоне используется программная система с открытым кодом для построения Grid-инфраструктур и Grid-приложений, именуемая Globus Toolkit [3]. На настоящее время этот инструментальный комплекс является стандартом среди систем подобного назначения. В программном комплексе Globus Toolkit для разработки Grid-систем и приложений реализован богатый набор протоколов и сервисов, в частности: сервисы и протоколы управления ресурсами, информационные сервисы, сервисы и протоколы безопасности, сервисы и протоколы передачи и управления данными.

Программный комплекс Globus Toolkit предоставляет программные интерфейсы промежуточного уровня для использования перечисленных выше сервисов. С его помощью разработчик Grid-приложений получает безопасный прямой доступ к вычислительным и информационным ресурсам, расположенным в большом числе различных административных доменов. Вместе с тем, необходимо отметить, что с использованием сервисов Globus Toolkit даже запуск и остановка выполнения приложений, а тем более их эффективное распределение по ресурсам вычислительного поля остаются делом достаточно сложным даже для опытного программиста-администратора (разработчика) и требует использования дополнительного высокоуровневого программного обеспечения.

С точки зрения процесса управления ходом выполнения вычислительного приложения ключевое место среди компонентов Grid-систем занимает тот, который реализует функции управления заданиями и, как правило, именуется планировщиком. В перспективе его реализация должна требовать минимального участия от конечного пользователя. Пользователь должен лишь запустить свое Grid-приложение, используя соответствующую команду доступного интерфейса, и через некоторое время получить результаты. Как и где будет исполняться приложение, должно быть определено планировщиком при минимальной дополнительной информации в его адрес со стороны пользователя. Следует отметить, что работа планировщика является определяющим фактором для обеспечения эффективности любых распределенных вычислений.

Полностью текст, отражающий состояние работ по развитию территориально-распределенного Grid-полигона, опубликован в **Приложении к журналу "Информационные технологии" № 7 за 2009 г.** В нем представлены результаты проведенных авторами исследований и опытно-конструкторских работ в направлении развития Grid-полигона и изложены идеи, на основе которых он эволюционирует. В Приложении представлена и обсуждается модель организации управляющего программного обеспечения Grid-полигона, а также ее программная реализация для управления ходом выполнения ресурсоемких вычислительных приложений. Рассматриваются несколько часто встречающихся на практике классов задач и предлагаются сценарии выполнения некоторых из них на Grid-полигоне.

Список литературы

1. **Foster I.** What is the Grid? A three point checklist // GRIDtoday. 2002. July. Vol. 1. N 6.

2. **The physiology** of the Grid: An open grid services architecture for distributed systems integration: Tech. rep. / I. Foster, C. Kesselman, J. M. Nick, S. Tuecke. 2002.

3. **Foster I.** Globus toolkit version 4: Software for service-oriented systems // IFIP International Conference on Network and Parallel Computing. Springer-Verlag. 2005. Vol. 3779 of Lecture Notes in Computer Science. P. 2—13.

ДИСКУССИОННЫЙ КЛУБ

УДК 001

А. С. Нариньяни, канд. физ.-мат. наук,
РосНИИ Искусственного Интеллекта,
e-mail: narin@aha.ru

Между эволюцией и сверхвысокими технологиями: новый человек ближайшего будущего

Через 10—15—20 лет, т. е. в самом непосредственном будущем, сегодняшний цивилизованный человек превратится в eHOMO — новый вид, сохраняющий биологическую принадлежность к Homo Sapience, но качественно отличающийся от него за счет симбиоза с новой окружающей средой, порождаемой стремительно развивающимися сверхвысокими технологиями. В работе делается попытка представить, каким будет этот eHOMO и формирующаяся вместе с ним цивилизация ближайшего будущего.

Ключевые слова: *эволюция, сверхвысокие технологии, новый человек ближайшего будущего, Ноосфера*

Первая версия этого текста была представлена на конференции Диалоге в 2004 году. За прошедшие три с лишним года содержание несколько раз перерабатывалось, а поскольку тема неисчерпаема, автор надеется, что обсуждение с участием аудитории журнала поможет ее углубить и расширить.

Введение

Ноосфера вступила в новую фазу эволюции. Мы стремительно врастаем во все более высокие технологии (*hi-tech*, технологии прорыва) и они врастают в нас. Сегодня в авангарде *hi-tech* информационно-коммуникативные технологии (ИКТ): персоналка, интернет и мобильник — это лишь первые проявления того лавинообразного процесса, который, если не накроет, то радикально изменит нашу цивилизацию в ближайшие 10—20 лет.

Однако параллельно с ИКТ развиваются и другие направления прорыва, из которых ключевыми в данный момент становятся нанотехнологии и биотехнологии с доминантой приставки гено-. И то и другое направление через 5—10 лет могут стать не менее значимыми для ближайшего будущего, чем ИКТ являются сегодня. Поэтому заглядывая за ближайшие горизонты, мы должны принимать во внимание как каждую из этих составляющих *hi-tech* в отдельности, так и их тесное взаимодействие в стремительном общем развитии. Конечно, *человек* понятие весьма многомерное: от гения до дауна, от инвалида до олимпийского чемпиона, от негодяя до святого... Однако *Homo Sapience* остается видовым наименованием, хотя второй его части соответствует, мягко говоря, далеко не каждый.

В фокусе этой работы — технологическая перспектива ближайших десятилетий и связанная с ней метаморфоза усредненного *human being*, которого сегодня еще можно называть просто *Homo*. Биологически за этот период он заметно измениться не может, а вот симбиоз с новыми технологиями способен преобразовать его радикально, превращая сегодняшнего *Homo* в некоторый новый вид, которому мы присвоим очень условно этикетку *eHOMO*.

Очевидно, что *eHOMO* не возникнет спонтанно и ниоткуда: мы уже превращаемся в него, шаг за шагом приближая тот рубеж, когда новый вид станет очевидной реальностью. В любом процессе есть пионеры, так что можно сказать, что *eHOMO* уже среди нас, да и сами мы, пусть немного, но им являемся. Трансформация только начинается, она затрагивает каждого в разной степени, но мы уже не те *Homo*, которыми были 20 лет назад, хотя и еще не такие, какими станем через 10—20 лет. Мы у порога новой эры — очередного захватывающего этапа в развитии нашей глобальной цивилизации, касающегося любого человека ближайшего будущего, его личности, судьбы, тела, и — рискнем добавить — даже души. Процесс пошел, как любил говорить один незабываемый лауреат Нобелевской премии мира.

1. Общий взгляд на предмет

Образ eHOMO нам придется синтезировать по отдельным уже угадываемым деталям, которые можно разделить на два плана:

- общий, отражающий влияние *hi-tech*, и в первую очередь ИКТ, на нашу цивилизацию (Ноосферу) в целом.
- индивидуальный, — черты отдельного *eHOMO*, то сочетание "два в одном", которое отражает симбиоз *Homo* и все то, что нам придется аккумулировать в приставке "e".

Очевидно, что оба эти плана будут меняться достаточно кардинально и, конечно, взаимозависимо. При этом трудно сказать, что именно — Homo или цивилизация —будет более динамичным.

Очевидно, что Ноосфера, т. е. человечество как целое, есть нечто гораздо более масштабное и инертное. Даже под действием самых радикальных факторов Ноосфера до сих пор перестраивалась намного медленнее, чем можно было бы ожидать. Этот консерватизм, с одной стороны, позволял ей выживать в периоды социальных и технологических революций, а с другой стороны, приводил к тому, что отдельные группы людей, а иногда и целые страны, настолько откланялись от общего среднего, что казались принадлежащими совершенно иной цивилизации.

До сих пор устойчивость цивилизации была связана с разнородностью и разобщенностью человечества, которые обеспечивали замедление любой динамики "в мировом масштабе". Однако глобализация, Интернет и рывок сверхвысоких технологий начинают расшатывать эту разнородность и инертность, в результате чего Ноосфера в целом может обойти среднего *Homo* по скорости качественных изменений примерно так же, как "наиболее продвинутые" страны уходили "в отрыв" от основной массы населения планеты.

Но не менее вероятно обратное: в ближайшей перспективе темп формирования *eHOMO* через молодежь будет опережать эволюцию основных структурных составляющих общества: политики, экономики, социальных процессов.

2. Основные составляющие и горизонты

Пытаясь представить нашего ближайшего потомка и его цивилизацию, мы рассмотрим далее некоторые из основных составляющих их развития, которым будут посвящены следующие темы:

- Естественный отбор в мире *hi-tech*
- Родео верхом на сверхвысоких технологиях
- Роботы внутри нас
- Наш Милый е-Друг — симбиоз с потомком мобильника
- Среда е-обитания и глобализация общения
- Е-возможности формирования личности
- Конец "всего святого" и смерть бюрократии
- Е-свобода: eHOMO и Большой Брат
- eHOMO крупным планом

К сожалению, объем этого текста не позволит рассмотреть множество других интересных аспектов будущего, таких как интеллектуальный дом, транспорт, политика, война и некоторые другие, каждая из которых заслуживает не только отдельной статьи, но и специальной монографии. Зато в наш обзор попадет не слишком далекая перспектива ухода в историю нескольких базовых компонентов цивилизации, прежде всего рынка и бюрократии, а завершится он рассмотрением двух нетривиальных тем, которые существенно расширяют рамки нашего рассмотрения:

- От alter ego к бессмертию
- Солипсизм в конце туннеля.

Поэтому уже в начале нашего путешествия за горизонт стоит оговориться, что этих горизонтов получится не меньше трех. Ближайший, отстоящий от нас на 5—10 лет, виден сравнительно ясно. До следующего, который можно соотнести примерно с 2030 годом, прогноз еще достать может, но из-за неизбежных в прогнозе непредсказуемых факторов многие детали этого горизонта теряют определенность. И, наконец, в третьем горизонте, относящемся ближе к середине века, можно попытаться разглядеть лишь отдельные черты, только зарождающиеся сегодня, но кажущиеся такими же фантастическими, какими представляется эскиз взрослого организма при рассмотрении эмбриона на ранней стадии развития.

Таким образом, наше суммарное изображение *eHOMO* будет напоминать совмещение портретов оригинала в юности и лет двадцать спустя плюс попытки с помощью ретуши заглянуть еще на пару десятилетий дальше. В прогностике это естественно, поскольку привязать все элементы будущего к одной временной шкале просто невозможно, — по крайней мере, наш эксперимент за задачу не берется.

3. Естественный отбор в мире *hi-tech*

Интеллектуализация ИКТ — это ближайший этап, необходимость которого подталкивается гиперактивностью трех тесно взаимосвязанных процессов:

- растущим потоком всякой е-техники: множась и эволюционируя, элементы этого потока отчаянно борются за свое право на жизнь, т. е. место на рынке, изо всех сил пытаясь доказать свою полезность;
- всеобщей информатизацией, начинающей проникать не только на каждое рабочее место, но и в карман, на кухню, и даже внутрь организма;
- возрастанием сложности ПО, которая выходит за рамки возможностей текущего поколения технологий как по трудоемкости производства, так и по доступности для массового применения (пример — версии ОС Windows, которые непрерывно латаются на протяжении всего своего жизненного цикла и совершенно непостижимы для рядового конечного пользователя).

Любому устройству от микро до макро приходится бороться за выживание в этом жестоком естественном отборе, который уже исчерпал возможности периода вегетативного развития и требует не только удешевления и миниатюризации, но и все более высокого уровня интеллекта. ИКТ вступают в эпоху, когда требуется понимание пользователя "на лету", часто даже лучше, чем он понимает себя сам.

Эра, когда в очередь на каждую ЭВМ стояли яйцеголовые, готовые за счастье работать на ней постигать китайскую грамоту примитивных технологий программирования, прошла навсегда. Казавшийся когда-то передовым лозунг "программирование — вторая грамотность" выглядят первоапрельской шуткой по отношению к сотням миллионов наивных пользователей: школьников, чиновников, солдат, домашних хозяек. Тут и с первой-то грамотностью не все просто, так что насчет второй и речи быть не может.

Двадцать — тридцать лет назад классики информатики еще могли путать, где Магомет, а где гора в запутанных отношениях человек — компьютер. Но начало нового века все расставило по своим местам. Обычный человек — завтрашний *eHOMO* — будет пользоваться всеми этими предлагаемыми ему е-чудесами ровно настолько, насколько новые продукты смогут быть ему понятными и полезными, превращаясь в привычные компоненты его "электронной среды обитания".

Те же этапы, только в ускоренном темпе, будут проходиться в развитии параллельных направлений, — нано- и биотехнологиями. Существенная разница тут только в том, что ИКТ преодолевают их как бы самостоятельно, т. е. "в одиночку", в то время как другие два направления двигаются в форваторе ИКТ и имеют возможность использовать все уже достигнутые результаты лидеров *hi-tech* для умножения своих успехов.

4. Родео верхом на сверхвысоких технологиях

Сегодня у будущих *eHOMO* в одной руке ноутбук, а в другой — мобильник. Первый становится все более интеллектуальным и компактным, второй — бойко вбирает в себя все больше функций (фото, видео, плеер, диктофон, видеофон, телевизор, органайзер, и т. д.), двигаясь дальше к полной интеграции с остальной информационно-коммуникационной средой через всевозможные быстро размножающиеся сети.

Уже понятно, что функция телефона в мобильнике далеко не основная, а скорее повод к рождению чего-то качественно нового. Происходящая ИКТ революция использовала его в качестве того топора, из которого она варит щи нашего близкого будущего. Еще немного, и мобильник вырастет до статуса нашего е-Помощника и даже е-Друга, превратившись в не таком далеком будущем в наше *Alter ego*, нашу е-Тень.

При этом префикс "е" у eHOMO и других используемых в этом тексте е-сочетаний быстро теряет исходный смысл: вся сегодняшняя суперсовременная электроника — всего лишь макет недалекого будущего, в котором она станет таким же этапом прошлого, как ламповая ЭВМ, паровоз, галера или каменный топор. Очень скоро она уступит место технологиям еще более супер с приставками био-, гено-, нано-и многими другими, нам пока неизвестными.

В качестве иллюстрации темпа развития компьютеров нам постоянно напоминают принцип Мура, согласно которому скорость процессора удваивается каждые два года. За этой расхожей формулой ухитряются не видеть реального прогресса вычислительной техники, который можно оценить, только учитывая, кроме скорости процессора, и все остальные составляющие компьютера — от объема памяти всех уровней до степени развития интерфейсов и периферии. Между тем, отношение качества к цене ПК растет *на порядок в год*, в то время как для развития традиционных высоких технологий (авиация, фармацевтика, оптика и т. п.) считается сенсационным скачок годовых показателей качества в десяток процентов, т. е. в *сто* раз меньше, чем в ИКТ.

Сегодня средний ноутбук превосходит отечественного лидера начала шестидесятых БЭСМ-6 примерно в 1000 раз по каждому из трех основных параметров: производительности, объему ОЗУ и размеру внешней памяти. Итого миллиард, не считая уровня программных технологий, качества интерфейса и много чего еще. Таким образом, рядовой ноутбук на несколько порядков эффективней всей вычислительной техники мира сорок (а может быть и только тридцать) лет назад.

На сколько же выросли суммарные ИКТ ресурсы за последние десятилетия, если на тот уровень, который был тогда у всего человечества, сегодня вышли возможности отдельного *Homo*? И каковы будут его ресурсы через 10—20 лет с учетом того, что сам он при этом будет окружен кучей процессоров собственной среды, полностью включенной в мировую сеть этого недалекого будущего? Как суммарный объем интеллекта десятков тысяч ноу-хау, вложенных в эту информационную мощь Ноосферы, расширит, углубит, умножит ее влияние на ближайшую перспективу нашей цивилизации?

5. От бифуркации к бифуркации

Хорошо известно, что траектория любого развития состоит из чередования участков поступательного процесса и *точек бифуркации*, т. е. крутых поворотов, отличающихся, как правило, своей внезапностью. Эта внезапность и эффект скачка тесно связаны. Общество настолько привыкает к текущему привычному эволюционному ходу событий, что как бы не видит или не может вообразить все более очевидное приближение предстоящего драматического перелома, который часто называют революцией.

В политике бифуркации нередко происходят за дни или даже часы, в технологиях — когда-то за века и еще недавно за десятилетия. Сегодня темп развития вырос настолько, что на радикальные смены технологий тратится всего несколько лет, а иногда и того меньше.

При сохранении такого темпа определение общей картины в перспективе 15—20 лет становится задачей сложной, а, скорее всего, невыполнимой. Если же учесть, что само развитие ИКТ за свою короткую историю чуть больше полувека уже прошло множество бифуркаций разного масштаба, то экстраполяция "умножением на порядок в год" может помочь только в грубой оценке отдельных составляющих, но никак не в надежном предвидении полной панорамы будущего.

Проведем мысленный эксперимент, — вернемся в 1990 год. Прогресс базовых программных технологий за истекшие 19 лет очевиден, но здесь явно обошлось без бифуркаций, особенно в сравнении с остальными составляющими ИКТ. Изменения в "железе" более контрастны, однако и тут революции были локальными, — в конце концов, в сегодняшнем ПК вполне узнается персоналка того времени. Но на этом эволюционная часть картины кончается: Интернет и мобильная телефония тогда были еще в зародыше и их сегодняшнюю роль предвидеть было вряд ли возможно. Что же касается гено- и наноперспективы на 20 лет вперед, в 1990 году обсуждение ее вообще не имело бы какого-либо смысла.

Этот опыт показывает, что объект текущего прогноза всегда напоминает комплексное число: его действительная часть экстраполируется на перспективу вполне осмысленно, в то время как "мнимая" часть абсолютно непредсказуема, поскольку в ней скрыты будущие бифуркации, недоступные нашему воображению сегодня.

Одна из них — наноперспектива — вышла за последнюю пару лет на передний план и породила лавину информации о фантастических успехах в самых разных направлениях от электроники до медицины. Наверное, уже можно оценить некоторые составляющие этой перспективы. Например, какие бы потрясения не рисовались на горизонте в области ИКТ, они касаются прежде всего скачка в миниатюризации элементной базы, ее стоимости и качества, но никак не вызовут принципиальных изменений. Но в нанотехнологии два направления стоит выделить уже сейчас. Одно из них — это радикальное изменение в такой вяло эволюционирующей "инженерной" области как материаловедение. Казалось бы, что чрезвычайного может произойти здесь? Однако, это более чем обманчиво: именно в материаловедении решительно преобразуется все — от быта (одежда и строительные материалы) до самой сложной и масштабной техники (корабли, авиация, космос). Качественно новые свойства материалов могут не только изменить на порядки параметры корпуса автомобиля или космического корабля, но и привести к появлению принципиально новых решений в любых типах двигателей. Между большинством сегодняшних материалов и их нанопотомками разница может быть более контрастной, чем между современным процессором и его ламповым предком. Второе направление — это вышеупоминавшиеся наноустройства, которые могут сыграть непредсказуемую роль в любой области, но уже сейчас очевидно, что им предстоит ключевая функция на "внутреннем фронте" *eHOMO* в не слишком далекой перспективе.

Добавим к этой картине только начинающую формироваться, но быстро уходящую в прорыв генную инженерию. Уже угадываются ее будущие возможности в таких областях как: "оптимизация" нашего организма и продление срока жизни, формирование искусственных микроорганизмов и их интеграция с нанотехнологиями в создании гетерогенных микроустройств, использующих достижения как гено- так и нанотехнологий. Совсем фантастической вырисовывается возможность непредсказуемой трансформации биологического мира, — в качестве иллюстрации можно привести сообщение из текущей прессы: *генетически модифицированные тополя со встроенными генами кролика во много десятков раз быстрее очищают почву от промышленных загрязнителей...*

6. Роботы внутри нас

Начнем с тех высоких технологий, которые прямо нацелены на вторжение в наш организм. Этот комплекс средств можно разделить на три уровня:

- долговременные или пожизненные составляющие;
- элементы длительного (недели, месяцы) действия;
- элементы оперативного (часы, дни) действия.

На обсуждаемом периоде в 20—40 лет не предполагается прямого тотального включения личности в какую-то суперсеть в стиле "Матрицы". Однако вот-вот начнется все более широкое внедрение в организм датчиков и эффекторов, уже получивших наименования микро- и нанороботов. Вся эта невидимая туча искусственных помощников наводнит наш организм для того, чтобы заниматься нашим здоровьем: биомониторингом, физиологической регуляцией, медицинским контролем и лечением.

Очевидно, что датчики, эффекторы и создаваемые на их основе все более масштабные армии микроустройств — это базис, необходимый для контроля и оптимизации действия не только самих этих комплексов, но и более традиционных средств воздействия на организм. В ближайшем будущем тело человека будет становиться все более прозрачным. И управляемым. Чуть ли не до каждой клетки. В этом контексте "микроустройство" совсем необязательно означает привычные чипы, какими бы они миниатюрными не становились. Самой этой текущей полупроводниковой силиконовой парадигме жить осталось не так уж долго. Ее вытеснят — по крайней мере на территории нашего организма — другие, гораздо более тонкие элементные базы, интегрирующие возможности механических, электронных, синтетических биологических и генетических технологий, все более приближающиеся к молекулярному и атомному уровню, прежде всего те, которые видны уже сегодня: нано-, био-, генотехнологии.

Эти технологии будут работать для реализации и внутренних, и внешних функций. Первые будут направлены на совершенствования "системы тела", что предполагает как коррекцию подаренного нам природой организма, так и механизмов его "реинжиниринга". Например, запуском соответствующих внутренних программ можно будет оптимизировать свою печень, а то и перестроить пропорции тела, получив фигуру культуриста без многочасовых занятий бодибилдингом. Развитие же внешних функций будет направлено на прямое взаимодействие организма *eHOMO* со средой: бытовой, производственной, с другими *eHOMO*, например, на управление техническими компонентами среды сигналами мозга. Другая технология, направленная к мозгу извне, будет обеспечивать расширение возможностей личности и воздействие на центральную нервную систему в лечебных целях, а также, возможно, использоваться для коррекции психики *eHOMO*: ограничение агрессии, блокирование боли, мобилизация, и т. п. И здесь, как всегда, границы между воздействием в интересах личности, в интересах общества или в интересах "третьей стороны" очень условны и предоставляют все более широкий простор для самых непредсказуемых направлений в психологии и социологии.

7. Милый е-Друг — наш симбиоз с потомком мобильника

Подросток в наушниках плеера, занятый игрой на мобильнике, временами заглядывающий на трансляцию матча на его экране и обменивающийся СМСками — это эмбрион будущего *eHOMO*, "погруженность" которого в электронный мир растет прямо на глазах. Симбиоз начинает походить на наркотическую зависимость, — *мобиломанию*, новое появившееся несколько лет назад психологическое расстройство. Подверженные сотовой зависимости люди, забыв или потеряв свой телефон, испытывают стресс, постоянную раздражительность и склонны проявлять немотивированную агрессию.

Уже сейчас для многих потерю личного компьютера можно сравнить с серьезной инвалидностью, — урон может быть невосстановим: контакты, архивы за многие годы, тексты, фотографии, музыка... — утрата огромной части личности. Это при том, что наш будущий е-Друг представляет собой пока в основном довольно пассивную, не очень толково организованную внешнюю память. Если отвлечься от прикладных пакетов для специалистов, то сегодня он умеет только самое простое: поискать текст по ключевым словам, проверить его орфографию и помочь оформить для печати; выполнит функцию калькулятора, примет почту, защищая ее от вирусов, а в последнее время немного и от спама. Может работать средней стенографисткой (для английского) и посредственным переводчиком. В отдельных, довольно узких, областях может выступить в качестве примитивного эксперта, а для массового пользователя, он, в основном, — привычное средство для почты, связи с Интернет и игр. Конечно, ему пока еще очень далеко до толкового секретаря, помощника, консультанта. Далеко сегодня, но вполне достижимо к 2015—2020 году в результате форсированного роста интеллектуализации ИКТ. В результате этого симбиоза к середине века *eHOMO* с рождения и до старости будет находиться в своего рода личном информационном коконе, который будет его воспитателем, защитой, помощником, дополнением, усилителем, продолжением, — его *Alter Ego*, участвующим в его развитии и развивающимся вместе с ним.

Нарастающий поток информации извне, умноженный на тьму помощников *внутри организма* и просто невероятный прогресс виртуальной ре-

альности, начинают размывать и без того нечеткую грань между объективным восприятием и субъективным, синтезируемым искусственно.

Подменять реальность — обманывать, пугать, возбуждать — умеют и животные. Возможности манипулирования своим и чужим сознанием человек каждый раз начинал осваивать одновременно с каждым появлением у него очередного нового канала информации. К естественным средствам человек добавлял синтетические: рисунок, речь, письменный текст, музыку, театр. За последний век к этому присоединились фото, кино, телевидение. Сегодня в этот спектр включился быстрорастущий объем электронной информации самого разного типа. Стремительное развитие мобильника превращает этот поток в наводнение, к которому вот-вот добавятся тактильные ощущения, запахи, эмоции.

Совсем не так уж далеко до того времени, когда на любом расстоянии станет возможным самый прямой контакт с близким человеком. Или с его имитацией. Или с виртуальной правнучкой резиновой куклы, продаваемой сегодня в лавке "интим".

8. Среда е-обитания и глобализация общения

Погружение *Homo* в быстрорастущую и усложняющуюся среду обитания, т. е. переход в вид *eHOMO*, можно сравнить с приездом деревенского подростка в мегаполис: многократное расширение окружающего мира; умножение его сложности, возможностей и соблазнов.

Интернет наступает. Он уже вылез из компьютера, становится частью любого мобильника, каждого дома и квартиры. В конце концов, наверняка попадет к нам под кожу. Разрастаясь и усложняясь, всемирная сеть становится единой нервной системой Ноосферы, а вход в нее — окном и дверью в *brave new world* новых возможностей е-цивилизации. Эти возможности включают чуть ли не все: справки, услуги и консультации; знакомства; электронная торговля; банковские расчеты; реклама; все виды обучения и образования; фантастический спектр развлечений; все виды хобби; возможности реализации в науке, искусстве, бизнесе, а может быть, и в личной жизни, ограниченные только персональными способностями и удачей.

Одним из основных свойств среды обитания *eHOMO* становится глобализация личного контакта.

Если я любитель игры в го, мне требуется среда общения. Для "фана" необходимо сообщество единомышленников, интегрирующее энергетику всех для стимулирования соответствующей активности внутренней жизни каждого. Еще недавно такое сообщество формировалось локально, ограничиваясь территорией, размеры которой определялись традиционными средствами коммуникации. Интернет уже разрушает этот региональный принцип организации: глобальные сети делают естественный контакт в реальном времени таким же привычным элементом образа жизни, каким во второй половине двадцатого века стал телевизор. Общество любителей игры в го получает возможности глобализации, обеспечивающие равные возможности участия для своих членов в любой точке земного шара. Если у вас нет партнера во дворе, вы можете найти его на другом континенте. В чем-то это даже комфортнее, поскольку такой контакт работает только до тех пор, пока устраивает обе стороны. Это означает, что государственные границы все больше теряют смысл: растущие мировые сети обеспечивают русскоязычной диаспоре гораздо более широкие каналы контакта, чем сегодня по месту жительства или в офисе. Живя в африканской деревне, можно будет получать образование в Оксфорде и консультации в лучшем центре тибетской медицины. А найти друзей и партнеров по хобби, или даже спутника жизни, на другом конце Земли можно и сейчас.

Нервная система Ноосферы становится все более сложной. Ее роль "во благо" кажется очевидной. Уровень и эффективность ее организации продолжает расти и специализироваться, многократно расширяя спектр возможностей и творческой реализации для каждого человека — ее виртуальной клетки.

Но и эффект "во вред" также неизбежен, хотя пока невозможно предвидеть и оценить его в полной мере. Информационная избыточность сродни свободе слова: в ее "белом шуме" можно найти любую точку зрения, но чтобы быть услышанным, необходим мощный усилитель. Да и любая сложность — сестра уязвимости: мы видим результаты действия все более "хитрых" вирусов и талантливых хакеров, способных взламывать самые "непреодолимые" системы защиты. Реклама, пиар и манипулирование сознанием приспосабливают для себя Интернет и прочие составляющие е-цивилизации, используя те же приемы, которые уже отработаны в полностью подмятых ими СМИ.

Каждый шаг технологического прогресса врастает в ткань Ноосферы, становясь ее необходимым компонентом, укрепляя ее в одном плане и делая все более уязвимой в другом. Так, серьезное отключение электричества или канализации в городе может вызвать катастрофу, а в небольшой деревне — только вполне поправимые трудности.

9. Все е-возможности

Панорама следующего этапа развития образования, искусства и науки в рамках е-цивилизации для нашего обзора настолько необъятна, что здесь ее придется пройти только пунктиром по диагонали.

Сегодня электронное образование только формируется и набирает силу. От простого текстового пособия к обучающим системам и, наконец, к дистанционному образованию. Учиться всему, чему угодно, *eHOMO* сможет, не выходя из дома. Привлечение ведущих специалистов, умноженное на потенциал мультимедиа, позволяет создать учебные программы, намного превосходящие по качеству уровень среднего ВУЗа. Развитие этого сектора будет менять и саму традиционную систему обучения в школах, колледжах, институтах, даже на курсах, кроме тех составляющих образования, где виртуальная реальность никак не может полностью заменить прямой опыт. Но и сама необходимость прямого опыта может сокращаться до минимума. Достаточно вспомнить, например, о тренажерах для пилотов, — системы такого типа, еще 10—15 лет назад представлявшие собой дорогое "спецоборудование", в ближайшие годы могут стать доступными для каждого начинающего водителя, который сможет получать первые недели опыта, разъезжая по виртуальной Москве в условиях, мало отличимых от реальных, кроме возможности попасть в настоящее ДТП.

ТВ в XX веке "отгрызло" у театра и кино значительную часть территории свободного культурного времени человечества. Оно продолжило начатый комиксами процесс замены текста картинками, окончательно сместив книгу с принадлежавшего ей несколько сот лет места "властителя дум". В последнее десятилетие компьютер продолжил это наступление, все шире раздвигая рамки масс-культуры и, как волшебный крысолов, все дальше завлекая детей и взрослых в гипнотизирующий инфантильный мир игровой виртуальной реальности.

Виртуальная реальность не только завлекает, — она победно наступает: большой и сложный фильм уже снимается в пустом и темном зале, где вся жизнь от пейзажей до исторических интерьеров и деталей синтезируется на компьютере натуральней и дешевле, чем в натуре. Еще немного, и живые актеры будут участвовать в роли свадебных генералов, в качестве звезд, приглашаемых "для перчика", — такой же экзотики, которой вчера были виртуальные динозавры. Цифровая реальность приходит и на эстраду, и в театр. Писатели-фантасты нередко оживляли быт далекого будущего виртуальной обстановкой, — сегодня ясно, что оно совсем не так уж далеко.

При этом электронные виды искусства размножаются как новые виды спорта. Возможности человеческого воображения, умноженного на компьютерные технологии, делают перспективы их применения непредсказуемыми. Они захватывают не только зрительное пространство, но и звук, включая музыку, вторгаются в мир осязания и запахов, сферу эмоций. А главное, сам процесс творчества — наша монополия — совсем не так неприступен, как это кажется. По крайней мере, нет никаких сомнений, что у масс-культуры есть все для того, чтобы выступить генеральным заказчиком тотальной компьютеризации самой себя.

Странно, но в то время как искусство тонет в потопе компьютеризации, науке пока не страшны никакие цунами. Вчерашняя башня слоновой кости на наших глазах превращается в свою вавилонскую предшественницу, строившуюся для штурма неба: чем быстрее ИКТ поглощает ее нижние этажи, тем более стремительно растут верхние. Теоретические области науки вообще не требуют теперь физического присутствия сообща в одном месте, поскольку прямые контакты обеспечиваются видеосвязью и телеконференциями. Быстро расширяющиеся возможности сочетания компьютерного моделирования и виртуальной реальности позволяют все больше вытеснять материальный эксперимент машинным примерно так же, как тренажер заменяет реальный самолет или автомобиль.

Распределенные профессиональные команды, объединенные в коллективы сетями прямой связи и использующие доступные через Интернет почти неограниченные вычислительные ресурсы, смогут работать практически в любой области при значительном сокращении капитальных вложений и материальных ресурсов. Реальный эксперимент останется необходимым только на переднем крае науки, где объект исследования еще не настолько изучен, чтобы превратить его в эквивалентную компьютерную модель.

10. Конец "всего святого"

Под "всем святым" в этом контексте я имею в виду привычные СМИ, бизнес, рынок и бюрократию. Такое развитие процесса неизбежно, каким бы невероятным оно не казалось.

Традиционные СМИ доживают последние годы. Телевидение превращается в цифровое, широкоэкранное, карманное, интерактивное и чем дальше, тем все более персональное. Сайт становится не менее важной частью газеты и журнала, чем его привычная бумажная форма. Блог позволяет каждому завести свою собственную публичную трибуну. Время окончательной победы ин-

тернет-изданий над традиционными определяется только темпом роста ширины доступных каналов и размеров мобильных экранов. У привычных телевизоров, если они и выживут, останется только та часть населения, которая, не ощущая потребность выбора, предпочтет существовать в привычном мире искусственного информационного питания.

Еще одно направление наступления новой эры — это е-бизнес. Уже сейчас он максимально активен, напористо проникая в жизнь по всему фронту — от спама до электронных бирж, от хранилищ данных до систем поддержки решений и ситуационных центров: B2B, ERP, CRM, IB и т. д. и т. п. Деловая контора становится насквозь электронной, система управления компанией, как пылесос, втягивает функции белых воротничков. Блондинка за компьютером в приемной скоро превратится в такой же эксклюзивный элемент стиля офиса как музейная картина или антикварный экспонат.

"Исчезают из обращения" и в конце концов уйдут совсем *наличные*: *cash* навсегда утратят материальность и перейдут, как и много другое, в категорию сущностей виртуальных. Электронные счета и новые формы расчета, интегрирующие кредитные карты и мобильники ближайшего будущего, сделают дензнаки не только лишними, но и мешающими тотальной прозрачности финансовых потоков.

В результате всего этого самое святое нынешней цивилизации — рынок, если и выживет, то может трансформироваться в свою противоположность. Компьютеризация бизнеса и, прежде всего банковской и налоговой систем, делает их все более прозрачными, оптимизируемыми и детерминированными. Тут не так далеко до формирования е-экономики — сообщества систем управления, в котором каждый участник рассчитывает наилучший вариант своей и чужой игры на N ходов вперед. В результате этого процесса свободная рыночная экономика может сама "стихийно" шаг за шагом превратиться в стратегически сбалансированное плановое хозяйство. Похоже, что если бы Советская власть прожила еще три десятка лет, то экономические системы СССР и "цивилизованных" стран стали бы практически неотличимы. А поскольку экономика и социальная сфера — это "близнецы-братья", то сходство может стать просто поразительным.

11. Смерть бюрократии

Вряд ли уцелеет и последний краеугольный камень нашей цивилизации — бюрократия: она исчезнет как класс и также станет виртуальной, превратившись в е-бюрократию. В такое счастье невозможно поверить, однако этот процесс неизбежен. Он естественным образом и почти незаметно уже идет снизу: бумаги и формы переходят в е-текст. Электронный документооборот еще туповат, но с каждым поколением ИКТ (т. е. каждые несколько лет) будет становиться все более интеллектуальным, окончательно вытесняя бумажные канцелярию и архив. Пока компьютерные программы заменяют в основном бухгалтерию, складские и прочие технические службы. Но этот фронт быстро расширяется. Принять оптимальное оперативное решение при достаточно полной информации несложно. Стратегическое, конечно, намного сложнее, но ведь компьютер уже сегодня выигрывает в шахматы у чемпиона мира.

Завершение эпохи чиновника неизбежно, поскольку одновременно с ускоряющимся усложнением Ноосферы, объем и неэффективность бюрократии растет в кубе. Она становится проблемой номер один не только для устойчивого развития, но и для самого выживания человечества, отнимая приоритет глобальной угрозы у экологии и локальных войн.

При концентрации усилий интеллектуальные ИКТ вполне могут покончить с нижней частью пирамиды бюрократии в ближайшие 5—10 лет. С завершением создания общих баз данных о населении и внедрением технологий автоматической идентификации личности ненужными станут справки. За этим последует "творческий" уровень оформления отчетов и распоряжений. Далее — уровень простейших решений.

Пирамида будет все более проседать, теряя в этажности и объеме, становясь все более прозрачной и детерминированной. Исчезнут различные окошки, включая единое окно, приемные, залы для совещаний, а потом и кабинеты лиц, уполномоченных решать и согласовывать. В конце концов, только на самой макушке бывшего кафкианского замка бюрократии как дань традициям останутся в качестве флюгеров символические функции VIP.

12. E-свобода

До этой главы наш герой *eHOMO* рассматривался в основном как развитие *Homo Sapience* в контексте возможностей создаваемой им самим новой технологической цивилизации, которая помогает ему превратиться в суперличность (этот вариант цивилизации можно обозначить индексом *A*).

Однако надо признать, это направление путешествия в будущее не единственное и, возможно, далеко не самое вероятное. За бурную историю последнего века развитие событий неоднократно

принимало форму, когда бифуркация не просто корректирует направление прогресса, а совершает нечто вроде мертвой петли. В результате доминантой становится злокачественное сочетание политтехнологий, лингвистического программирования и масс-культуры, которое подготавливает оптимальный уровень разжижения мозгов, обеспечивающий питательную среду для цивилизации типа Б — той антиутопии, которая проглядывала местами и в наших предыдущих разделах.

Как это ни печально, но для очередного поворота к цивилизации Б есть все предпосылки.

Чем выше технический уровень Ноосферы, тем более зависимым от ее многомерной сложности становится *eHOMO*. Контроль его индивидуальности уже не ограничивается отпечатками пальцев. Биометрия каждый год расширяет набор граней идентификации: голос, зрачок, ДНК.... Того и гляди, вшитый чип будет не только сообщать твой личный набор параметров и хромосом, но и место на карте с точностью до метра.

"Плохому" *eHOMO* будет все сложнее уходить от присмотра Большого Брата. А кому считаться плохим, насколько и почему, будет становиться вопросом все более тонким и деликатным, объективность которого будут определять неизвестные нам весы добра и зла надвигающейся е-цивилизации.

Появляется ощущение, что напор ИКТ формирует новый мир, подмывая фундамент (конечно, виртуальный) — принцип либерализма: свободу личности и противопоставление ее "системе". Этому процессу как бы помогает провидение, послав ему в качестве катализатора неуклонно ширящуюся всеобщую борьбу с мировым терроризмом, очень напоминающую ту самую борьбу за мир, в результате которой может не остаться камня на камне от того, за что боролись.

На самом же деле, панорама еще более драматична: очень похоже, что сегодняшний цивилизованный мир хоронит самого себя, активно совмещая функции повивальной бабки и крестного отца нашего е-будущего.

В результате, в наступающем мире *eHOMO* прозрачным станет он сам, его финансы и имущество. А также, как это обсуждалось выше, и вся опирающаяся на рынок экономика, в которой прозрачность и уровень автоматизации решений превратит ее в нечто вроде шахматного турнира компьютерных программ. С этого рынка уйдут те, у кого программы будут похуже, а оставшиеся равные попадут в равновесие патового баланса сил, в котором решения будет принимать кто-то за сценой.

К этой впечатляющей картине осталось добавить широкие возможности манипулирования как отдельными гражданами, так и самыми различными группами. Такие возможности и сегодня близки к совершенству, но трансформация *Homo* в *eHOMO* добавляет к этому искусству дополнительный набор средств.

Сама прозрачность члена общества практически под любым углом сделает его идеальным объектом морально-психологического воздействия в "правильную сторону". Потенциал использования достижений *hi-tech* в различных частях тела *eHOMO*, включая мозг, неограничен. Легко представить самые широкие возможности объекта этого совершенствования и как личности, и как элемента е-цивилизации. Хотя, конечно, почти все это достижимо и сегодня без каких-либо микро-, нано- и геночудес, но системе еще потребуется пара десятков лет до достижения абсолютного совершенства.

Остается надеяться, что цивилизация *Б* в этом соревновании перспектив не окажется единственной альтернативой, и уж если вариантам *А* и *Б* придется как-то сосуществовать, то в самом крайнем случае пусть это будет не хуже, чем сегодня.

13. *eHOMO* крупным планом

Мы уже обсудили набросок *brave new world eHOMO*. С одной стороны, включение в глобальную е-цивилизацию с ее океаном возможностей образования, коммуникации, личного развития и развлечений, с другой стороны, — радикальная трансформация его организма, и с третьей стороны, — растущая зависимость от среды вплоть до тотального контроля, уровень которого не снился и самым авторитарным системам.

Осталось поближе познакомиться с самим *eHOMO*, чтобы представить, кем станет *Homo* сегодняшний. Какой импульс получат его индивидуальные способности, какие подпорки и усилители предоставят ему сверхвысокие технологии, в каких направлениях сделают они возможным как развитие "традиционных" сторон его личности, так и добавление качественно новых?

Фантастика и футурология уже породили на эту тему море теорий и догадок. Их обсуждение нам придется оставить в стороне, поскольку задача этой статьи ограничена, в основном, экстраполяцией нескольких уже сложившихся линий развития ИКТ. Поэтому мы сократим рассмотрение основных пунктов этого раздела до минимума, достаточного для суммарного отображения вполне реальных сторон портрета нашего героя.

Выше уже высказывалась гипотеза о том, что к середине века *eHOMO* будет с момента рождения находиться в своего рода коконе — информаци-

онной оболочке, выполняющей функции его воспитателя и помощника. Фактически эта оболочка начинает формироваться уже сейчас как индивидуальная информационная поддержка, база данных и знаний. Все более усложняясь и интеллектуализируясь, она превратится в расширение и продолжение оригинала, помогающее ему в развитии и развивающееся в симбиозе с ним, совершенствуя его интеллект, способности, психологию и физические возможности в контексте будущей е-цивилизации.

Интеллект. Для *eHOMO* становится досягаемой вся накопленная человечеством информация, которая уже лет через 15—20 будет оцифрована практически полностью. В его распоряжении будут по существу неограниченные резервы памяти, ему будут доступны мощные глобальные технологии вычислений, поиска, содержательной обработки данных и знаний, оценки, выводов и обобщений. Развитые экспертные системы в любой области будут доступны ему для нахождения оптимальных решений.

Все это перечисляется здесь не для очередного обзора возможностей, предоставляемых *eHOMO* его цивилизацией, а как иллюстрация масштаба средств умножения его личного потенциала. Так же как средний ноутбук сегодня превосходит по ресурсам всю компьютерную технику мира 40 лет назад, так и *eHOMO* с природными данными рядового научного сотрудника может оказаться равным крупному современному исследовательскому институту, а то и нескольким. И дело тут не только и не столько в резервах предоставленного ему "компьютерного разума" (или протеза). Умножаются и *способности* самого *eHOMO*. Он с детства станет объектом анализа и оценки его индивидуальных особенностей и талантов, воздействия специальных технологий их развития, реализации и "раскручивания". Глобальная информационная среда будет служить катализатором активного взаимодействия одаренных *eHOMO*, необходимого для их дальнейшего совершенствования.

Психология eHOMO потенциально может превзойти достижения техники Дзен или йогов в освоении методов и приемов адаптации, концентрации, саморегуляции, мобилизации и всего прочего, не только или не столько опираясь на их методы, сколько на поддержку в этом плане своего е-кокона.

Физиология eHOMO получит свой спектр средств совершенствования организма, ориентированных на стимулирование физических возможностей, наилучшего режима функционирования внутренних органов, мозга, мышц.

Те же технологии станут частью будущей *медицины*, которая будет опираться на все возможности от будущих нано- и гено- *hi-tech* до концентрации наиболее передового опыта в интеллектуальных приборах и системах. Нет сомнений, что детальная модель конкретного организма будет доступна каждому человеку в качестве активного компонента его е-кокона, совмещающего оптимальное авторегулирование системы тела с внешними воздействиями в случае необходимости.

Медицина не такого уж далекого будущего сделает возможным невозможное в восполнении и восстановлении утраченных или ограниченных по отношению к норме способностей, компенсации и протезировании функций глаза, уха, руки, ноги, сердца, реабилитации пострадавших участков мозга, восстановлении провалов памяти или забытых событий прошлого.

Думаю, что тут стоит остановиться, поскольку далеко не все возможности человека достаточно изучены или даже достоверно известны, и именно развитие этих глубинных неисследованных сторон личности может изменить портрет *eHOMO* более радикально, чем все перечисленные выше сверхвысокие технологии.

14. От *alter ego* к бессмертию

Если все разделы выше касались так или иначе экстраполяции текущих процессов, то данный можно с полным правом отнести к пересечению футурологии и научной фантастики. Однако мне этот сюжет представляется не таким уж фантастическим, за исключением, может быть, одной-двух фраз, относящихся к распределению ролей в тандеме "душа — тело". В любом случае, обсуждаемая здесь тема становится все более актуальной, хотя ее "окончательная" реализация выходит за рамки обсуждаемой ближайшей перспективы, причем, возможно, и относительно далеко.

Два фактора строят технологическую дорогу к индивидуальному бессмертию: быстро расширяющийся объем и разнообразие оцифровываемой "персональной" информации и доступность неограниченных по сути резервов памяти для ее накопления.

Еще полтора века назад "следы личности" ограничивались материальными составляющими: письмами, дневниками, портретами, частями гардероба. Затем туда добавились фото- и киноматериалы, аудио- и видеозаписи. Вчера еще "штучные", такие элементы архива сегодня штампуются лавинообразно: электронная фотография и кино, почта и любые текстовые материалы в памяти компьютера. Эти горы информации становятся все более важной частью среды обитания: уже ве-

дутся работы, решающие задачу превращения мобильника в летописца жизни его обладателя, в устройство, записывающее его жизнь, автоматически систематизируя всю информацию, проходящую через его личные каналы: сообщения, снимки, видео- и звуковые клипы. Нетрудно представить, как этот конвейер отпечатков действительности превращается в непрерывный поток, копирующий весь процесс существования, составляющие которого неотличимы от реальности. Видео-, аудиопространство становится трехмерным, к нему добавляются запахи, тактильные ощущения, информация от многочисленных электронных элементов, отражающих состояние организма, биотоков и эмоций...

Если когда-нибудь к этому добавится возможность фиксировать внутренний виртуальный мир, — сны, воспоминания, переживания, воображаемые образы, — то полнота и воспроизводимость этой информации в некотором смысле отделит душу от тела. Причем именно эта е-душа получит в активном *Alter ego* 0,99... "оригинала", число девяток которого будет постоянно расти, а тело, таким образом, превратится в некое специальное устройство для записи и воспроизведения этого мегаархива. Таким образом, личность, в конце концов, может окончательно переселиться в *Alter ego*, а тело станет заменяемым на новое при достаточном сходстве настройки основных параметров.

Добавьте сюда аппарат стволовых клеток и генную инженерию. За следующие несколько десятков лет они могут добиться результатов, которые сегодня невозможно даже вообразить. И данное нам природой сложное устройство тела станет реальным заменять по частям или целиком, подправляя ошибки природы и наследственные недостатки, с полной гарантией бессмертия души на протяжении всего периода цивилизации, способного обеспечить стабильность такого процесса.

Параллельно на будущее е-бессмертие работают уже сегодня быстро разворачивающиеся технологии создания тех упоминавшихся выше микророботов, армия которых будет следить в организме человека за функционированием всего комплекса органов и систем, корректируя их в случае отклонений от заданного оптимума. В результате, человек физически будет постоянно молодым и здоровым, так что ничто не будет мешать ему жить вечно.

15. Солипсизм в конце туннеля

Солипсизм — философское направление, согласно которому единственно существующим является только субъективное Я и содержание его сознания. (Краткая философская энциклопедия. М.: Прогресс, 1994)

Воплощение чистого солипсизма — это крыса с встроенным в мозг источником импульсов" кайфа", на контакт которого она может давить сколько угодно вплоть до собственной гибели. *Homo*, конечно, — не крыса, так что одним контактом тут не обойдешься, но зато, будучи *Sapience*, он всевозможные источники воздействия на различные центры своей психики постоянно изобретает сам. И, кажется, достаточно близко приближается к "окончательному решению" проблемы.

Все известные цивилизации проходили примерно один и тот же цикл развития. Начиная с энергичного материализма, они успешно развивались до своего зенита, проявлявшегося прежде всего в покорении соседних народов. Успех триумфаторов и роль локальных хозяев ойкумены начинал кружить им голову. А завоеванное благополучие вызывало некоторое пресыщение "побежденной" реальностью и потребность в расширении ее границ, за счет той или иной виртуальности. В результате этого эффекта "поздней империи" картина мира начинала терять свой материалистический фундамент, а вслед за этим таяло и адекватное восприятие действительности. Естественным следствием такого "цивилизационного солипсизма" был упадок и неизбежная гибель под натиском очередных народов-варваров, энергичный материализм которых обеспечивал им право на новую главу всемирной истории.

Таким образом, прошлое учит нас, что степень субъективизма или виртуальности текущей стадии культуры является своего рода лакмусовой бумажкой, свидетельствующей о близости ее конца. Пока Ноосфера представляла букет цивилизаций, находящихся в разных фазах их жизненного цикла, каждая такая культурная бифуркация была драмой "местного значения", придававшей всеобщей истории очередной импульс для выхода на новую спираль развития.

Но текущая стадия глобализации явно стремится к унификации попавших под ее обаяние культур, следствием чего становится синхронизация их циклов эволюции. Возникает естественный вопрос, а найдется ли среди них та, которая сумеет сохранить достаточный резерв энергичного материализма, чтобы принять на себя ответственность за будущее Ноосферы, выскальзывающее из рук впадающего в делириум солипсизма "цивилизованного мира". Человечество явно завершает этот же цивилизационный цикл теперь уже на глобальном уровне. Для пещерного человека мир был материален объективно, — непрерывная борьба за выживание на заре человечества вряд ли оставляла много места виртуальности.

К античности соотношение материального и духовного стало настолько симметричным, что философия занялась поиском ответа на вопрос, что из них первично. Однако для рядового *Homo* этот вопрос оставался за рамками его текущего бытия вплоть до XX века, когда появление кино и телевидения сделало виртуальность привычной частью повседневности.

16. *Deus ex machine* и глобальная бифуркация Ноосферы

С наступлением XXI века становится все более ясно, что полная замена реальной составляющей картины мира виртуальной не только возможна, но до нее не так уж и далеко. Более того, на эту тотальную бифуркацию Ноосферы работают пять мощных факторов нашей сегодняшней цивилизации.

1. Глобальная машина СМИ управляет процессом формирования нашей картины мира, а, следовательно, мозгом и даже душой каждого. "Сознание первично": манипулируя сознанием можно как угодно переустраивать реальность, заменяя ее виртуальностью.

2. Масс-культура — "всеобщий китч как образ жизни" — превратилась в доминанту новой эры. Оседлав СМИ, она буквально на глазах совершенствуется в мастерстве употребления интеллектуальных ресурсов человечества для забивания "гвоздей" в центры кайфа *Homo* с целью окончательной ампутации у него обременительной функции *Sapience*. Разжижение мозгов уже приближается к порогу, за которым способность отличать настоящую реальность от виртуальной исчезает полностью.

3. Намертво сросшиеся политика и бизнес — основные заказчики на технологии манипулирования сознанием. Масс-культура при этом решает стратегические задачи общей подготовки "клиента", превращая его в материал, оптимально пластичный для очередных оперативных трансформаций действительности. Меньше чем за век реклама и пропаганда стали неразличимы в качестве составляющих пиара, который взял на себя функции философского камня современности по превращению белое в черное и *vice versa*.

4. Ротор научно-технического прогресса стремительно увеличивает обороты. Растет не только сложность нашей среды обитания, растут также число ее измерений и темп перемен. Превращения происходят не только в сфере промышленных технологий, где революции в различных областях случаются чуть ли не каждый год. Меняется *modus vivendi* человечества: быт, стиль жизни, этика и мораль. "Время перемен" учит новые поколения адаптироваться не столько к самим переменам, сколько к скорости их смены. Патология симбиоза общества потребления и поспешности прогресса все очевиднее: важной становится не качество продукта, а степень его новизны, т. е. не реальность прогресса, а виртуальность его производной.

5. И наконец, тема этой статьи: сверхвысокие технологии в роли троянского коня, на котором виртуальность победно въезжает в наш и без того не слишком крепкий мозг. *Hi-tech* становится все более важной составляющей личности каждого и Ноосферы в целом. Это новое измерение готово принять в себя все, что можно оцифровать, а оцифровать, как выясняется, можно практически все.

Эти пять демонов породы *deus ex machina* дружно работают над приближением всеобщего солипсизма, т. е. такого состояния Ноосферы, когда виртуальность, формируемого сознания *eHOMO*, станет для его рассудка и даже подсознания более близкой и реальной, чем та конкретная материальность, которая останется за рамками его прямого восприятия.

Как в фокусе с зеркалами: если сами зеркала не воспринимаются, то Зазеркалье невозможно отличить от действительности.

Заключение

Очевидно, что каждая из рассмотренных в этой статье тем в отдельности достаточно известна и не предлагается в качестве откровения. Однако, как это часто бывает, сумма или произведение очевидных фактов может послужить основой для новой оценки общей картины и неожиданных выводов. Далеко не все составляющие эволюции сверхвысоких ИКТ, определяющие образ *eHOMO* и его цивилизации, удалось затронуть в этой статье даже бегло. Кроме того, наверняка за рассматриваемый период будущего в некоторых, если не в большинстве, из этих составляющих произойдут свои, локальные бифуркации, — революции, которые в той или иной степени повлияют и на общую картину.

Тем не менее, мне кажется, что в первом приближении проведенная реконструкция достаточно близка к будущему оригиналу. Хотя этот оригинал и несколько двоится, совмещая план ближней перспективы, отстоящей на 10—20 лет, и пару следующих, раза в два-три более далеких. Приближающаяся грань цивилизационной бифуркации "*eHOMO*" слишком близка, чтобы спокойно относить ее к сфере научной или ненаучной фантастики. Она — у порога, ее приближение отчетливо ощущается уже сегодня. Мы сами вносим вклад в ее формирование.

Возможно, кому-то представленная панорама покажется излишне черной, особенно ее послед-

ний раздел. Но у меня нет ни малейшего желания нагнетать ужасы относительно будущего человечества. Их сегодня и так уже достаточно много. Тем более меня не привлекает "творческая удача" писателя Робертсона, предсказавшего за 14 лет до гибели "Титаника" всю драму с точностью до деталей. Предпочитаю оказаться неправым и надеюсь, что судьба Ноосферы будет более счастливой и очередного "конца света" она сможет избежать, как ей это удавалось неоднократно в прошлом. Мое дело выстроить честную экстраполяцию, как я ее вижу. "Кто предупрежден, тот вооружен" — завещали нам римляне.

Похоже, что подобно остальным глобальным супертехнологиям, — атому, космосу, генетике и другим, — ИКТ на очередном этапе своего развития предлагает нам в начале XXI века все тот же сомнительный подарок — Ящик Пандоры, завернутый в Скатерть Самобранку...

Я выражаю искреннюю благодарность тем писателям science fiction, которые наряду с учеными и конструкторами могут с полным правом считаться соавторами как самого eHOMO, так и наступающей e-цивилизации.

А также признательность всем, участвовавшим в обсуждении этого текста. Надеюсь на продолжение обсуждения, которое позволит уточнить перспективу и сообща подумать над тем, как сделать ее более оптимистичной.

Остается добавить, что статья публиковалась еще до того, как стало ощущаться наступление кризиса. Поэтому наша экскурсия за горизонт отражает экстраполяцию мира годичной давности. Однако, чем дальше, тем сильнее складывается подозрение, что происходящая мировая бифуркация может развиваться столь круто, что у Ноосферы через пять лет будут совсем другие доминанты и приоритеты. Это не означает, что eHOMO не состоится, — просто очень может быть, что жить и развиваться ему придется совсем в другом мире.

Информация

Уфа: 29 марта — 2 апреля 2010 г.

ПаВТ'2010 — международная научная конференция, четвертая в серии ежегодных конференций, посвященных развитию и применению параллельных вычислительных технологий в различных областях науки и техники.

Организаторы конференции:

- Российская академия наук
- Суперкомпьютерный консорциум университетов России

Главная цель конференции — предоставить возможность для обсуждения перспектив развития параллельных вычислительных технологий и представления результатов, полученных ведущими научными группами в использовании суперкомпьютерных технологий для решения задач науки и техники.

Тематика конференции покрывает все аспекты применения высокопроизводительных вычислений в науке и технике, включая приложения, аппаратное и программное обеспечение, специализированные языки и пакеты.

В первый день работы конференции будет объявлена 12-я редакция списка Top50 самых мощных компьютеров СНГ.

Индустриальная сессия: Программный комитет придает особое внимание привлечению к работе конференции представителей промышленности. С этой целью в рамках конференции будет организована индустриальная сессия. На сессию принимаются высококачественные презентации по коммерческому аппаратному и программному обеспечению, ориентированному на применение суперкомпьютерных и параллельных вычислительных технологий в различных областях науки и техники.

Место и время конференции: ПаВТ'2010 будет проходить 29 марта — 2 апреля 2010 г. в Уфимском государственном авиационном техническом университете.

Языки конференции: русский, английский.

Информация о конференции на сайте http://www.parallel.ru/

CONTENTS

We suggest method for quantitative impact assessment for information security threats in corporative information network. Assets critical level assessment based on experts answers, AHP, corporative information network model. We suggest algorithm "Experts category assessment" for quantitative assessment critical level for information assets.

Keywords: information security of corporative information network, assets classification, information security threats.

In article the complex estimation of property "security" of information systems is given. On the basis of methodology of Common Criteria the set of mechanisms of protection appropriate to a set of functions of safety is defined. The parameter of security of information system taking into account accommodation of mechanisms of protection in system of information safety is formed. The approach of two-step optimization of system of information safety is considered. On the one hand, set of functions of safety at presence of budget restrictions. On the other hand, accommodation of mechanisms of protection in structure of system of information safety by criterion of a maximum of a parameter of security of information system.

Keywords: information systems, estimation, criterion of a maximum, mechanismus of protection, property "security".

This paper deals with data protection methods against information leakage, which might occur in spite of cryptographic protection, in case attackers use IP-datagram length modulation to encode covert messages. The methods considered suggest using additional or multiplicative randomized jams into the data flow.

Keywords: covert channel, data protection, information leakage, traffic masking, crypto router.

The paper is devoted to survey of ontology application in intellectual technologies and systems. The ontology engineering problems are discussed. The approach to decisions support based on semantic nets of design patterns is suggested.

Keywords: ontology, knowledge base, intellectual technology, decision support.

This paper introduces an overview of existing multidimensional data visualization approaches, describes their highs and lows, states criterions that should fulfil sought visualizer. Concepts of the visualization, the visualizer and the visualization method, which uses the gravitational clustering algorithm, are presented.

in terms of reliable communications network performance and the lack of additional buses for communications. The comparative analysis of access methods with a choice of an optimal method for various operating modes of a network is carried out.
Keywords: telecommunication network, network message, message delivery time, data transmission, performance indicator, optimization, loading, priority service, access method.

This work presents the results of the scientific research work which aim was to create program tools for maintaining of compute-intensive applications on geographically distributed Grid systems. Also questions dealed with the organization of work process in such complexes are concerned and their utilization methods for solving different kinds of tasks are discussed.
Keywords: distributed systems, Grid-systems, pilot test site, mathematical model.

Адрес редакции:
107076, Москва, Стромынский пер., 4
Телефон редакции журнала **(499) 269-5510**
E-mail: it@novtex.ru

Дизайнер *Т.Н. Погорелова*. Технический редактор *О. А. Ефремова*.
Корректор *Г. Д. Назарьева*
Сдано в набор 05.11.2009. Подписано в печать 16.12.2009. Формат 60×88 1/8. Бумага офсетная. Печать офсетная.
Усл. печ. л. 9,8. Уч.-изд. л. 10,66. Заказ 19. Цена договорная.
Журнал зарегистрирован в Министерстве Российской Федерации по делам печати, телерадиовещания и средств массовых коммуникаций.
Свидетельство о регистрации ПИ № 77-15565 от 02 июня 2003 г.
Отпечатано в ООО "Подольская Периодика"
142110, Московская обл., г. Подольск, ул. Кирова, 15