ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

6(142)
2008

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

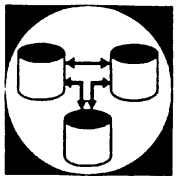
Издается с ноября 1995 г.

УЧРЕДИТЕЛЬ
Издательство "Новые технологии"

СОДЕРЖАНИЕ

Аннотации статей размещены на сайте журнала по адресу http://www.**informika.ru/text/magaz/it/** или **http://novtex.ru/IT.**

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



БАЗЫ ДАННЫХ

УДК 004.658:007.51

А. Д. Брейман, канд. техн. наук, доц.,
Московский государственный университет приборостроения и информатики

Средства автоматизации администрирования персональных баз данных

Описаны ключевые требования, предъявляемые системами персональных баз данных к средствам автоматизации их администрирования. Рассмотрены тенденции развития таких средств в составе коммерческих СУБД. На основе понятия транспарентности взаимодействия пользователя с базой данных предложена концепция развития средств автоматизации администрирования персональных баз данных.

Введение

Проблемы повышения эффективности систем персональных баз данных (ПБД) обусловлены, по существу, одной фундаментальной причиной — быстро растущей сложностью управления современными вычислительными системами, включающими большое число технических и программных компонентов, обладающих принципиальными ограничениями по надежности, объему используемой памяти, в сочетании с ограничениями на объем, пропускную способность и надежность памяти пользователя, а также с учетом других особенностей восприятия и коммуникации.

В свое время технология баз данных (БД) сформировалась как средство централизованного хранения и модификации информации, достоверной и доступной для пользователей и программных приложений. Усложнение БД вызвало появление специальных систем управления базами данных (СУБД), позволяющих пользователю взаимодействовать с БД на основе некоторого унифицированного упрощенного абстрактного представления данных — модели данных.

Необходимость создания ПБД вызвана в первую очередь задачей обеспечения доступности данных, удовлетворяющих разнообразные текущие и долговременные информационные потребности пользователя, обеспечивающих, в свою очередь, удовлетворение его витальных потребностей. Особенности коммуникационного поведения пользователя требуют разработки адекватных моделей данных и методов организации персональных БД и СУБД, обеспечивающих доступность данных.

Всякий раз создание более высокого уровня управления требует формализации модели объекта управления: модели данных при создании БД, модели базы данных при создании СУБД. Эффективное решение задач администрирования БД также невозможно без создания формализованного описания как собственно объекта администрирования (базы данных, ее пользователей, источников информации), так и процессов администрирования. При этом модель должна учитывать наличие ограничений на доступные ресурсы и параметры аппаратного, программного и организационного обеспечения.

Управление такими объектами требует, помимо всего прочего, наличия прогностических функций и функций распознавания предаварийных ситуаций. Реализация именно этих функций и представляет наибольшую проблему при попытках автоматизации администрирования БД. Так, настройка производительности (и ряд других функций) часто уже может осуществляться автоматически, но, например, восстановление данных после крупной аварии осуществляется обычно вручную и/или с помощью специальных инструментов.

При решении некоторых задач администрирования пока невозможно обойтись без прогностических творческих способностей специалистов. Автоматизация таких задач включает мониторинг ситуации и предложение набора стандартных решений, выбор же нужного решения (и, возможно, его модификацию) осуществляет администратор БД. Такой подход к снижению затрат на администрирование, как правило, малоэффективен, но он позволяет снизить требования к администратору БД и упростить его работу. Поэтому именно такой способ автоматизации администрирования наиболее распространен в среде корпоративных информационных систем.

Для индивидуальных информационных систем такой подход в большинстве случаев может стать приемлемым при условии, что соответствующие программные продукты при их использовании не потребуют специальных знаний и больших затрат времени. Для индивидуальных систем представляется целесообразным сочетание следующих способов администрирования:

- решение долговременных задач с помощью внешних специалистов (абонентское обслуживание, Интернет);
- решение оперативных задач с помощью автоматического администрирования;
- использование методов автоматизированного администрирования с интерактивным участием пользователя.

При этом, например, задача проектирования и модернизации ПБД должна быть автоматизированной, интерактивной и максимально упрощенной, а размещение в памяти данных в зависимости от их актуальности, скорее всего, должно осуществляться автоматически.

Таким образом, магистральным направлением совершенствования администрирования БД представляется автоматизация его функций, основанная на комплексном подходе, увязывающем решение частных задач ад-

министрирования в единую систему. Если для корпоративных информационных систем допустим невысокий уровень автоматизации администрирования БД, то для индивидуальных информационных систем уровень автоматизации должен быть максимизирован, чему может способствовать лучшая формализация объекта, сниженный уровень требований к оперативности, надежности, безопасности и т. д.

Таким образом, главным направлением повышения эффективности администрирования ПБД следует считать создание автоматической системы администрирования с элементами автоматизированного управления администрированием отдельных задач. Поэтому ПБД должны обладать свойствами самонастройки и приспособления к изменяющимся условиям без явных указаний пользователя. Такие системы БД называют самоуправляемыми (англ. *self-managing*) и самонастраиваемыми (англ. *self-tuning*) [1].

Самоуправляемые системы баз данных

Современные подходы к построению самоуправляемых (самонастраиваемых, автономных) систем основаны на использовании обратной связи. Наибольшее развитие получила инициатива фирмы IBM по разработке автономных вычислительных систем (англ. *autonomic computing*) [2]. В качестве фундаментальных характеристик автономной системы рассматривают ее способности к самоконфигурированию, самолечению, самооптимизации и самозащите [3].

Под самоконфигурацией (англ. *self-configuring*) понимается адаптация системы к быстрым изменениям среды: динамическое наращивание информационной инфраструктуры новыми функциональными возможностями, программным и аппаратным обеспечением без прерывания нормального функционирования системы.

Самолечение (англ. *self-healing*) — это процесс выявления и диагностики повреждений и отказов, а также действия по их предупреждению. Процесс самолечения подразумевает выявление и отключение отказавшего компонента системы, его починку или замену и обратное подключение без прерывания нормального функционирования системы. Для успешного самолечения в большинстве случаев требуется упреждающее резервирование компонентов.

Под самооптимизацией (англ. *self-optimizing*) понимается автоматическая настройка и балансировка нагрузки для максимально полного использования имеющихся ресурсов.

Самозащита (англ. *self-protecting*) — это автоматический процесс предсказания и выявления атак, их идентификации и принятия мер защиты.

Традиционный подход к построению автономных систем — внедрение программно-аппаратных автономных компонентов. Автономный компонент охвачен обратной связью, на основе данных мониторинга он принимает решения о необходимых действиях, приближающих систему к заданной цели, имеет средства воздействия на управляемые элементы и обучается на прошлом опыте.

Согласно [1], создание самоуправляемых СУБД требует проведения исследований в следующих областях:

- автоматизация планирования мощностей, в том числе моделирование и оценка систем с неполными спецификациями, модификация моделей при развитии аппаратного и программного обеспечения;
- автоматизация установки СУБД с учетом дополнительного требуемого программного обеспечения;
- автоматизация проектирования БД, в том числе:
  - логического проектирования БД, включающего нормализацию схем БД;
  - физического проектирования БД, включающего выбор дополнительных структур данных (индексов и материализованных представлений), способов размещения данных (кластеризация, фрагментация, динамическое обеспечение хранилищами);
- автоматизация настройки производительности, в том числе выявление медленно выполняющихся запросов и динамическая перенастройка при изменении нагрузки;
- автоматическое обслуживание, в том числе определение режимов резервного копирования, реорганизации БД, сбора статистики, обновления программного обеспечения;
- самолечение, в том числе определение:
  - методов сбора данных при мониторинге;
  - методов выявления режимов функционирования системы, близких к предельным;
  - способов фильтрации "диагностического шума", способов связывания журналов с компьютеров с несинхронными таймерами;
  - методов выявления корневой проблемы в каскаде ошибок;
  - способов организации нечеткого поиска в БД симптомов проблем;
  - методов моделирования и определения причины и способов исправления для неизвестных заранее проблем;
- автоматическое управление системой, в том числе:
  - упорядочение задач и назначение им приоритетов;
  - разрешение конфликтов правил и приоритетов;
  - снижение влияния процессов управления на обработку рабочей нагрузки;
  - определение частоты и объема мониторинга, достаточных для разрешения проблем без существенного ухудшения процесса обработки рабочей нагрузки.

Автоматизация администрирования в коммерческих СУБД

Проект **AutoAdmin** [4] выполняется группой Data Management, Exploration and Mining Microsoft Research. Результаты этого проекта используются в Microsoft SQL Server 2000 и Microsoft SQL Server 2005. Основная цель проекта заключается в предоставлении администратору БД специализированных инструментов для повышения производительности системы за счет оптимизации физической структуры БД. При этом под вариантом физической структуры понимается конфигурация — набор индексов, материализованных представлений, кластеров, разбиений.

Система **AutoAdmin** ищет конфигурацию, для которой стоимость выполнения заданного набора запросов (нагрузки, англ. *workload*) при заданных ограничениях на объем памяти минимальна. Для настройки используется информация о нагрузке на СУБД, представленная набо-

ром операторов select, insert, update, delete, полученных при мониторинге системы в рабочем режиме функционирования. Оценка конфигураций выполняется с помощью той же стоимостной модели оптимизатора, что и при реальном выполнении запросов. Сравнение планов выполнения запроса, порождаемых оптимизатором для реально используемой конфигурации и для предлагаемой конфигурации выполняется без реализации последней, что позволяет существенно снизить затраты на оценку конфигураций.

Проект **IBM DB2 Autonomic** [5] основан на внедрении в ядро СУБД средств малозатратного мониторинга нагрузки, такого как регистрация отклонения числа записей, полученных в результате запроса, от априорной оценки этого числа оптимизатором, регистрация частоты обращения к буферной памяти и др. Подобно AutoAdmin [4], подсистема автоматизации DB2 использует рабочий оптимизатор запросов для оценки планов выполнения запросов на предлагаемых конфигурациях, с теми же метриками и моделями стоимости запросов. Для оценки стоимости совместного выполнения множества запросов применяются эвристические высокоуровневые модели. Для снижения нагрузки на сервер, порождаемой подсистемой автоматизации, используется оформление ее заданий в виде пакетов, обрабатываемых в фоновом режиме.

Подсистема автоматизации администрирования СУБД **Oracle Database Server 10g** [6] также основана на использовании обратной связи, состоящей из трех этапов, которым соответствуют компоненты этой подсистемы: наблюдение—диагностика—разрешение (англ. *Observe—Diagnose—Resolve*). Для унификации оценки затрат на обработку запросов вместо отдельных метрик (частота обращения к буферам, задержка чтения с диска/записи на диск и т. д.) вводится единая метрика DbTime (от англ. *database time* — время базы данных), определяемая как суммарное время, потраченное СУБД на обслуживание запроса. Компонент Observe один раз в секунду получает и агрегирует данные об использовании ресурсов, обращениях к объектам. Компонент Diagnose оформлен в виде набора модулей-советников (англ. *Advisors*), выдающих рекомендации, выполнение которых позволит сократить DbTime.

Таким образом, поставщики коммерческих СУБД уделяют основное внимание оптимизации производительности и рационализации использования объемов доступной памяти. Однако основные проблемы автоматизации администрирования ПБД связаны не столько с обеспечением высокой производительности, сколько с устранением возникающих коллизий конвенционального характера, вызванных сложностью взаимодействия индивидуального пользователя с ПБД [7].

Направления автоматизации администрирования ПБД

Функции СУБД порождаются информационными потребностями пользователей. Следует отметить, что информационные потребности порождаются витальными потребностями, однако степень их поддержки путем удовлетворения информационных потребностей с помощью информационной системы может быть различной. Можно сказать, что информация ответа системы может быть релевантна витальной потребности, а может быть и пертинентна ей. Очевидно, что качество системы тем выше, чем выше пертинентность предоставляемой ею пользователю информации.

В процессе функционировании ПБД возможны как повреждения, так и отказы, в том числе сбои. Однако помимо объективных неисправностей могут наблюдаться моральное старение программно-аппаратного обеспечения, неэффективное использование возможностей ПБД по номенклатуре и полноте, ошибочные действия пользователей, в том числе ошибки ввода данных и ошибки, приводящие к невозможности получения требуемой информации пользователем. Зачастую пользователь рассматривает результат собственных некорректных действий как отказ системы, хотя объективно система сохраняет работоспособное состояние. Такое событие можно назвать псевдоотказом (в стандарте IEEE 1044—1993 используется термин "воспринимаемый общий отказ" (англ. *perceived total failure*) [8]).

Будем называть коллизиями все возникающие при функционировании ПБД неисправности, отказы, псевдоотказы, аномалии. Выбор термина "коллизия" (лат. *collisio* — столкновение) обусловлен тем, что обычно под коллизиями понимают столкновение противоположных взглядов, интересов, стремлений, а также некоторые виды противоречий. В нашем случае коллизии, возникающие при взаимодействии пользователей с ПБД, представляют собой противоречия между текущими потребностями пользователей и их удовлетворением.

Следует также отметить, что выражение витальных потребностей, а соответственно и форма, семантика и объем требуемой информационной поддержки, существенно зависят от состояния пользователя, которое претерпевает изменения со временем. Пусть текст T_i^S описывает i-ю функцию ИС, тогда вся система может быть представима текстом T^S:

$$T^S = \bigcup_i T_i^S.$$

Пусть текст $T_j^P(t)$ описывает j-ю частную информационную потребность, тогда общие информационные потребности определяется текстом T^P:

$$T^P(t) = \bigcup_i T_j^P(t).$$

Тогда T^S является функцией T^P:

$$T^S(t) = F[T^P(t)].$$

Поток административных задач Λ определяется построением системы:

$$\Lambda(t) = F_A[F(T^P(t))].$$

Поток коллизий зависит от текущего уровня подготовки пользователя:

$$\Lambda_K(t) = F_K[\Lambda(t),\ T^R(t)],$$

где $T^R(t)$ — текст, определяющий степень готовности пользователя к управлению ПБД.

Текст $T^R(t)$ определяется разностью текстов управления ПБД и текстов, которыми владеет (помнит) пользователь.

Основной задачей системы администрирования ПБД является минимизация интенсивности потока коллизий при максимально полном удовлетворении информационных потребностей пользователя, т. е. обеспечение прозрачности взаимодействия пользователя с ПБД. Будем называть это свойство транспарентностью. Если при удовлетворении информационных потребностей коллизий не возникает, то можно считать, что администрирование данной ПБД по отношению к данному пользователю абсолютно транспарентно. В этом случае наличие администрирования никак себя не проявляет по отношению к пользователю. Другими словами, управление ПБД во всех режимах эксплуатации не вызывает затруднений у пользователя — не создает коллизий, вынуждающих пользователя прибегать к дополнительным трудозатратам на изучение проблемы или привлечение специалистов.

Задачей создания транспарентно администрируемой ПБД являются определение и адаптивное изменение функциональных зависимостей F_K, F_A, F при заданных номенклатуре информационных потребностей, динамических характеристиках их потока в целях минимизации интенсивности потока коллизий:

$$\begin{cases} \Lambda_K(t) = F_K[\Lambda(t),\ T^R(t)] = \\ = F_K\left\{F_A\left[F\left(\bigcup\limits_{j=1}^{N(t)} T_j^P(t)\right)\right],\ \bigcup\limits_{j=1}^{N(t)} T_j^P(t)\right\} \to \min; \\ \bigcup\limits_{j=1}^{N(t)} T_j^P \to \max; \\ \bigcup\limits_{j=1}^{N(t)} T_j^R \leqslant \bigcup\limits_{k=1}^{M(t)} T_k^{RE}, \end{cases}$$

где T_k^{RE} — k-е слово активного словаря пользователя; T_j^R — j-е слово словаря ПБД; $N(t)$ — объем словаря ПБД; $M(t)$ — объем активного словаря пользователя.

Определим степень транспарентности как относительное число коллизий, возникающих у пользователя в процессе удовлетворения информационных потребностей при участии системы администрирования и при ее отсутствии. Абсолютной транспарентности соответствует нулевая интенсивность потока коллизий. Пусть при этом степень транспарентности равна единице. Тогда степень транспарентности администрирования $K(t)$ можно определить через интенсивность потока коллизий следующим образом:

$$K(t) = 1 - \lambda(t),$$

где $\lambda(t)$ — текущая интенсивность потока коллизий, $0 \leqslant \lambda(t) \leqslant 1$.

Если все время взаимодействия тратится на разрешение коллизий, то система администрирования абсолютно не транспарентна ($K(t) = 0$) и полностью лишает пользователя возможности удовлетворять его информационные потребности (пользователь "не видит" ответы на запросы сквозь систему администрирования).

Пользуясь оптической терминологией, можно ввести понятия поглощения, отражения, пропускания информационных запросов в зависимости от транспарентности языка запросов (аналогом является зависимость пропускания/поглощения света от частоты) и другие, которые могут оказаться полезными и наглядными характеристиками процессов взаимодействия пользователя с ПБД.

Так, пропускание света определяется отношением интенсивностей прошедшего и падающего света. В нашем случае это можно интерпретировать как отношение интенсивности исходного потока команд, сформированных пользователем, к интенсивности потока команд, воспринятых системой.

Тогда степень пропускания информационных запросов (т. е. степень транспарентности администрирования) можно определить следующим образом:

$$K(t) = \frac{\lambda_{kv}(t)}{\lambda_k(t)} = \frac{\lambda_k(t) - \lambda_0(t)}{\lambda_k(t)} = 1 - \frac{\lambda(t)}{\lambda_k(t)},$$

где $\lambda_{kv}(t)$ — интенсивность потока команд пользователя; $\lambda_0(t)$ — интенсивность потока команд пользователя, отфильтрованных системой администрирования; $\lambda_k(t)$ — интенсивность потока команд, воспринятых системой.

Это выражение связывает активность пользователя и возникающие при этом коллизии со степенью транспарентности администрирования.

Следует отметить, что степень транспарентности администрирования определяется не только собственно системой администрирования, но и транспарентностью языка пользователя, языка интерфейса ПБД, языков интерфейсов используемых приложений, а также языков интерфейса администрирования. Каждое новое приложение за счет возможности появления дополнительных коллизий, по крайней мере, не увеличивает транспарентность системы.

Таким образом, в качестве критериев качества ПБД можно выделить две взаимосвязанные группы параметров: степень соответствия организации и структуры ПБД информационным и витальным потребностям пользователя и ее транспарентность по отношению к непрофессиональному индивидуальному пользователю.

Функциональная зависимость F отражает принципиальную возможность реализации системой группы частных информационных потребностей. При этом выделяются две основные группы задач администрирования: выбор и настройка программных приложений и создание БД, содержащей необходимую информацию.

Функциональная зависимость F_A отражает интенсивность порождения задач администрирования при данной организации ПБД и ее конкретной аппаратно-программной реализации. На этом уровне также возникают две группы задач по администрированию: техническое администрирование и семантическое администрирование.

Как техническое, так и семантическое администрирование требуют оптимизацию функциональной зависимости F_K для обеспечения согласования словаря ПБД с активным словарем пользователя.

Заключение

Предметная область, которую может отражать ПБД, охватывает большое число различных сторон жизни пользователя. Степень этого охвата может изменяться от функций игровой приставки до полноценной информационной поддержки различных сторон быта, обучения,

образования, развлечений, спорта, творчества, профессиональной деятельности и др. Объем предметной области определяется, в первую очередь, текущими потребностями пользователя. Эти потребности зависят от возраста пользователя, его подготовки, круга интересов, социального статуса, профессии и претерпевают существенные изменения на протяжении жизни.

Современные средства коммуникаций, хранения и обработки информации позволяют накапливать и создавать информационные средства поддержки широкой номенклатуры и большого объема, однако отсутствие средств ее структуризации делает ее труднодоступной, поэтому ПБД в первую очередь должны решать задачу простого и быстрого доступа к накопленной информации, адаптировать параметры ПБД к текущим потребностям и текущему состоянию пользователя.

Список литературы

1. **Chaudhuri S., Dageville B., Lohman G.** Self-Managing Technology in Database Management Systems // Proceedings of the 30th International Conference on Very Large Data Bases, Toronto, Canada, August 31—September 3. (VLDB'04). 2004. P. 1243.
2. **Horn P.** Autonomic computing: IBM's perspective on the state of information technology. IBM Corporation, October 15, 2001. http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf
3. **Ganek A., Gorbi T.** The dawning of the autonomic computing era // IBM Systems Journal, March 2003. N 42-1. http://www.research.ibm.com/journal/sj/421/ganek.pdf
4. **Agrawal S., Bruno N., Chaudhuri S., Narasayya V.** AutoAdmin: Self-Tuning Database Systems Technology // IEEE Computer Society Bulletin of the Technical Committee on Data Engineering. September 2006. Vol. 29. N 3. P. 7—15.
5. **Lightstone S., Lohman G. M., Haas P. J., Markl V., Rao J., Storm A., Surendra M., Zilio D. C.** Making DB2 Products Self-Managing: Strategies and Experiences // IEEE Computer Society Bulletin of the Technical Committee on Data Engineering. September 2006. Vol. 29. N 3. P. 16—23.
6. **Dageville B., Dias K.** Oracle's Self-Tuning Architecture and Solutions // IEEE Computer Society Bulletin of the Technical Committee on Data Engineering. September 2006. Vol. 29. N 3. P. 24—31.
7. **Брейман А. Д.** Задачи администрирования баз данных информационных систем, связанные с конвенциональными ограничениями // Вестник МГАПИ. М.: МГАПИ, 2004. Вып. 1.
8. **IEEE** Std 1044—1993 (2003R) Classification for Software Anomalies.

УДК 004.657

Л. Е. Борчук, аспирант
Череповецкий государственный университет

Совершенствование процесса настройки запросов пользователей на основе асимптотических оценок затрат ресурсов

В связи с непрерывным увеличением объемов и усложнением структуры хранимой в реляционной СУБД информации все актуальнее становится задача локальной настройки запросов пользователей. Рассмотрен вопрос совершенствования решаемых в процессе настройки подзадач путем использования предложенной математической модели затрат ресурсов на основе асимптотических оценок числа действий выполняемых реляционных операций.

В условиях непрерывного увеличения объема и усложнения структуры хранимой информации в реляционных системах управления базами данных (СУБД) возникает прагматическая задача, связанная с необходимостью сохранения стоимости выполнения запроса или недопущения ее существенного повышения. Решение данной задачи связано с учетом следующих факторов.

I. Реляционная модель представления данных не является точной моделью предметной области, так что требуемую информацию можно описать разными способами. В связи с этим одинаковый результат может быть получен не эквивалентными с точки зрения реляционной алгебры запросами. Планы выполнения этих запросов будут различаться, в том числе и стоимостью.

II. В качестве запроса к реляционной СУБД может быть выбран любой из альтернативных способов описания. При этом пользователю нет необходимости обеспечивать выбор запроса с оптимальной стоимостью выполнения. Достаточно, чтобы его стоимость удовлетворяла требованиям на граничное значение. На рис. 1 представлена ситуация, когда текущая стоимость выполнения запроса соответствует требованиям, однако существуют другие варианты написания запроса, планы выполнения которых могут иметь меньшую стоимость. Граничное значение стоимости показано горизонтальной сплошной линией. Каждому запросу может быть поставлено в соответствие несколько планов, одни из которых удовлетворяют требованиям (на рис. 1 показаны точками), другие — нет (показаны крестиками). Штриховой линией обозначена область, содержащая возможные планы выполнения запроса. Оптимизатор волен выбирать способы выполнения только внутри соответствующей запросу области. Всего на рисунке три таких области, что соответствует случаю существования трех не эквивалентных с точки зрения реляционной алгебры запросов, возвращающих одинаковый результат.

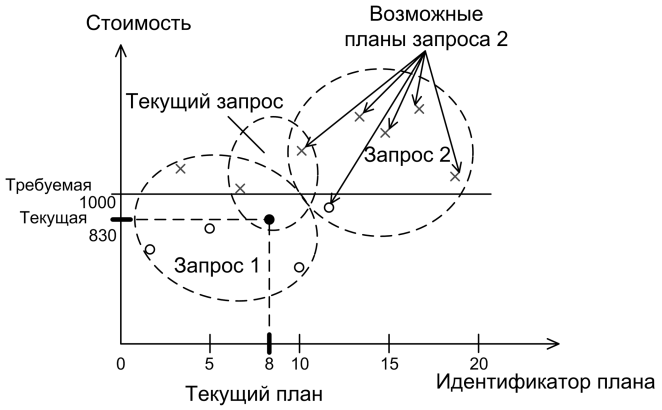


Рис. 1. Пример стоимостей разных планов выполнения различных запросов, возвращающих одинаковый результат

III. С увеличением объемов данных, обрабатываемых в процессе выполнения запроса, может потребоваться настройка системы с целью получить план выполнения со стоимостью, соответствующей требованиям. На рис. 2 показаны планы выполнения запросов рис. 1 с изменившейся вследствие роста объемов обрабатываемых данных стоимостью. Для текущего способа написания запроса стоимость всех возможных способов выполнения больше граничного значения. При этом существуют другие варианты написания запроса (запрос 1 и запрос 2), для которых найдутся способы со стоимостью, удовлетворяющей требованиям. Возникает задача настройки запроса, в ходе решения которой запрос может быть преобразован, и оптимизатор выберет удовлетворяющие требованиям планы (показаны на рисунке стрелками, исходящими от текущего плана выполнения).

IV. Следует ожидать, что с усложнением структуры данных время настройки и требования к инженерной квалификации обслуживающего персонала будут возрастать. Это связано со все возрастающей потребностью хранения разнородных данных в реляционном виде, с одной стороны, и

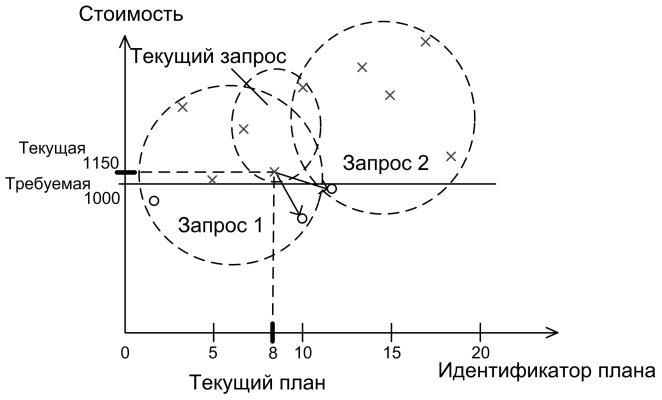


Рис. 2. Пример увеличения стоимостей планов выполнения вследствие роста объемов хранимых данных

упрощением модели вследствие ее реляционного представления — с другой. Несоответствие между моделируемой предметной областью и ее реляционным представлением ведет к тому, что по мере усложнения структуры данных число различных способов написания запроса увеличивается. Это повышает как частоту возникновения задачи настройки, так и ее трудоемкость.

В большинстве случаев настройка запросов выполняется локальным методом [1]. Метод локальной настройки имеет своей целью уменьшение стоимости выполнения отдельных запросов. Задача локальной настройки запросов состоит в нахождении такого плана выполнения среди множества альтернатив, стоимость выполнения которого была бы глобально оптимальная либо не превышала некоторого заданного значения стоимости (квазиоптимальна). Процесс настройки представляет собой итерационную процедуру получения плана выполнения с требуемой стоимостью при известном прогнозе на объемы хранимых данных. В процессе настройки необходимо последовательно решить ряд подзадач:

- провести анализ запроса и сделать оценку возможности его дальнейшей настройки;
- используя существующие рекомендации обработки данных по предметной области, методами индукции или дедукции определить предполагаемый план выполнения
- выполнить настройку запроса для получения предполагаемого плана выполнения;
- оценить полученный результат на соответствие требованиям для прогнозируемых объемов данных.

Анализ запроса в настоящее время выполняется методом экспертных оценок. Целью анализа является выявление частей (компонентов) плана выполнения запроса, являющихся основными потребителями ресурсов. Составными частями анализа являются подзадачи идентификации (выявление ресурсоемких частей плана выполнения) и классификации (определение особенностей представления информации, вызывающих повышенное потребление ресурсов).

Недостатком практикуемого метода экспертных оценок является недетерминированность времени настройки. Она возникает вследствие необходимости угадывания особенности представления информации, вызывающей ресурсоемкие операции, из множества альтернатив. Использование существующих моделей учета затрат ресурсов не нашло широкого применения для решения подзадач идентификации и классификации в связи с большим числом учитываемых параметров модели.

Для определения соответствия стоимости выполнения запроса граничным требованиям на

прогнозируемых объемах данных практикуется проведение процедуры аттестационного тестирования, которая реализуется путем выполнения комплекса тестов для различных наборов данных. Это позволяет выяснить, удовлетворяет ли поведение системы предъявляемым требованиям в достаточно представительном числе сеансов обмена данными. Выполнение комплекса тестов в большинстве случаев требует больших временных затрат, в связи с чем исчерпывающее тестирование является непрактичным как с экономической, так и с технической точек зрения. В результате можно говорить только о повышении вероятности обнаружения несоответствия запроса требованиям. Сокращение объема тестирования приводит к понижению вероятности, что не всегда допустимо.

В работе предпринимается попытка решения подзадач классификации ресурсоемких операций и определения соответствия плана выполнения запроса требованиям путем построения новой математической модели стоимости на основе асимптотических оценок затрат ресурсов на выполнение операций реляционной алгебры. Предлагаемый способ оценки позволяет снизить число учитываемых параметров, что упрощает этапы построения и анализа модели. Как следствие, становится возможным говорить об уменьшении общего времени настройки запроса путем повышения вероятности правильной классификации особенности представления информации, позволяющей получить более эффективный способ написания запроса. Такой эффект достигается путем введения функции ранжирования особенностей. Кроме того, использование математической модели для оценки соответствия требованиям позволяет в ряде случаев отказаться от ресурсоемкой процедуры исчерпывающего аттестационного тестирования. Аналогичные по достоверности результаты, но с существенно меньшими затратами времени на настройку, могут быть получены путем вычисления стоимости на всем прогнозируемом объеме данных.

Компонентами предлагаемой математической модели являются шаги плана выполнения или реляционные операции над отношениями. Эндогенными переменными математической модели являются затраты ресурсов на выполнение реляционной операции. Поставим в соответствие каждому учитываемому ресурсу номер, который обозначим как r ($r = 1, 2, ...$). Экзогенными переменными являются вероятности использования ресурсов при обработке одного кортежа реляционного отношения и статистическая количественная информация о хранимых в реляционной СУБД отношениях и выполняемых над ними реляционных операциях (компонентах). Поставим в соответствие каждой экзогенной переменной номер, который обозначим как t ($t = 1, 2, ...$). Значения экзогенных и эндогенных переменных конечны.

Зададим множество P, представляющее собой множество отношений, хранящихся в БД в материализованном виде (на носителе). Обозначим Q множество отношений, содержащее множество P, пока не определяя его элементы. Определим Ω как множество используемых в процессе выполнения запроса реляционных операций F вида $Q \times Q \to Q$; $Q = < Q, \Omega >$ как алгебру реляционных отношений. Из множества P могут быть получены множества:

$$Q_1 = \{q: q = F(p_1, p_2), p_1, p_2 \in P, F \in \Omega\};$$

$$Q_2 = \{q: q = F(q_1, q_2), q_1, q_2 \in P \cup Q_1, F \in \Omega\};$$

...

$$Q_i = \{q: q = F(q_1, q_2), q_1, q_2 \in$$

$$\in P \cup \left(\bigcup_{j=1}^{i-1} Q_j \right), F \in \Omega\};$$

...

Составим множество Q отношений, являющихся результатом выполнения запросов к исследуемой БД, как $Q = P \cup \left(\bigcup_{j=1}^{\infty} Q_j \right)$. Таким образом, заданная в виде Q алгебра реляционных отношений является P порождающим множеством. Множество $Q\backslash P$ представляет собой множество производных отношений БД, вычисляемых по необходимости с затратами определенных вычислительных ресурсов.

Введем на множестве Q отношение эквивалентности σ такое, что выполняется условие стабильности относительно главных операций. На множестве P два отношения будем считать эквивалентными при совпадении кортежей их именованных типов данных физического представления. На множестве $Q\backslash P$ будем считать, что два отношения эквивалентны, если их выражения реляционной алгебры содержат одни и те же элементы порождающего множества и одни и те же реляционные операции над ними. Фактор-множество Q/σ обозначим W. Эквивалентность σ является также и конгруэнцией для алгебры Q, поэтому вследствие выполнения условия стабильности относительно главных операций порождает фактор-систему $W = Q/\sigma = < W, \Omega >$.

Пусть задано выражение реляционной алгебры для получения ответа на запрос пользователя как отношение w:

$$w = F(w_A, w_B), \qquad (1)$$

где $w \in Q\backslash P/\sigma$ — искомое отношение; A — идентификатор левого операнда операции; B — идентификатор правого операнда операции; $w_A, w_B \in W$ — операнд-отношение, параметры которого известны.

Требуется найти затраты ресурсов на получение отношения w.

Математическую модель затрат ресурсов на выполнение операции будем искать в виде вектор-функции $D(w)$, содержащей t независимых (экзогенных) переменных и r зависимых (эндогенных) переменных затрат ресурсов на получение отношения.

Для $w \in P/\sigma$ вектор-функция $D(w)$ будет задана явно на основе характеристик данного отношения и с учетом специфики его хранения на носителе.

Пусть задан ресурс r.

$D_r(w)$ для $w \in Q \backslash P/\sigma$ будем искать в виде

$$D_r(w) = \Psi_{w,r}(D(w_A),\ D(w_B),\ D_t(w)), \qquad (2)$$

где $D(w_A)$ — вектор характеристик отношения w_A; $D(w_B)$ — вектор характеристик отношения w_B; $D_t(w)$ — вектор известных характеристик операции, может быть измерен, например, при тестовом выполнении запроса пользователя; $\Psi_{w,r}$ — вектор-функция затрат ресурсов на выполнение операции.

В общем случае задача вычисления точного значения функции $\Psi_{w,r}$ эквивалентна задаче выполнения запроса пользователя с точностью до последней операции $F \in \Omega$, так как до выполнения запроса точные значения $D(w_A)$ и $D(w_B)$ могут быть неизвестны.

Способ приближенного вычисления функции $\Psi_{w,r}$ был предложен в работе [2] в виде следующей функции:

$$\Psi_{w,r} = P_{A,r}C_A g_A[D(w_A),\ D(w_B)] + \\ + P_{B,r}C_B g_B[D(w_A),\ D(w_B)], \qquad (3)$$

где $P_{A,\ r}$, $P_{B,\ r}$ — вероятность использования ресурса при обработке одного кортежа отношений-операндов A и B соответственно; $g_A[D(w_A),\ D(w_B)]$, $g_B[D(w_A),\ D(w_B)]$ — асимптотическая оценка O числа обрабатываемых кортежей операнда A и B соответственно; C_A, C_B — значение констант асимптотической оценки O при непредельных значениях параметров $D(w_A)$ и $D(w_B)$ соответственно.

Методика построения модели запроса на основе предложенной модели операции (3) представляет собой следующую итерационную процедуру [3, 4].

1. Анализ системы существующими программными средствами. Определение запроса пользователя, подлежащего настройке, целей настройки и доступных для измерения параметров. Определение диапазонов изменения параметров.

2. Выделение значимых показателей. Построение асимптотических моделей операций реляционной алгебры $g[.]$.

3. Выполнение запросов пользователей на существующем наборе данных, определение характеристик (времени выполнения, числа итераций) каждой операции плана $D_t(w)$.

4. Последовательное построение моделей (3) для каждой операции в порядке их выполнения.

5. Выбор среди всех возможных моделей запроса модели, реализующей максимум времени выполнения.

6. Определение адекватности модели внутри области изменения параметров.

Построение модели затрат ресурсов рассмотрим применительно к СУБД Oracle версии 9.2.0.5, ОС Windows 2003, процессор Intel Pentium 4, объем оперативной памяти 1 Гбайт, жесткий диск SATA 7200 rpm. Физически организация способа хранения таблиц в СУБД — локально управляемое табличное пространство с ручным управлением списка сегментов, размер блока данных 4 Кбайт. В БД содержатся две таблицы A и B, соответствующие реляционной модели хранения двух связанных отношением 1:М сущностей. По таблицам построены соответственно уникальные индексы *A_id, B_id* по первичным ключам и неуникальный индекс *A_idx* по одному из полей, содержащему 100 распределенных по нормальному закону значений.

Выберем в качестве асимптотик модели число строк n_A таблицы A, число строк n_B таблицы B и селективность r_{A_idx} индекса *A_idx*. Такое предположение определяется моделью "сущность—связь", а также видом исследуемых операций.

Точные модели операций, требуемые для построения, изложены в [5]. Основы построения асимптотических моделей операций реляционной алгебры и способы оценки их адекватности предложены в [2—4].

На примере ситуации, допускающей локальную настройку, покажем процесс решения подзадач идентификации и классификации. Для этого определим функцию ранжирования параметров модели следующим образом. Выберем точку на границе прогнозируемого объема данных и зафиксируем в ней значения всех параметров, кроме одного. Прирост стоимости при изменении значения этого параметра от текущей (используемой при вычислении коэффициентов модели) до прогнозируемого значения назовем его весом. Значение функции ранжирования в точке — совокупность упорядоченных по убыванию весов параметров.

В примере 1 рассмотрен процесс локальной настройки запроса посредством использования свойства транзитивности и моделей затрат ресурсов на основе асимптотических оценок. Для простоты будем рассматривать не все виды ресурсов, а ограничимся только оценкой числа обработанных блоков данных, поскольку на практике уменьшить время выполнения запроса можно за счет их числа.

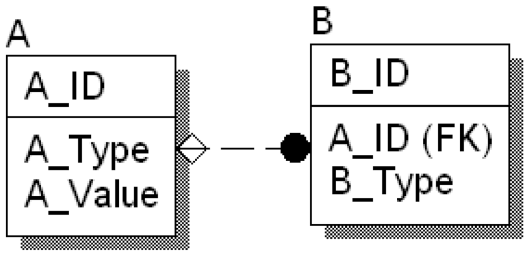


Рис. 3. E-R-диаграмма модели хранения оперативной и архивной информации

Пример 1. *Локальная настройка посредством использования свойства транзитивности.*

Рассмотрим модель реляционного представления данных в виде двух связанных отношением 1:M сущностей <архивные данные> — <оперативные данные > (рис. 3), хранимых в таблицах A и B соответственно. Требуется выбрать записи определенного типа из архивной таблицы A, присутствующие в оперативной таблице B.

Запрос на выборку и план его выполнения приведены на рис. 4.

По плану выполнения (рис. 4) итерационно была построена модель стоимости отношения w, содержащего ответ на запрос пользователя, которая после проведения процедуры рекурсивного раскрытия на основе одного тестового испытания при $n_B = 1000$ и $r_{A\_idx} = 1$ определяется соотношением

$$\Psi_{w_5} = 0{,}085n_B + 2n_B r_{A\_idx}. \qquad (4)$$

В (4) значение параметра $r_{A\_idx} = 1$ при данном способе моделирования (см. рис. 3) является минимальным. Так что согласно введенной функции ранжирования, уменьшение затрат ресурсов следует начинать с уменьшения числа обрабатываемых строк n_B. При этом в процессе выполнения запроса требуется около 4 % ресурсов потратить на чтение таблицы B (первое слагаемое) и 96 % на обработку каждой ее строки для определения соответствующего кортежа таблицы A. Для уменьшения числа обрабатываемых строк n_B и, как следствие, затрат ресурсов перепишем запрос, использовав предикат ограничения `A_type = 0` на таблице B за счет свойства транзитивности (`A. A_type = B. B_type`), не отраженного в представленной на рис. 3 реляционной модели. Это позволило уменьшить число обрабатываемых строк таблицы B с 1000 до 83.

```
SQL> explain plan for select * from A, B
       where A.A_id = B.A_id and A.A_type = 0;
-------------------------------------------------------------------------------
| Id  | Operation                      |  Name       | Rows  | Bytes | Cost  |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |             |  1000 |   109K|  2085 |
|   1 |   NESTED LOOPS                 |             |  1000 |   109K|  2085 |
|   2 |    TABLE ACCESS FULL           | B           |  1000 |  6000 |    85 |
|*  3 |    TABLE ACCESS BY INDEX ROWID| A           |     1 |   106 |     2 |
|*  4 |     INDEX UNIQUE SCAN          | A_ID        |     1 |       |     1 |
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   3 - filter("A"."A_TYPE"=0)
   4 - access("A"."A_ID"="B"."A_ID")
Statistics
----------------------------------------------------------
       3556  consistent gets
       1550  physical reads
```

Рис. 4. Запрос на выборку данных из оперативной и архивной таблиц и план его выполнения

Модель затрат ресурсов при выполнении нового запроса определяется соотношением

$$\Psi_w = 0{,}085n_B + 0{,}166n_B r_{A\_idx}. \qquad (5)$$

В рассмотренном примере использование математической модели позволило определить, что изменить затраты ресурсов можно за счет ограничения числа обрабатываемых строк таблицы B. Дальнейшая модификация запроса в целях изменения кардинальности проводилась исследователем на основании знаний о предметной области и используемой реляционной модели. □

На основе модели затрат ресурсов проведем сравнительный анализ производительности различных способов организации доступа к данным. Пусть имеется секционированная по ключу поиска таблица данных. Требуется оценить затраты ресурсов на поиск данных по сравнению с поиском в несекционированной таблице с построенным по ключу поиска неуникальным индексом. Процесс построения моделей планов выполнения запросов и их анализ рассмотрены в примере 2.

Пример 2. *Сравнение затрат ресурсов на выполнение запроса на доступ к секционированным данным и сканирования неуникального индекса.*

Запрос на поиск данных в секционированной таблице по ключу секционирования и план его выполнения приведены на рис. 5.

Выберем параметрами модели число строк секции m_A и общее число секций k. Но плану выполнения рис. 5 итерационно была построена модель стоимости отношения w, содержащего ответ на запрос пользователя, которая после проведения процедуры рекурсивного раскрытия на основе одного тестового испытания при $m_A = 1 \cdot 10^4$ и $k = 10$ определяется соотношением

$$\Psi_w = 0{,}395 \cdot 10^{-5} m_A \times \\ \times (0{,}1419\ln(k) - 0{,}0126), \qquad (6)$$

где величина $(0{,}1419\ln(k) - 0{,}0126)$ характеризует уменьшение числа многоблочных чтений при уменьшении секции.

В целях оценки сокращения времени исследования системы была проведена практикуемая для этого процедура аттестационного

```
SQL> explain plan for select * from A where val=:v1;
-------------------------------------------------------------------------------
| Id  | Operation             | Name       | Rows  | Bytes | Cost  | Pstart| Pstop |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |            | 10000 |  1035K| 15182 |       |       |
|   1 |  PARTITION RANGE SINGLE|           |       |       |       |   KEY |   KEY |
|*  2 |   TABLE ACCESS FULL   | A          | 10000 |  1035K| 15182 |   KEY |   KEY |
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
 2 - filter(A.VAL=TO_NUMBER(:Z))
```

Рис. 5. Запрос на линейный поиск данных в секционированной таблице и план его выполнения

```
SQL> explain plan for select * from A where val=:v1;
-------------------------------------------------------------------------------
| Id  | Operation                    | Name       | Rows  | Bytes | Cost  |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |            | 10000 |  1035K| 10042 |
|   1 |  TABLE ACCESS BY INDEX ROWID| A          | 10000 |  1035K| 10042 |
|*  2 |   INDEX RANGE SCAN           | A_idx      | 10000 |       |    42 |
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
 2 - access(A.VAL=TO_NUMBER(:Z))
```

Рис. 6. Запрос на поиск данных по значению ключа неуникального индекса и план его выполнения

тестирования. Для ее проведения был создан набор таблиц с различным числом секций (от 10 до 100 с шагом 10), что заняло порядка 12 ч. Составление комплекса тестов заняло 0,5 ч. Далее была проведена процедура аттестационного тестирования, время выполнения которой составило 0,2 ч. Процесс построения модели запроса (6) занял 0,5 ч. Время вычисления модели в точке оценивается величиной $O\ln(m_A)$ и по сравнению с другими этапами пренебрежимо мало. Результаты тестирования показали, что относительная ошибка модели не превышает 5 % от фактических затрат времени на выполнение запроса.

Сравним модель затрат ресурсов при доступе к секционированным данным с моделью затрат сканирования неуникального индекса. Запрос пользователя на поиск данных и план его выполнения, содержащий операцию доступа по значению ключа неуникального индекса, представлены на рис. 6.

По плану выполнения (рис. 6) итерационно была построена модель стоимости отношения w, содержащего ответ на запрос пользователя, которая определяется соотношением

$$\Psi_w = 0{,}8 \cdot 10^{-3} r_{A\_idx}. \qquad (7)$$

Сравним модели. Пусть $m_A = 1 \cdot 10^4$, $k = 100$.

Определим, при каких значениях параметра r_{A_idx}, выраженного как $r_{A\_idx} = m_A x$, $0 < x \leqslant 1$, сканирование неуникального индекса будет иметь меньшую оценку, чем полное сканирование секции. Для этого необходимо решить уравнение $0{,}0008 \times \times 1000x = 3{,}954 \cdot 0{,}1 \cdot 0{,}6409$, откуда $x = 0{,}3167$. Резюмируя, секционирование числом секций $k = 100$ будет менее выгодным по сравнению с построением неуникального индекса, содержащего в качестве лидирующего столбца ключ секционирования только в том случае, если индекс имеет селективность, меньшую чем 0,31 от числа строк секции. Подобный эффект достигается использованием многоблочных чтений при доступе к данным, хранимым на диске. При этом число считываемых за одну порцию блоков данных секции снижается по мере роста числа секций, так что выбор той или иной схемы разбиения данных зависит от специфики предметной области и способа ее моделирования. □

Проведенное исследование процесса настройки запросов с использованием моделей затрат ресурсов на основе асимптотических оценок выявило следующие пути его совершенствования.

I. Математическое моделирование позволяет формализовать этап идентификации и классификации проблемы при настройке запроса пользователя. Однако для решения задачи настройки необходимо по завершению этапа классификации сделать дедуктивный вывод о способе модификации запроса или системы. Предлагаемая модель пока не позволяет решить задачу в полном объеме. Тем не менее, проведенная формализация дает возможность сократить сроки настройки за счет повышения вероятности правильной классификации и использования для этого средств автоматизации.

II. Предлагаемая модель на основе асимптотических оценок затрат ресурсов позволяет проводить оценку числа логических чтений, не отличающуюся от результатов аттестационного тестирования более чем на 5 %. Вместе с этим использование модели в рассматриваемом примере позволило сократить время проведения оценки в более чем 25 раз. Результаты моделирования могут быть использованы для оценки системы и в более широкой области изменения параметров, однако в этом случае могут потребоваться однократные проверки адекватности с помощью тестов в некоторых точках области.

Список литературы

1. **Новиков Б. А., Домбровская Г. Р.** Настройка приложений баз данных. СПб.: БХВ-Петербург, 2006. 240 с.
2. **Борчук Л. Е.** Асимптотическая модель затрат ресурсов вычислительной системы на выполнение реляционного запроса в РСУБД System R // Кибернетика и высокие технологии XXI века "C&T*2006": Матер. 7-й междунар. науч.-техн. конф. Воронеж: ВГУ, 2006. С. 363—373.
3. **Плашенков В. В., Борчук Л. Е.** О модели асимптотической оценки ресурсоемкости реляционного запроса // Интеллектуальные системы и компьютерные науки: Сб. тр. участников 9-й междунар. конф. М.: Мехмат МГУ, 2006. Т. 1. Ч. 2. С. 198—204.
4. **Борчук Л. Е.** Анализ адекватности одной математической модели реляционного запроса // Моделирование. Теория, методы и средства: Матер. VI междунар. науч.-практ. конф. Новочеркасск: ЮРГТУ, 2006. С. 50—54.
5. **Lewis J.** Cost-Based Oracle Fundamentals. Apress, 2006. 506 p.

УДК 519.686

Д. А. Максименков,
ЗАО "МЦСТ", г. Москва
shark@mcst.ru

Система генерации тестов со случайным набором команд в кодах x86

При создании сложных программных продуктов, таких как системы бинарной компиляции, требуется широкий набор различных отладочных тестов. В статье рассмотрен опыт создания генератора тестов в кодах архитектуры x86 со случайной последовательностью команд. Приведен анализ ряда практических проблем автоматической генерации и предлагаются методы их эффективного решения.

Введение

В статье рассматривается технология тестирования системы бинарной компиляции (СБК) [1], разрабатываемой для вычислительного комплекса "Эльбрус-3М1" (далее ВК) [1]. Основные компоненты СБК — интерпретатор, шаблонный транслятор, оптимизирующие компиляторы 3-го и 4-го уровней, профилировщик и наборщик регионов. Главная цель СБК — обеспечение возможности запуска приложений в бинарных кодах одной платформы на другой архитектурной платформе. В зависимости от профиля исполняемого кода используются те или иные компоненты СБК, обеспечивающие наиболее эффективное исполнение кода.

Для тестирования этих компонентов, самих средств отладки и для проверки аппаратуры ВК предлагается технология генерации тестов в кодах исходной архитектуры (в рассматриваемой СБК это архитектура IA-32 [2]). Генераторы тестов для тестирования бинарных компиляторов и аппаратуры можно встретить, например, в проектах корпораций *Transmeta* — TiGeR [3], *IBM* — Genesys-x86 [4] и др.

Статья является продолжением публикаций [5, 6], в которых излагаются способы организации тестирования оптимизирующего бинарного компилятора на примере создания генератора самопроверяющихся тестов со сложным графом управления для архитектуры IA-32. Такие тесты используют довольно большой, но все же ограниченный, набор наиболее часто используемых команд, учитывают особенности оптимизирующего бинарного компилятора и хорошо подходят для организации круглосуточных запусков при контроле надежности системы. В них не возникают какие-либо исключительные ситуации, все они являются детерминированными и хорошо подходят для тестирования статических оптимизирующих компиляторов [1], что зачастую позволяет упростить процедуру выявления ошибок.

Ниже предлагается еще одна технология отладки на примере создания генератора тестов со случайным набором инструкций. Целью тестирования с их помощью является проверка максимально большого числа всевозможных последовательностей инструкций IA-32 (в том числе и недопустимых), редко встречающихся в реальных задачах, и их различных комбинаций на СБК ВК. Также одной из главных целей рассматриваемого тестирования является формирование маленьких простых тестов, удобных для анализа и не требующих больших ресурсов при их генерации и запуске. Помимо этого ставится задача создания тестов, способных вырабатывать различные исключительные ситуации, что позволило бы проверить адекватное поведение системы при возникновении различных типов исключительных ситуаций во время исполнения кодов IA-32.

Описание технологии

В CISC-архитектурах, к которым, в частности, относится и семейство процессоров IA-32 длина команды не имеет фиксированного размера и может варьироваться от 1 до 15 байт [2]. Таким образом, чтобы перебрать все возможные операции, необходимо сформировать 256^{15} различных вариантов сочетаний бит и проверить их работоспособность на СБК, с одной стороны, а также использовать при тестировании аппаратуры ВК, с другой. Число различных комбинаций таких команд резко увеличивается с ростом длины цепочки операций. Нельзя забывать и о большом разнообразии возможных аргументов для таких команд. Итак, в нашем распоряжении имеется большое поле деятельности в отношении проверки надежности СБК. Особый интерес в этом переборе команд представляют операции, редко или вообще не встречающиеся в реальных задачах, и таким образом являющиеся либо плохо оттестированными, либо вообще не тестированными на рассматриваемой СБК. Также интересна реакция оптимизирующих компонентов компилятора на случайные комбинации команд, хотя для большей части таких инструкций возможность попадания в регионы [1], подвергающиеся в дальнейшем оптимизациям, ограничена. В то же время срабатывания этих ограничений и корректную работу компилятора в таких ситуациях также необходимо проверить.

От случайных комбинаций команд со случайными аргументами стоит ожидать различных исключительных ситуаций: в цепочке инструкций может оказаться недопустимая операция, обращение к закрытой на чтение/запись странице памяти, переход на не принадлежащий программе адрес, некорректный вызов функции

и т. д. Здесь возможен любой ситуационный сценарий, приводящий к возникновению различных исключений. Все эти ситуации также представляют интерес при проверке надежности системы. СБК должна быть отказоустойчивой при исполнении любой комбинации байт, интерпретируемой в качестве команд процессора x86.

Алгоритм генерации тестов со случайным набором команд может быть следующим. За основу берется шаблон простой программы на ассемблере, где вместо тела теста стоит макрос, определенный в отдельном внешнем файле. Это делается для того, чтобы отделить основной код исходного теста программы на ассемблере, который не меняется при каждой генерации нового теста, от изменяемой ее части — подменяемого макроса (рис. 1).

Данный макрос представляет собой строчку символов, заполненную произвольным набором байтов. Например, это может быть строчка из потока случайных символов /dev/urandom в ОС Linux (рис. 2). Длина строчки L варьируется режимом тестирования и может составлять от нескольких символов до сотен байт, в зависимости от целей тестирования. Нужно отметить тот факт, что при увеличении длины генерируемой строки возрастает вероятность появления в тесте некорректных инструкций, приводящих, в свою очередь, к проявлению различных исключительных ситуаций. В случае успешной отработки тела теста (подставляемого макроса из случайных команд) приложение завершается системным вызовом exit с кодом возврата 0. Именно по нему можно будет судить об успешной работе созданной цепочки операций.

После того, как строчка случайных символов занесена в тело макроса, находящегося в отдельном файле, запускается процесс компиляции и линковки теста. Эта операция не потребует больших ресурсов для такого небольшого кода. При исполнении задачи описанным способом в ОС Linux результатом ее работы может быть одна из следующих ситуаций:

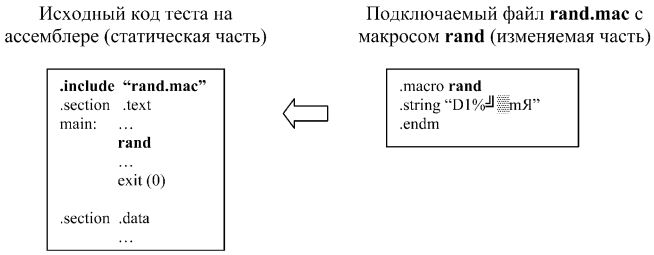


Рис. 1. Исходные коды теста на ассемблере

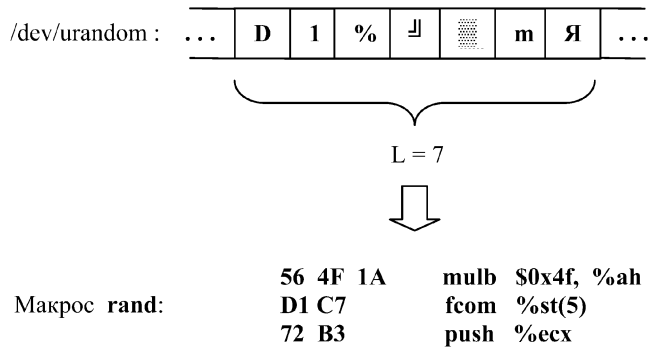


Рис. 2. Механизм формирования случайной цепочки команд

- успешное завершение с нулевым кодом возврата и передача управления в командный интерпретатор;
- зацикливание теста (довольно редкий случай — при возникновении бесконечного цикла из случайных инструкций макроса);
- выработка тестом сигнала, являющегося следствием возникновения исключительной ситуации (*Segmentation fault, Illegal instruction* и т. д.) — наиболее вероятная ситуация, напрямую зависящая от числа созданных случайных команд.

В первом случае никаких дополнительных действий предпринимать не нужно.

В случае зацикливания теста поможет запуск с ограничением по времени работы задачи. Например, для командного интерпретатора bash в ОС Linux строчка запуска может выглядеть так:

bash—icv "ulimit—t**$TIME**;./**$TEST**",

где TIME — время, отводимое для работы теста в секундах; TEST — имя запускаемого теста. Как показал опыт, нескольких секунд вполне достаточно для того, чтобы задачи, не имеющие бесконечных циклов, успели завершить свою работу.

Поведение, описанное в последнем случае, было бы критичным для бинарных компиляторов приложений. Например, бинарный компилятор *Execution Layer* фирмы *Intel* для процессоров *Itanium2* [7] не гарантирует корректного исполнения задачи при возникновении в результате ее вычислений не-чисел (Nan, Onan, Inf), не говоря уже о корректной работе приложения со случайным набором команд. В СБК, реализованной на процессоре "Эльбрус" ВК "Эльбрус-3М1", такое поведение тестов вполне допустимо, и система должна отрабатывать возникающие сигналы исключительных ситуаций аналогично архитектуре IA-32. Именно это и позволяет проверять тесты, создаваемые описанным алгоритмом.

Однако при генерации достаточно длинных цепочек случайных байтов, интерпретируемых в качестве инструкций процессора, у команд, находящихся в конце программы, при описанном подходе вероятность быть исполненными оказывается меньше всего, ввиду возможных прерываний на более ранних операциях. На рис. 3 (см. третью сторону обложки) приведена гистограмма вероятности возникновения исключительной ситуации при исполнении таких команд, сформированных из случайной последовательности байтов. Как видно из гистограммы, при интерпретации 10 байт кода в качестве инструкций x86 вероятность возникновения сигнала оказывается выше 90 %.

Для работы шаблонного кода системы бинарной трансляции и оптимизирующих компиляторов необходимо многократное повторение одних и тех же команд приложения для определения системой "горячих" участков задачи с целью их дальнейшей оптимизации [1]. При описанной реализации теста такого эффекта можно достичь несколькими повторными запусками одного и того же приложения. Но этот способ оказывается неэффективным вследствие сравнимости времени запуска/окончания работы задачи в ОС со временем исполнения инструкций самого теста. Гораздо эффективней было бы иметь внутренний цикл программы, повторяющий цепочку случайных операций заранее заданное число раз. Но как этого добиться, если вероятность возникновения

прерывания (а значит и досрочного завершения работы задачи) при генерации случайной последовательности команд оказывается довольно высока?

Предлагается следующий механизм, устраняющий описанные выше проблемы. На наиболее часто возникающие исключительные ситуации в самом тесте проводится регистрация обработчика сигналов — функции, вызываемой при появлении исключительных ситуаций в задаче. Рис. 3 (см. третью сторону обложки) показывает статистику возникновения различных сигналов для случайной цепочки байтов. Оказывается, что при длине цепочки команд в 10 байт вероятности появления сигналов SEGV (*Segmentation fault*) и ILL (*Illegal instruction*) составляют 90 % и 5 % соответственно от общего числа возникающих сигналов. Такая тенденция сохраняется и при более длинных цепочках байтов, исполняемых в качестве команд процессора x86. Остальные 5 % составляют сигналы TRAP, BUS, FPE, INT и другие, встречающиеся достаточно редко. Такое неравномерное распределение вероятности возникновения различных типов прерываний создает дисбаланс в тестировании исключительных ситуаций. Для подавления "лидеров" — сигналов SEGV и ILL в тесте регистрируется обработчик на эти типы сигналов. Тем самым мы собираемся откорректировать работу теста в 9 из 10 случаев возникновения исключительных ситуаций при исполнении программы. Кроме того, коррекция этих ситуаций должна уменьшить число случаев досрочного завершения работы теста из-за возникающих прерываний. Это позволит увеличить число тестов, имеющих "горячие" участки кода, т. е. пригодных для набора регионов (тесты с кодом возврата 0). Одновременно возрастает вероятность проявления при работе теста других исключительных ситуаций, ранее маскируемых сигналами SEGV и ILL.

Операционная система в момент выполнения инструкции, приведшей к возникновению исключительной ситуации, посылает тесту сигнал. Работа программы прерывается с сохранением ее контекста, и управление передается соответствующему обработчику сигнала. В случае отсутствия в пользовательском приложении такого обработчика запускается стандартный обработчик, который завершает работу программы с соответствующим данному сигналу ненулевым кодом возврата. Так, например, для сигнала SEGV код возврата равен 139. Если же программист хочет отработать данную исключительную ситуацию особым образом (как обстоит дело в нашем случае) и для этих целей зарегистрировал свой обработчик сигналов, то ОС передает управление обработчику (в нашем случае единому для нескольких сигналов) вместе с дополнительной информацией о контексте возникновения исключения, в том числе и адрес команды.

Внутри самого обработчика запускается алгоритм, корректирующий в теле теста код инструкции, на которой возникло исключение, путем замены ее другой случайной цепочкой значений, начиная с адреса некорректной команды. Чтобы исправить ситуацию, заменять всю цепочку не нужно — достаточно модифицировать несколько первых байтов. Замена должна иметь случайный, но детерминированный характер. Новые значения будут зависеть от значений байтов старой цепочки. Это делается для того, чтобы поведение теста не менялось от запуска к запуску. В описываемой реализации используется коррекция 32 бит цепочки (переменная value) по следующей формуле генератора псевдослучайных чисел:

```
long value;
value = 1664525L *value + 1013904223L.
```

После коррекции команды (фактически ее подмены) обработчиком, управление возвращается в тело теста на адрес, на котором было прервано его исполнение, и восстанавливается исходный контекст. Далее снова делается попытка выполнить инструкцию, находящуюся по этому адресу. В этот раз там окажутся новые значения байтов кода, которые в соответствии с системой команд IA-32 будут интерпретированы как другая команда. Если при ее исполнении вновь возникнет исключительная ситуация, мы опять окажемся в обработчике сигналов и заново скорректируем код. Если же команда получилась корректной, то тест продолжит свою работу до следующей недопустимой команды. Этот процесс будет повторяться до тех пор, пока мы не достигнем конца цепочки операций. Иллюстрация описанного выше алгоритма приведена на рис. 4 (см. третью сторону обложки).

Здесь в качестве примера приведена цепочка из 8 команд макроса rand длиной в 13 байтов (рис. 4, *а*). Цифрами показаны длины команд. Во время исполнения 2-й инструкции размером в 3 байт возникла исключительная ситуация. Управление передается обработчику, который изменяет 4 байт цепочки, начиная с адреса команды, на которой возникло исключение. В результате на месте второй команды получилась новая цепочка операций, состоящей из двух-, одно- и начала трехбайтовой операции (рис. 4, *б*). Все они отрабатывают успешно. Пятая команда цепочки (рис. 4, *б*) оказывается некорректной, и управление снова передается обработчику. Образованный им код (рис. 4, *в*) оказался также недопустимым, и мы вновь запускаем функцию, корректирующую исполняемый код. Полученная цепочка (рис. 4, *г*) дорабатывает до конца макроса **rand** успешно.

Чтобы избежать затирания команд, которые стоят в тесте после вызова макроса, во время процесса коррекции недопустимого кода, обработчиком сигналов после макроса **rand** в тесте изначально добавляется несколько **nop** — команд-пустышек. Таким образом, если исключение возникло на однобайтовой команде, являющейся последней в цепочке операций макроса, то функция-корректор исправит эти операции **nop**, не испортив идущий далее осмысленный код теста (например, системный вызов exit).

Теперь у нас есть цепочка команд, образованная самим тестом в результате нескольких итераций автокоррекции кода. Полученную последовательность инструкций процессора уже можно поместить в цикл, число итераций которого будет достаточным для формирования в тесте "горячего" участка исполнения кода программы с заданным числом повторений. Запуск такого теста в системе бинарной компиляции позволит подключить к работе важные компоненты системы [1]. Например, начнет работу шаблонный транслятор, профилировщик, оптимизирующие компиляторы разных уровней и другие сложные компоненты системы [1]. Это дает возможность тестирования перечисленных компонентов СБК и аппаратуры ВК "Эльбрус-3М1" при работе на случайной цепочке команд.

Далее, при создании тестов необходимо обратить внимание на воспроизводимость выявляемых ими ошибочных ситуаций. Невоспроизводимых ошибок не суще-

ствует. Бывают не до конца воспроизведены условия (окружение запуска), при котором возникла ошибка. Для того, чтобы повысить вероятность получения детерминированного теста, а значит, в случае обнаружения ошибки повысить шансы на ее воспроизведение, необходимо учитывать следующее. Поскольку создаваемый код теста является случайным и может при вычислениях использовать различные значения регистров, то разумно было бы перед его исполнением провести инициализацию всех регистров общего назначения, а также статусных и управляющих регистров [2]. Значения для инициализации также могут быть получены случайным образом, как и при получении случайных команд. То есть в сегмент данных вставляется макрос со случайной строчкой символов, интерпретируемой далее как значения для перечисленных выше регистров, инициализация которых стоит в начале теста. Теперь исходный код нашего теста будет выглядеть, как показано на рис. 5.

В начале теста регистрируется обработчик сигналов на ситуации, описанные выше (SEGV и ILL). Далее идет инициализация регистров (целочисленных, вещественных, векторных, статусных и управляющих), значения которых могут влиять на ход исполнения программы. Инициализация проводится посредством случайных значений бит, доступных по метке **mem** в сегменте данных. В данном случае за этой меткой расположен макрос **rand2,** определяемый аналогично макросу **rand** в подключаемом файле **rand.mac.** Размер строки во втором макросе должен быть достаточным для независимой инициализации всех регистров (на рисунке приведен пример укороченной строки в макросе **rand2**).

После инициализации регистров в программе организован цикл с управляющей 32-битной переменной **counter** в памяти, обеспечивающей выполнение 32 тыс. итераций тела цикла. В качестве последнего выступает случайный набор команд, определенный в макросе **rand**. За ним следуют три операции **nop** для предотвращения описанного выше краевого эффекта при самомодификации кода. В сегменте данных помимо массива **mem** инициализируется счетчик цикла **counter**. Число итераций цикла можно изменить, задав необходимое начальное значение этой переменной.

При инициализированных всех регистрах и коррекции обращений в недопустимую память вероятность недетерминированного поведения задачи и использование неинициализированных данных, влияющих на поведение задачи, резко уменьшается. Также стоит проводить

Исходный код теста на ассемблере (статическая часть)

```
.include  "rand.mac"
.section  .text
main:     init Handler
          …
          movl mem, %eax
          movl mem+4, %ebx
          movl mem+8, %ecx
          …
loop:     rand
          nop (3)
          decl counter
          jnz loop
          exit (0)

.section  .data
counter: .long  32000
mem:      rand2
```

Подключаемый файл **rand.mac** с макросами (изменяемая часть)

```
.macro rand
.string "D1%╜▒mЯ"
.endm

.macro rand2
.string "Ы<#2+(sV.q~w¶"
.endm
```

Рис. 5. Модифицированный исходный код теста

запуск теста в ОС с обнулением окружения командного интерпретатора, т. е. предложенная выше строчка запуска принимает вид:

bash—norc—icv "ulimit—t**$TIME**; exec—c./**$TEST**"

Однако несмотря на все наши усилия остается вероятность некорректного поведения задачи, например, при генерации перехода на доступный адрес внутри теста или вызов библиотечной функции с некорректными параметрами. Несомненно, внутри такого кода произойдет исключительная ситуация и никакая модификация инструкций не позволит корректно закончить работу теста. Такие случаи следует отсеивать и не рассматривать в процессе тестирования как допустимые. Для этого внутри обработчика сигналов реализован механизм контроля модифицируемых адресов. Если адрес команды, на которой возникло исключение, принадлежит промежутку значений от метки loop до адреса loop+L, код нужно скорректировать и продолжить его исполнение. Если же по каким-то причинам мы оказались вне макроса **rand**, то управление теста было передано в недопустимую область программы, и продолжать исполнение кода бессмысленно. В этом случае управление передается системному вызову exit с кодом возврата — 1. Именно по нему мы можем отсеивать такие "бракованные" тесты.

Результаты

После отсеивания "бракованных" тестов, составляющих почти 50 % от общего числа получаемых при генерации задач, для оставшихся тестов распределение получаемых кодов возвратов приведено на круговой гистограмме (рис. 6, см. третью сторону обложки). 57 % тестов заканчивают свою работу с кодом возврата 0, т. е. являются пригодными для тестирования различных компонентов системы бинарной компиляции (на таких тестах цепочка случайных команд повторяется заданное число раз, достаточное для формирования региона). Несмотря на регистрацию обработчика сигналов SEGV, 19 % тестов завершают работу с кодом 139. Такая ситуация возникает при переполнении стека в результате исполнения случайной цепочки команд или вследствие перезаписи регистра %esp — стекового указателя. С бесконечными или долго работающим циклами получаются 8 % тестов. Остальные 16 % составляют другие сигналы, такие как TRAP, BUS, FPE, INT и т. д. Добиться уменьшения тех или иных исключений можно, зарегистрировав тот же самый обработчик на соответствующий тип сигналов, при этом процент тестов с кодом возврата 0 будет увеличиваться.

Тесты, получаемые генератором, реализованным по описанной выше технологии, имеют следующие свойства:

- они невелики по размеру (несколько килобайт) и имеют небольшое время исполнения (меньше 1 с);
- их формирование не требует больших временных и машинных ресурсов, а также сложных скриптов генерации и запуска;
- благодаря своей простоте и небольшому размеру они удобны для анализа и разбора ошибок;
- вероятность воспроизведения ошибок, выявленных на таких тестах, довольно высока;
- возможность легкого повторного воспроизведения всех обнаруженных на них ошибок;
- в получаемых тестах есть самомодификация кода, что позволяет проверить некоторые режимы рабо-

ты наборщика регионов и базы кодов регионов системы [1];
- во время исполнения тестов возможно возникновение любых исключительных ситуаций, что также актуально при отладке системы;
- в создаваемых тестах возможны любые (что немаловажно) последовательности IA-32 инструкций (в том числе и недопустимые), позволяющие проверить работу СБК при различных комбинациях команд и их аргументов;
- такие тесты адаптированы для активизации работы всех уровней СБК, благодаря наличию в них "горячих" участков кода внутри циклов.

Для тестирования можно использовать два режима генерации тестов:
- более простой, с запуском непосредственно на отлаживаемой системе процесса генерации теста, его сборку и исполнение (генерация тестов без эталонной x86-машины);
- более сложный, с регистрацией обработчиков сигналов для автокоррекции кода, с отсеиванием "бракованных" тестов на эталонной машине и с последующим запуском итогового набора тестов на СБК.

Получаемые тесты не являются самопроверяющимися, поэтому тестирование и выявление ошибок при их запуске рассчитано на внешние средства верификации. В рассматриваемой СБК обнаружение ошибок происходило:
- посредством всевозможных внутренних проверок отладочной версии СБК — с помощью чекеров и ловушек самой системы;
- с помощью системы сравнения оптимизированного и неоптимизированного кода СБК [8].

При тестировании кодов, полученных с помощью описанного генератора, обнаружено более 25 ошибок на уже отлаженной версии СБК, успешно работающей с различными ОС и пользовательскими приложениями. Среди найденных ошибок — нереализованные, редко используемые и недопустимые инструкции СК IA-32, ошибки в различных компонентах СБК, в частности, в наборщике и компиляторе регионов, ошибки в шаблонном коде и в интерпретаторе. С помощью таких тестов удалось обнаружить несколько ошибок и в средствах отладки; была также выявлена аппаратная проблема, приводящая к остановке ВК "Эльбрус-3М1".

Создаваемые тесты являются 32-битными пользовательскими приложениями и предназначены для запуска в ОС Linux.

Предполагается дальнейшее развитие описанной технологии с возможностью интегрирования ее в другие автоматизированные системы генерации тестов для верификации и тестирования СБК, отладочных средств и аппаратуры ВК в рамках проекта "Эльбрус" [1].

Список литературы

1. **Волконский В. Ю.** Оптимизирующие компиляторы для архитектур с явным параллелизмом команд и аппаратной поддержкой двоичной совместимости // Информационные технологии и вычислительные системы. 2004. № 3. С. 4—26.
2. **Intel** Corporation. IA-32 Intel Architecture Software Developer's Manual. Vol. 1, 2.
3. **Nadkarni A. S., Kenville T.** Transmeta Corporation. TiGeR, the Transmeta Instruction GEneratoR: A production based, pseudo random, instruction, x86 test generator. Transmeta Corporation, 2004.
4. **Arbetman Ya. et al.** Genesys-x86 An automatic test program generator for x86 microprocessors. IBM Haifa Research Lab, 1997.
5. **Максименков Д. А.** Генератор тестов для бинарного компилятора // Современные информационные технологии и ИТ-образование. М.: Макс-пресс, 2005. С. 445—452.
6. **Максименков Д. А.** Метод создания самопроверяющихся тестов для системы бинарной компиляции // Высокопроизводительные вычислительные системы и микропроцессоры. Вып. 9. М.: ИМВС РАН, 2006. С. 26—37.
7. **Baraz L., Devor T., Etsion O.** et al. IA-32 Execution Layer: a two-phase dynamic translator designed to support IA032 applications on Itanium-based system // Pr. of the 36th International Symposium on Microarchitecture (MICRO-36'03). 2003.
8. **Иванов А. А., Преснов Н. Ю.** Технология динамического сравнения трасс для отладки систем двоичной трансляции // Информационные технологии. 2005. № 2. С. 49—54.

УДК 621.3.049.77.001.2

И. И. Холод, канд. техн. наук, доц., **А. В. Кочетков,**
Государственный электротехнический университет "ЛЭТИ", Санкт-Петербург

Самовосстановление программных систем на основе автоматической классификации ошибок

Рассматривается проблема самовосстановления программных средств. Предлагается формальная модель, описывающая работу программного средства как последовательность событий. Данная модель позволяет применить известные методы классификации для автоматического определения класса возникающих ошибок и устранения их последствий за счет выполнения определенных для данного класса действий.

Введение

В настоящее время все большее место в жизни человека занимают компьютерные программы. Они сопровождают человеческую деятельность везде: в быту, на работе, в больницах, на отдыхе. В связи с этим вопросы надежности функционирования программных средств (ПС) являются актуальными и критичными. Особенно это важно в случаях, когда требуется обеспечивать работу системы в режиме 7 дней в неделю, 24 ч в сутки.

Под *надежностью* ПС понимают его способность безотказно выполнять требуемые функции при заданных условиях в течение заданного периода времени с достаточно большой вероятностью [1]. При этом под отказом в ПС понимают проявление в нем ошибки [2].

В технике используются следующие четыре подхода для обеспечения надежности [1]:

- предупреждение ошибок;
- обеспечение устойчивости к ошибкам;
- самообнаружение ошибок;
- самоисправление ошибок.

Для программных систем наиболее часто используется первый подход, в меньшей степени второй и крайне редко два последних, тесно связанных с понятием самовосстановления систем. Основной причиной этого являются существенные отличия ПС от аппаратных систем (например, отсутствие износа). В результате известные методы повышения надежности в технике, например, такие как резервирование, в программных системах малоэффективны и ведут к их усложнению, а следовательно, к повышению вероятности появления ошибок и снижению надежности.

В настоящее время восстановление ПС, находящихся в эксплуатации, осуществляется службой сопровождения. В результате цикл восстановления системы после сбоя будет выглядеть так, как это показано на рис. 1.

В процессе сопровождения ПС при возникновении ошибки выполняется ее анализ с целью определить тип ошибки и причину ее возникновения. В зависимости от результата анализа осуществляются действия по устранению последствий возникновения ошибки и восстановления работоспособности системы.

Существенным недостатком такой организации восстановления систем является длительное время, проходящее с момента возникновения отказа до восстановления работоспособности системы.

Опыт сопровождения ПС показывает, что большинство ошибок, возникающих в процессе работы, повторяются, а следовательно, и операции по восстановлению системы также будут одинаковыми. Основная задача эксперта сводится к правильной классификации ошибки на основании информации о предыстории работы ПС и среды выполнения. В зависимости от класса ошибки эксперт выполняет те или иные действия для восстановления системы.

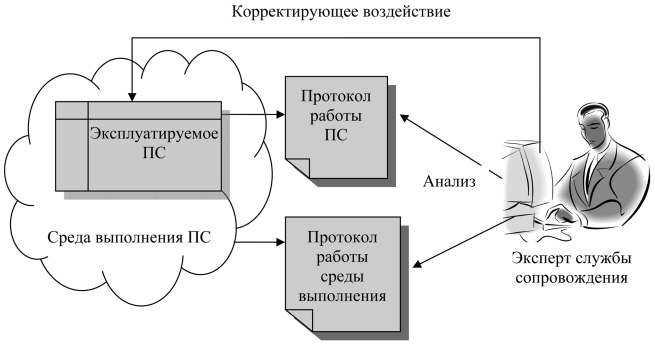


Рис. 1. Сопровождение ПС

Основываясь на этом факте, можно попытаться автоматизировать процесс восстановления ПС. Для этого необходимо разбить все ошибки по классам и для каждого из них определить операции по восстановлению системы. При этом основной задачей останется автоматическая классификация ошибок на основе информации о работе ПС. Для ее решения построим формальную модель, которая позволит применить известные методы классификации.

Виды событий в программных средствах

Любое программное средство работает в среде, включающей в себя аппаратную и программную составляющие, которые являются источниками событий, влияющих на работу ПС. Таким образом, среду выполнения ПС можно описать следующим образом:

$$M = \{H, S\},$$

где M — среда выполнения; H — аппаратная составляющая; S — программная составляющая.

Все множество событий, происходящих в процессе работы ПС, можно разделить на два подмножества:

$$E = E_{int} \cup E_{ext},$$

где E_{int} — внутренние, происходящие в самом ПС; E_{ext} — внешние, происходящие в программной и аппаратной составляющих.

Рассматривая ПС как "черный ящик" можно разделить все множество внутренних событий на два подмножества: входящие воздействия и реакции на них системы. Входящими воздействиями являются функции, которые вызывает пользователь. Таким образом, внутренние события

$$E_{iny} = E_{in} \cup E_{out},$$

где E_{in} — входящие события, функции, выполняемые ПС; E_{out} — исходящие события, реакции ПС.

Функции, которые может выполнять ПС (например, соответствующие функциональной спецификации на ПС) можно представить в виде множества:

$$F = \{f_1, f_2, \ldots, f_j, \ldots, f_m\},$$

где f_j — функция ПС.

Функции ПС могут быть параметризированы. В качестве параметров могут выступать данные, над которыми выполняется функция. Таким образом, множество выполненных ПС функций можно представить в виде отношения определенного на множестве функций F и множестве значений аргументов A_1, ..., A_r:

$$F' = \{(f_i,\ a_1^i\ ,\ a_2^i\ ,\ ...,\ a_k^i\ ,\ ...,\ a_r^i)|f_i \in F,\ a_1^i\ \in A_1,$$
$$a_2^i\ \in A_2,\ ...,\ a_k^i\ \in A_k,\ ...,\ a_r^i\ \in A_r\},$$

где f_i — функция ПС; a_k^i — значение k-го аргумента функции f_i из множества возможных значений A_k (в общем случае бесконечного).

Например, у функции f_i = "*запись файла*" в качестве единственного аргумента выступает записываемый файл a_1^i = "*readme.txt*", таким образом, выполняемая функция будет являться набором из двух элементов:

$$f_i = \{\text{"}\textit{запись файла}\text{"},\ \text{"}\textit{readme.txt}\text{"}\}.$$

На множестве выполненных функций F' можно задать два отношения эквивалентности:

- отношение эквивалентности по функции $E_f(f_i'\ f_j')$, имеющее предикат:

$$E_f(f_i'\ f_j') = \begin{cases} \text{истина, если } f_i = f_j; \\ \text{ложь, если } f_i \neq f_j; \end{cases}$$

- отношение эквивалентности по данным $E_d(f_i'\ f_j')$, имеющее предикат:

$$E_d(f_i'\ f_j') = \begin{cases} \text{истина, если } f_i = f_j \text{ и } a_k^i\ = a_k^j \\ \text{для всех } 1 \leqslant k \leqslant r; \\ \text{ложь, если } f_i \neq f_j \text{ или } a_k^i\ \neq a_k^j \\ \text{для любого } 1 \leqslant k \leqslant r. \end{cases}$$

Входящее событие, связанное с выполнением функции ПС в процессе его работы, дополнительно характеризуется временем и пользователем, выполняющим функцию. Таким образом, множество входящих событий E_{in} является упорядоченным по времени отношением, определенным на множестве функций ПС — F и множестве всех пользователей, имеющих доступ к ПС — $U = \{u_1, u_2, ..., u_i, ..., u_v\}$:

$$E_{in} = \{(f_i'\ ,\ u_i)|f_i'\ \in F,\ u_i \in U\},$$

где $f_i' \in F$ — функция, инициировавшая i-е событие; $u_i \in U$ — пользователь, при работе которого произошло i-е событие (обычно определяется при авторизированном входе в программу).

Для возможности сравнивать два события введем отношение эквивалентности входящих событий. Исходя из данного выше определения входящее событие можно характеризовать: функцией, аргументами функции, пользователем и временем выполнения. На основании этих характеристик получим разные отношения эквивалентности определенных на множестве E_{in}:

- отношение эквивалентность по функции $E_f((f_i'\ ,\ u_i),\ (f_j'\ ,\ u_j))$ имеет предикат

$$E_f((f_i'\ ,\ u_i),\ (f_j'\ ,\ u_j)) = E_f(f_i'\ ,\ f_j'\ );$$

- отношение эквивалентность по данным $E_d((f_i'\ ,\ u_i),\ (f_j'\ ,\ u_j))$ имеет предикат

$$E_d((f_i'\ ,\ u_i),\ (f_j'\ ,\ u_j)) = E_d(f_i'\ ,\ f_j'\ );$$

- отношение эквивалентности по пользователю $E_u((f_i'\ ,\ u_i),\ (f_j'\ ,\ u_j))$ имеет предикат

$$E_u((f_i'\ ,\ u_i),\ (f_j'\ ,\ u_j)) = \begin{cases} \text{истина, если } u_i = u_j; \\ \text{ложь, если } u_i \neq u_j; \end{cases}$$

- отношение полной эквивалентности $E((f_i'\ ,\ u_i),\ (f_j'\ ,\ u_j))$ имеет предикат

$$E((f_i'\ ,\ u_i),\ (f_j'\ ,\ u_j)) = \begin{cases} \text{истина, если } E_d(f_i'\ ,\ f_j'\ ) = \\ = \text{истина и } u_i = u_j; \\ \text{ложь, если } E_d(f_i'\ ,\ f_j'\ ) = \\ = \text{истина и } u_i \neq u_j. \end{cases}$$

Можно определить и другие отношения эквивалентности в зависимости от типа ПС. Например, для эквивалентности двух событий можно потребовать равное время от предшествующих событий. Такое условие может быть необходимо, например, для систем реального времени.

На функции, вызываемые пользователями, ПС реагирует, генерируя некоторые события. Примерами таких событий являются: открытие диалогов, установление соединений, изменение данных и т. п. Реакцией системы являются и ошибки, проявляющиеся в виде сообщений об ошибках или некорректном поведении системы (например, "зависании" — отсутствии каких-либо событий на длительном интервале времени). Таким образом, множество исходящих событий E_{out} является упорядоченным по времени отношением, определенным на множестве реакций ПС − R и множествах значений, характеризующих реакцию, — V_1, V_2, ..., V_h, V_z:

$$E_{out} = \{(r_i,\ v_1^i\ ,\ v_2^i\ ,\ ...,\ v_h^i\ ,\ ...,\ v_z^i\ )\ |\ r_i \in R,\ v_1^i\ \in V_1,$$
$$v_2^i\ \in V_2,\ ...,\ v_h^i\ \in V_h,\ ...,\ v_z^i\ \in V_z\},$$

где r_i — i-я реакция системы; v_h^i — значение h-го аргумента, определяющего реакцию r_i.

Для упрощения обозначим набор $(r_i,\ v_1^i\ ,\ v_2^i\ ,\ ...,\ v_h^i\ ,\ ...,\ v_z^i\ )$ как r_i'.

По аналогии с входящими событиями определим отношения эквивалентности на множестве исходящих событий E_{out}:

- отношение эквивалентности по реакции $E_u(r_i', r_j')$ имеет предикат

$$E_u(r_i', r_j') = \begin{cases} \text{истина, если } r_i = r_j; \\ \text{ложь, если } r_i \neq r_j; \end{cases}$$

- отношение полной эквивалентности $E(r_i', r_j')$ имеет предикат

$$E(r_i', r_j') = \begin{cases} \text{истина, если } r_i = r_j \text{ и } v_h^i = v_h^j \\ \text{для всех } 1 \leqslant h \leqslant z; \\ \text{ложь, если } r_i \neq r_j \text{ или } v_h^i \neq v_h^j \\ \text{для любого } 1 \leqslant h \leqslant z. \end{cases}$$

Ошибки являются реакцией системы и относятся к исходящим событиям.

Внешние события зависят от программных и аппаратных средств, составляющих среду выполнения ПС. К сожалению, не всегда имеется возможность получить подробную информацию об их работе. Во многом это связано с независимостью их разработки. Несмотря на это, как правило, существует возможность получения информации об основных внешних событиях, например, запуск/остановка программ, изменение объема доступной памяти и т. п. Источником информации об этих событиях, как правило, может служить протокол работы ПС и операционной системы (ОС). Например, в ОС Windows таким источником информации является Event Viewer,

Информация, доступная о внешних событиях, во многом зависит от их источника. Кроме источника, внешнее событие характеризуется также типом и аргументами. Таким образом, множество внешних событий E_{out} является упорядоченным по времени отношением, определенным на множестве источников событий — S, множестве событий — K и множествах значений, характеризующих события — $C_1, C_2, ..., C_q, ..., C_w$:

$$E_{ext} = \{(s_i, k_i, c_1^i, c_2^i, ..., c_q^i, ..., c_w^i) | s_i \in S, k_i \in K,$$
$$c_1^i \in C_1, c_2^i \in C_2, ..., c_q^i \in C_q, ..., c_w^i \in C_w\},$$

где s — источник i-го события; k_i — вид i-го события; c_q^i — q-я характеристика i-го события.

Для упрощения обозначим набор $(s_i, k_i, c_1^i, c_2^i, ..., c_q^i, ..., c_w^i)$ как k_i'.

По аналогии с внутренними событиями определим отношения эквивалентности для внешних событий:

- отношение эквивалентности по типу события $E_k(k_i', k_j')$ имеет предикат

$$E_k(k_i', k_j') = \begin{cases} \text{истина, если } k_i = k_j; \\ \text{ложь, если } k_i \neq k_j; \end{cases}$$

- отношение полной эквивалентности $E(k_i', k_j')$ имеет предикат

$$E(k_i', k_j') = \begin{cases} \text{истина, если } s_i = s_j \text{ и } k_i = k_j, \\ \text{и } c_q^i = c_q^j \text{ для всех } 1 \leqslant q \leqslant w; \\ \text{ложь, если } s_i \neq s_j \text{ или } k_i \neq k_j, \\ \text{или } c_q^i \neq c_q^j \text{ для любого } 1 \leqslant q \leqslant w. \end{cases}$$

Для выполнения анализа работы ПС представим процесс его выполнения в виде последовательности (в общем случае бесконечной) событий, происходящих в хронологическом порядке:

$$P = \{e_0, e_1, e_2, ..., e_i, ..., e_z, ...\},$$

где $e_i \in E$ — i-е событие, произошедшее при выполнении ПС (все виды событий, описанных выше, обозначим e_i). Начальное событие e_0 соответствует запуску ПС.

Интервалы работы программного средства

Процесс работы ПС можно представить временным рядом (рис. 2).

Конечную последовательность k событий, следующих друг за другом, будем называть интервалом работы длиной k и обозначим его следующим образом:

$$p_i = \{e_i, e_{i+1}, e_{i+2}, ..., e_{i+k-1}\}.$$

Результатом работы ПС за обозначенный интервал будем считать событие, следующее за последним событием интервала: $R(p_i) = e_{i+k}$. Будем говорить, что событие e_{i+k} порождается интервалом p_i, если $R(p_i) = e_{i+k}$.

Множество всевозможных интервалов работы ПС длиной k обозначим P_k^*. Для построения такого множества предлагается использовать метод "скользящего окна", применяемый в анализе временных рядов [3]. При этом процесс выполнения ПС рассматривается как временной ряд, а события — как элементы ряда. Для формирования множества P_k^* весь временной ряд "нарезается" на множество "окон" длиной k. Для получения очередного интервала "окно" сдвигается на один эле-

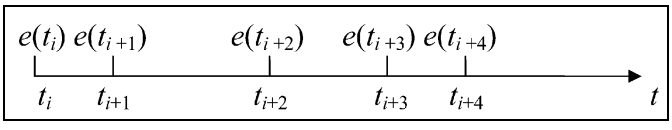


Рис. 2. Процесс работы ПС во времени

Разбиение работы ПС на интервалы

Интервал	События интервала работы ПС						Результат	Класс
p_0	e_0	e_1	...	...	e_{k-2}	e_{k-1}	e_k	C_1
p_1	e_1	e_2	...	...	e_{k-1}	e_k	e_{1+k}	C_4
...	...	...	...	...	...	...	...	...
p_i	e_i	e_{i+1}	...	...	e_{i+k-2}	e_{i+k-1}	e_{i+k}	C_b

мент вправо. Сформированные таким образом интервалы включаются в множество P_k^*, которое можно представить в виде таблицы.

Для классификации ошибок по определенным классам эксперт должен явно отнести каждый из элементов колонки "Результат" к одному из классов ошибок. Для построения классифицирующих правил с использованием известных методов классификации [3] необходимо определить понятие эквивалентности интервалов.

Введем отношение эквивалентности на множестве P_k^* — $E(p_r, p_h)$, имеющее следующий предикат:

$E_f(p_r, p_h) = E_f(e_{r+m}, e_{h+m})$ для любого $0 \leqslant m \leqslant k$.

Данное определение основано на отношении эквивалентности событий, которое по-разному определяется для внешних и внутренних, входящих и исходящих событий. Кроме того, выше для каждого вида событий определены несколько отношений эквивалентности. В связи с этим для интервалов также можно ввести следующие типы отношения эквивалентности:

- отношение эквивалентности по функциям входящих событий

$E_f(p_r, p_h) = E_f(e_{r+m}, e_{h+m})$ для всех $e_{r+m} \in E_{in}$, $e_{h+m} \in E_{in}$ и $e_{r+m} \in p_r$, $e_{h+m} \in p_h$;

- отношение эквивалентности по данным входящих событий

$E_d(p_r, p_h) = E_d(e_{r+m}, e_{h+m})$ для всех $e_{r+m} \in E_{in}$, $e_{h+m} \in E_{in}$ и $e_{r+m} \in p_r$, $e_{h+m} \in p_h$;

- отношение эквивалентности по пользователям входящих событий

$E_u(p_r, p_h) = E_u(e_{r+m}, e_{h+m})$ для всех $e_{r+m} \in E_{in}$, $e_{h+m} \in E_{in}$ и $e_{r+m} \in p_r$, $e_{h+m} \in p_h$;

- отношение полной эквивалентности по входящим событиям

$E_{in}(p_r, p_h) = E_u(e_{r+m}, e_{h+m})$ для всех $e_{r+m} \in E_{in}$, $e_{h+m} \in E_{in}$ и $e_{r+m} \in p_r$, $e_{h+m} \in p_h$;

- отношение эквивалентности по реакции исходящих событий

$E_r(p_r, p_h) = E_r(e_{r+m}, e_{h+m})$ для всех $e_{r+m} \in E_{in}$, $e_{h+m} \in E_{in}$ и $e_{r+m} \in p_r$, $e_{h+m} \in p_h$;

- отношение полной эквивалентности по исходящим событиям

$E_{out}(p_r, p_h) = E(e_{r+m}, e_{h+m})$ для всех $e_{r+m} \in E_{in}$, $e_{h+m} \in E_{in}$ и $e_{r+m} \in p_r$, $e_{h+m} \in p_h$;

- отношение эквивалентности по типу внешних событий

$E_k(p_r, p_h) = E_k(e_{r+m}, e_{h+m})$ для всех $e_{r+m} \in E_{in}$, $e_{h+m} \in E_{in}$ и $e_{r+m} \in p_r$, $e_{h+m} \in p_h$;

- отношение полной эквивалентности по внешним событиям

$E_{ext}(p_r, p_h) = E(e_{r+m}, e_{h+m})$ для всех $e_{r+m} \in E_{in}$, $e_{h+m} \in E_{in}$ и $e_{r+m} \in p_r$, $e_{h+m} \in p_h$.

Построение классификационных правил

Для формирования правил классификации ошибок сгруппируем эквивалентные интервалы. В результате получим следующее множество групп:

$$A = (A(p_r) | (A(p_r) = \{p_h | E(p_h, p_r)\} \text{è} A(p_r) \in P_k^*),$$

где $A(p_r)$ — множество интервалов длиной k, эквивалентных интервалу p_r.

В зависимости от используемого для группирования отношения эквивалентности интервалов можно получить различные группировки интервалов.

На основании определенных множеств групп можно найти бинарное отношение между множеством групп эквивалентных интервалов длиной k и классами событий (см. таблицу), которые они порождают:

$$G = \{(A(p_i),\ C_j) | p_i \in P_k^*\ \text{è}\ R(p_i) = C_j\}.$$

Левым сечением бинарного отношения G по группе эквивалентных интервалов $A(p_i)$ является подмножество событий разных классов, порождаемых интервалами, входящими в группу:

$$S_l(A(p_i),\ G) = \{e_j | (R(p_r) = e_w = C_j\ \text{è}\ p_r \in A(p_i))\}.$$

Правым сечением бинарного отношения G по событию класса C_j является подмножество эквивалентных интервалов работы ПС длиной k, порождающих события класса C_j:

$$S_r(G,\ C_j) = \{A(p_r) | (R(p_r) = e_{r+k} = C_j)\}.$$

Основываясь на этом, можно построить классификационные правила, в правой части которых

выполняется сравнение интервалов — представителей групп, а в левой части определяется класс события. Например:

$$\text{если } E(p_r, p_h) \text{ то } e_{h+k} \in C_j,$$

где p_r — интервал — представитель группы $A(p_r)$; p_h — интервал, порождающий классифицируемое событие; e_{h+k} — классифицируемое событие; C_j — класс, к которому относим классифицируемое событие.

Построенные правила имеют смысл только в случае, если выполняются следующие условия:

- число эквивалентных интервалов, порождающих события одного класса, близко к числу интервалов в данной группе:

$$|S_r(G, C_j)| \to |A(p_i)|;$$

- число событий, относящихся к разным классам, порождаемых эквивалентными интервалами, близко к единице

$$|S_l(A(p_i), G)| \to 1.$$

Построенные правила могут использоваться не только для классификации уже возникших ошибок, но и для прогнозирования появления новых.

Для количественного измерения эквивалентности интервалов можно ввести меру, определяемую, например, как число эквивалентных событий в двух интервалах. В этом случае можно говорить о степени принадлежности порождаемого события к классу как о количественной величине.

Заключение

Описанная модель позволяет, используя известные методы классификации, строить классификационные правила на основании существующей истории работы ПС. Они, в свою очередь, в дальнейшем могут быть использованы для автоматической классификации ошибок, возникающих в процессе работы ПС. Это позволяет предпринимать меры по устранению последствий без вмешательства человека, что значительно ускоряет процесс самовосстановления ПС, а также позволяет использовать данный подход в случаях, когда эксперту невозможно обеспечить доступ к ПС по соображениям безопасности или в случае отсутствия физического доступа к ПС. Работа самовосстанавливающейся системы, использующей описанный подход, будет выглядеть так, как это показано на рис. 3.

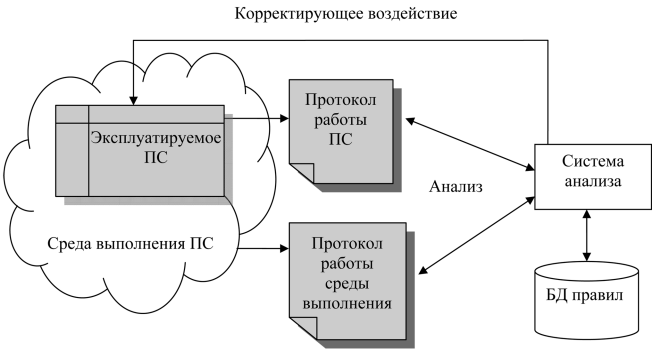


Рис. 3. Самовосстанавливающаяся ПС

Возможность прогнозировать появление ошибок на основе построенных классификационных правил позволяет предпринимать профилактические действия для предотвращения появления ошибок. Такого рода профилактика может быть оправдана из-за меньших затрат на предварительные операции по сравнению с устранением последствий возникновения ошибок.

Список литературы

1. **Criteria** for Evaluation of Software. ISO TC97/SC7#383.

2. **Майерс Г.** Надежность программного обеспечения. М.: Мир, 1980.

3. **Барсегян А. А., Куприянов М. С., Степаненко В. В., Холод И. И.** Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP. 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2007. 384 с.

УДК 621.391(07),004.04.

Ю. И. Сенкевич, канд. техн. наук,
Санкт-Петербургский институт информатики и автоматизации РАН

Метод лингвистического анализа сигналов

На основании положений математической лингвистики разработан метод анализа сигналов, включающий оригинальный алгоритм преобразования физических сигналов в условный символьный код, выделение совокупности параметров в виде грамматических форм и синтаксических правил, построение критериев, обнаруживающих изменчивость параметров, косвенно связываемую с событиями, происходящими в системе, генерирующей анализируемый сигнал. Представлен пример применения метода для оценки функционального состояния групп людей в процессе их адаптации к изменяющимся условиям среды обитания.

Введение

В ходе выполнения измерений система обнаруживает себя материально через изменение некоторого физического поля, воспринимаемого сенсорами как сигнал, косвенно отражающий определенную сторону динамики событий, происходящих в системе. С позиции математической лингвистики проявление в сигнале этих правил и закономерностей рассматривается как грамматика языка, на котором генерируется сообщение [1, 2]. В этом случае исследовательская цель обработки сигнала сводится к обнаружению системы признаков и выработке критериев, позволяющих на следующем этапе анализа выделить грамматику языка сообщения. Зная грамматику языка и анализируя изменение совокупности выделенных признаков во времени, можно косвенно судить о событиях, происходящих в системе. Назовем такую последовательность действий лингвистическим анализом сигнала и представим его в виде схемы (рис. 1).

Описание метода

Первый этап анализа состоит в преобразовании физического сигнала в символьное сообщение. Для этого в сигнале измеряются максимальные и минимальные значения функции его огибающей — экстремумы и интервалы между ними. В дальнейшей обработке используются только эти показатели, сохраняющие информацию об амплитудно-фазовых характеристиках исходного сигнала [3]. Представим измеренную последовательность показателей сигнала в виде упорядоченного множества $N-1$ пар чисел: $x_0,\tau_0\ \ x_1,\tau_1 \ldots x_{N-2},\tau_{N-2}$, где x_i — амплитудное значение i-го экстремума сигнала; τ_i — интервал между i-м и $(i+1)$-м экстремумами. Проведем сравнение значений амплитуды и интервала i-го экстремума с последовательностью соответствующих значений для каждой из M пар справа по следующим правилам:

$$r_{i,i+m} = \begin{cases} 1, & x_i > x_{i+m} \\ 0, & x_i \leqslant x_{i+m}; \end{cases}$$
$$\omega_{i,i+m} = \begin{cases} 1, & \tau_i > \tau_{i+m} \\ 0, & \tau_i \leqslant \tau_{i+m}, \end{cases} \qquad (1)$$
$$m = 1 \ldots M \ (M \leqslant N - i),$$

где $r_{i,i+m}$ — результат логического сравнения i-го и $(i+m)$-го значений амплитуд экстремумов; $\omega_{i,i+m}$ — результат логического сравнения i-го и $(i+m)$-го значений интервалов экстремумов.

Упорядочим результаты в виде бинарных матриц для сравнения амплитудных значений экстремумов:

$$\mathbf{R}_i = \begin{pmatrix} r_{i,i} & r_{i,i+1} & \cdots & r_{i,i+(M-2)} & r_{i,i+(M-1)} \\ r_{i+1,i} & r_{i+1,i+1} & & & r_{i+1,i+(M-1)} \\ \vdots & & \ddots & & \\ r_{i+(M-2),i} & & & r_{i+(M-2),i+(M-2)} & r_{i+(M-2),i+(M-1)} \\ r_{i+(M-1),i} & r_{i+(M-2),i+1} & \cdots & r_{i+(M-1),i+(M-2)} & r_{i+(M-1),i+(M-1)} \end{pmatrix} \qquad (2)$$

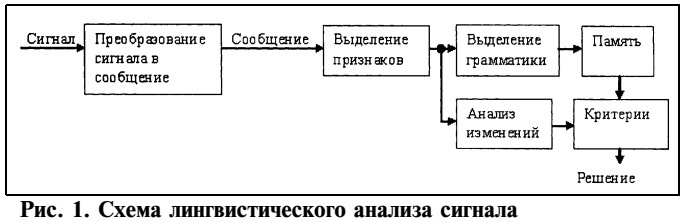


Рис. 1. Схема лингвистического анализа сигнала

и для сравнения значений интервалов между экстремумами вида

$$\mathbf{W}_i = \begin{pmatrix} \omega_{i,i} & \omega_{i,i+1} & \cdots & \omega_{i,i+(M-2)} & \omega_{i,(M-2)} \\ \omega_{i+1,i} & \omega_{i+1,i+1} & & & \omega_{i+1,i+(M-2)} \\ \vdots & & \ddots & & \\ \omega_{i+(M-2),i} & & & \omega_{i+(M-2),i+(M-2)} & \omega_{i+(M-2),i+(M-1)} \\ \omega_{i+(M-1),i} & \omega_{i+(M-2),i+1} & \cdots & \omega_{i+(M-1),i+(M-2)} & \omega_{i+(M-1),i+(M-1)} \end{pmatrix}. \quad (3)$$

Матрицы (2) и (3) — симметричные. Поэтому без потери информации о характеристиках сигнала можно выполнить преобразование матрицы (2) в нижнетреугольную матрицу $\mathbf{R}_i'$, а матрицы (3) — в верхнетреугольную матрицу $\mathbf{W}_i'$. На основании полученных матриц составим единую матрицу, объединяющую в себе отношения амплитуд и интервалов. Для этого выполним операцию алгебраического сложения матриц $\mathbf{R}_i'$ и $\mathbf{W}_i'$. Учитывая, что в соответствии с правилом (1) показатели на главной диагонали матриц (2) и (3) равны нулю, получим

$$\mathbf{D}_i = \mathbf{R}_i' + \mathbf{W}_i' = \begin{pmatrix} 0 & \omega_{i,i+1} & \cdots & \omega_{i,i+(M-3)} & \omega_{i,(M-2)} \\ r_{i+1,i} & 0 & & & \omega_{i+1,i+(M-2)} \\ \vdots & & \ddots & & \\ r_{i+(M-3),i} & & & 0 & \omega_{i+(M-3),i+(M-2)} \\ r_{i+(M-2),i} & r_{i+(M-2),i+1} & \cdots & r_{i+(M-2),i+(M-3)} & 0 \end{pmatrix}. \quad (4)$$

Матрица (4) представляет собой определенный код выбранного (i-го) экстремума в сигнале, характеризующий его амплитудное и временное положение по отношению к соседним экстремумам на глубину M вправо. Как следствие применения правил отношения (1), преобразование фрагмента сигнала из M экстремумов в матрицу (4) имеет свойство инвариантности к операциям амплитудного и временного транспонирования исходного сигнала. Полученное свойство вытекает из основного свойства неравенств: если $a > b$, то $a + c > b + c$ — при любом c, — для операции сжатия и растяжения сигнала во времени, и если $a > b$ и $c > 0$, то $ac > bc$ — для операции усиления сигнала. Поэтому каждой полученной матрице (4) можно составить графический инвариант формы сигнала.

К сигналу применяется следующее правило: если для некоторых выбранных экстремумов сигнала соответствующие им матрицы вида (4) одного порядка M совпадают, то принимается гипотеза о том, что такая матрица описывает устойчивый инвариант формы в сигнале, а саму матрицу можно интерпретировать как определенный знак — *символ*. Можно применить более жесткое ограничение, задаваясь бóльшим числом совпадений матриц для различных экстремумов — K. Назовем значение K статистическим порогом существования символа в сигнале, ниже которого все встречающиеся комбинации следует считать незначащими или нуль-символами (Z). Применяя правило выделения символов, сигнал можно преобразовать в некоторый символьный код — *сообщение* (рис. 2). Для символа величину, равную размерности M соответствующей ей матрицы, будем называть размерностью символа.

На рис. 2 (фрагмент I) можно видеть, как выделенные экстремальные значения сигнала преобразуются в бинарную матрицу (фрагмент IV). Затем с использованием вышеприведенного правила выделяются два символа, которым присвоены условные имена a и b с размерностями 5 и 7 соответственно. Ниже представлены инварианты символов (шаблоны формы), упорядоченные по амплитуде экстремумов (фрагмент II) и по значению интервала между экстремумами (фрагмент III).

На втором этапе лингвистического анализа из сообщения выделяются признаки. В соответствии с выбранным подходом это будут лингвистические формы. Назовем *алфавитом* A сообщения множество выделенных из сигнала символов. Назовем размером *алфавита* $D = |A|$ общее число обнаруженных символов алфавита в сообщении. Для примера, алфавит сигнального сообщения, представленный на рис. 2, есть $A = \{a, b\}$, $D = 2$. Сообщение в выделенных символах алфавита A выглядит как "abba".

Частота появления того или иного символа в сообщении различна и указывает на предпочтительные события, которые наиболее часто повторяются в системе, генерирующей сигнал. Для оценки частоты появления символа в сообщении удобно использовать *вероятность появления символа a_i алфавита A*, равную отношению числа появлений символа m_i в сообщении к общему числу символов в этом сообщении D:

$$P(a_i) = \frac{m_i}{D}. \quad (5)$$

Параметр (5) с точки зрения теории информации является еще и оценкой информационной нагрузки, приходящейся на каждый символ алфавита.

Если принять, что в некотором устойчивом состоянии набор событий, характеризующих состояние системы, не изменяется на протяжении длительного времени, то логично предположить, что состав алфавита A воспринимаемого сообщения, отражающий этот набор событий, и его размерность N, также не должны изменяться.

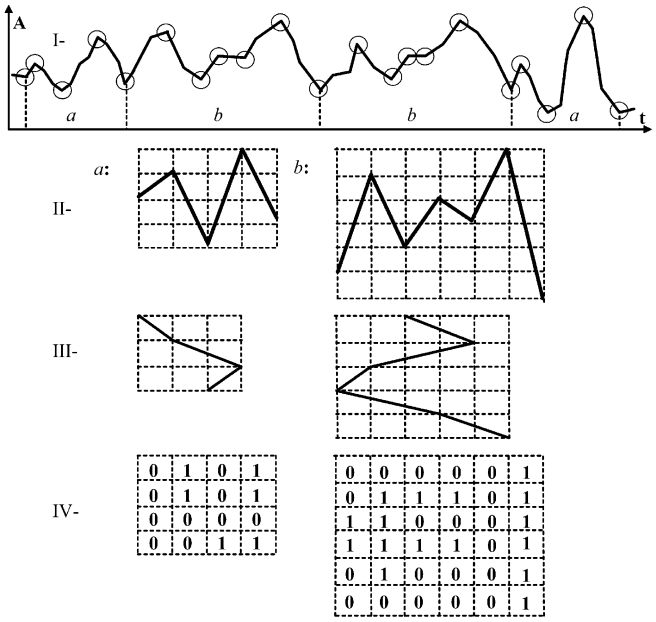


Рис. 2. Пример преобразования фрагмента сигнала

Меняться может только порядок появления каждого из символов сообщения или порядок последовательностей символов, косвенно отражающих некоторые события в системе. Иначе говоря, выдвигается гипотеза, что в рамках каждого устойчивого состояния система способна генерировать только фиксированный набор символов алфавита определенной размерности. Если произойдет изменение состояния системы, это должно повлечь за собой изменение набора событий, что, в свою очередь, должно вызвать качественные и количественные изменения в составе алфавита, которые можно обнаружить, используя элементы теории распознавания образов. Следуя рекомендациям теории распознавания образов [4], при разработке аппарата анализа необходимо предусмотреть возможность обучения, основанного на накоплении эталонов — некоторых фиксированных наборов параметров для последующих сравнений с текущим набором параметров. В рассматриваемом случае это будут наборы параметров, состоящие из алфавитов, информации о вероятностях появления символов различной размерности, которые будут храниться в блоке памяти схемы лингвистического анализа сигналов (см. рис. 1).

Для оценки связи двух алфавитов в анализе сообщения используется *коэффициент перекрытия алфавитов* k, представляющий собой отношение числа символов, которые совпадают для сравниваемых алфавитов, к суммарному числу символов в этих алфавитах. Пусть A и B — алфавиты двух сообщений с длинами N и M соответственно. Тогда из теории множеств мощность пересечения A и B будет представлять число элементов множества, полученного от их пересечения, а коэффициент перекрытия алфавитов A и B выразится как

$$k = 2\,|A \cap B|/(N + M). \qquad (6)$$

Следующий этап анализа направлен на поиск связей совокупностей символов алфавита, т. е. слов. *Слово* — устойчивая последовательность символов алфавита. Для поиска слов и выполняется алгоритм, включающий перебор возможных сочетаний символов выделенного алфавита по следующему правилу.

Если a_i и a_{i+1} — пара рядом стоящих символов сообщения встречается в этом же сообщении более чем $L \geqslant 2$ раз, то эта пара представляет устойчивую последовательность символов w. Назовем значение L статистическим порогом существования связи символов. В свою очередь, возможно, что сочетание w и a_{i+2} также окажется устойчивым. Проверка будет продолжаться до тех пор, пока статистика связи не станет ниже порога L. В последнем случае принимается решение, что последняя из обнаруженных связанная цепочка символов w есть более высокая грамматическая форма — слово.

Применяя правило поиска слов ко всему сообщению, выделяется весь набор слов $W = \{w\}$, составляющий *словарь сообщения* объемом $V = |W|$. На рис. 3 представлен пример обработки гипотетического текстового сообщения протяженностью 67 символов, составленного из алфавита в 15 символов, с выделенным словарем объемом в 6 слов.

Весьма вероятна ситуация, когда сообщение не будет полностью покрываться словами, и появятся непокрытые фрагменты сообщения. Это явление на синтаксическом уровне является указанием на недостаточность длины выборки текста сообщения для его корректного анализа, аналогично тому, как в спектральном анализе сигналов делается вывод о недостаточном времени накопления сигнала для его измерения [5].

Апробация метода

Среди различных применений представленного анализа наиболее интересные результаты исследований были получены в области медицинской диагностики. В настоящее время приведенные выше алгоритмы и правила анализа сигналов реализованы в виде компьютерной программы [6]. Программа использовалась для обработки результатов медицинского мониторинга процесса адаптации полярников в ходе ряда Российских антарктических экспедиций. Исследование влияния на адаптивные процессы организма метеорологических факторов проводились на выборке добровольцев из семи человек. Фиксировались элек-

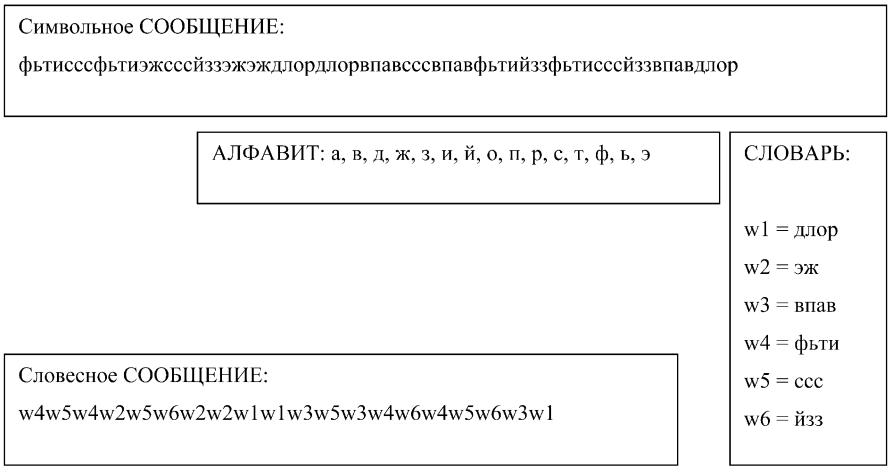


Рис. 3. Пример обработки сообщения

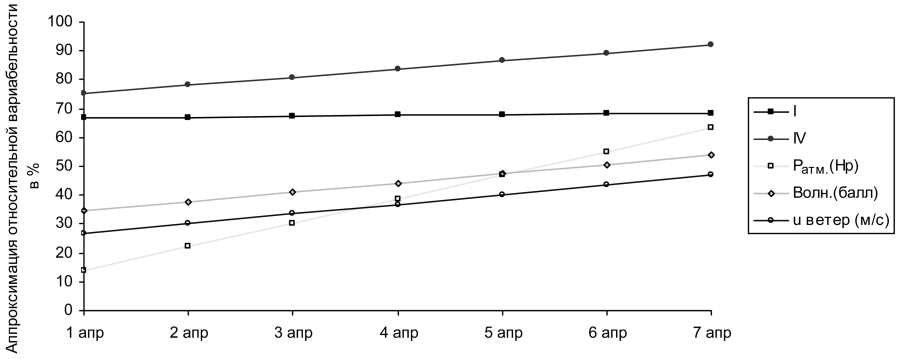


Рис. 4. Результаты расчета нормированных значений аппроксимации вариабельности значений размерности алфавитов электрофизиологических сигналов и тенденций атмосферного давления, скорости ветра и волнения моря на этапе "Адаптация 2" для участников 1-й подгруппы

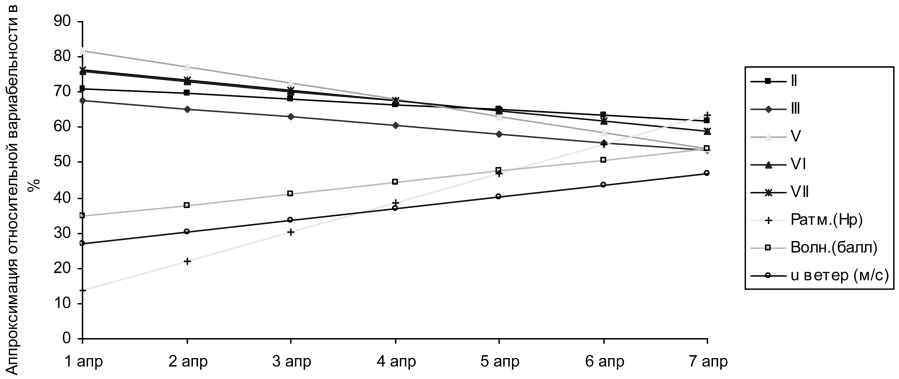


Рис. 5. Результаты расчета нормированных значений аппроксимации вариабельности значений размерности алфавитов электрофизиологических сигналов и тенденций атмосферного давления, скорости ветра и волнения моря на этапе "Адаптация 2" для участников 2-й подгруппы

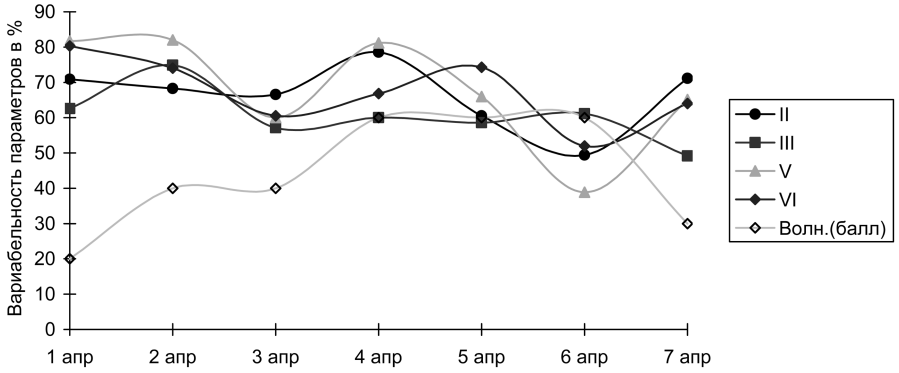


Рис. 6. Результаты расчета нормированных значений размерностей алфавитов для участников эксперимента 2-й подгруппы и волнения океана на этапе "Адаптация 2"

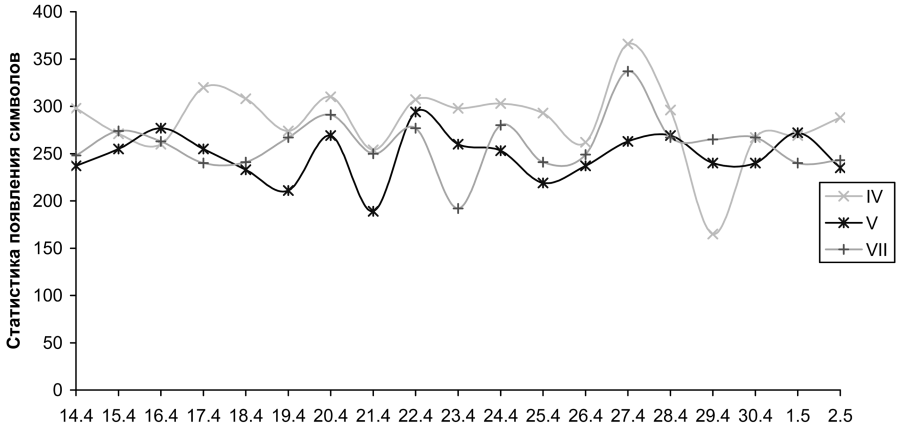


Рис. 7. Результаты расчета интегральной статистики появления символов алфавитов для трех наблюдаемых участников смешанной группы на этапе "Адаптация 3". Смена корабельного времени 18/04/04 на один час вперед и 25/04/04 на два часа вперед

трокардиограммы и значения показателей атмосферного давления, скорости ветра и волнения моря каждые сутки в одно и то же корабельное время. Полученные результаты обработки сравнивались по критерию (6) как между отдельными участниками, так и между последовательностями измерений; разнесенных во времени для каждого участника эксперимента, с учетом сопоставления результатов с измерениями климатических показателей.

Полученные в ходе наблюдений за экспериментальной группой файлы электрофизиологических сигналов сводились в электронные таблицы по временным этапам измерений с названиями "Адаптация 1", "Адаптация 2", "Адаптация 3", каждый из которых имел ярко выраженную климатическую особенность.

В ходе первичного анализа данных обнаружилось, что группа наблюдаемых в каждом из периодов разделяется на две подгруппы (2 и 5 человек) по своей реакции на изменчивость метеоусловий (метеолабильность). В первую подгруппу (условные имена — I и IV) попали наблюдаемые с позитивной тенденцией размерности алфавита на увеличение атмосферного давления, скорости ветра и волнения моря (рис. 4). Эта группа людей, мало подверженных влиянию изменяющихся метеоусловий. Во вторую подгруппу (условные имена — II, III, V, VI, VII) попали наблюдаемые с негативной тенденцией размерности алфавита на увеличение атмосферного давления, скорости ветра и волнения моря (рис. 5). Эта группа объединяет людей с четко выраженной зависимостью от состояния погоды. Для наглядности все сравниваемые результаты нормированы и даны в шкале линейной аппроксимации вариабельности представляемых показателей.

Показать обнаруженную функциональную связь реакции организма с изменчивостью метеоданных на примере измерений штормовой активности с использованием рассматриваемого метода можно с помощью рис. 6, на котором представлены результаты расчета нормированных значений размерностей полученных алфавитов для четырех наблюдаемых участников эксперимента и волнения океана.

Интересным результатом наблюдений стала связь интегральной статистики появления символов, рассчитанная для трех участников группы (рис. 7), проявившаяся при смене корабельного времени. На графике явно прослеживается сбой циркадных ритмов, обнаруживаемый статистическим показателем алфавитов электрофизиологических сигналов.

Полученные результаты исследований показывают, что лингвистический анализ может выступать чувствительным инструментом при классификации метеола-

бильных групп и оценке функционального состояния организма.

Оценка полученных результатов

Представленные выше этапы составляют метод обработки сигналов, формирующий пространство признаков, которое может быть использовано для поиска семантики сообщений. Лингвистический подход к анализу сигналов открывает перспективу широкого использования известных методов обработки и анализа текстовой информации для обработки и анализа физических сигналов.

Список литературы

1. **Гладкий А. В.** Формальные грамматики и языки. М., 1973.
2. **Сергиевский Г. М.** Лингвистические модели. М.: Изд-во МИФИ, 1983.
3. **Финк Л. М.** Сигналы, помехи, ошибки... Заметки о некоторых неожиданностях, парадоксах и заблуждениях в теории связи. 2-е изд., перераб., и доп. М.: Радио и связь, 1984. 256 с.
4. **Александров В. В., Горский Н. Д.** Алгоритмы и программы структурного метода обработки данных. Л.: Наука, 1983. 208 с.
5. **Марпл-мл. С. Л.** Цифровой спектральный анализ и его приложения. М.: Мир, 1990. 584 с.
6. **Сенкевич Ю. И.** Программа энтропийно-синтаксического анализа электрофизиологических сигналов (ESAES ver. 3.0). М.: ВНТИЦ, 2007. № 50200700165.

УДК 004.89

Норенков И. П., д-р техн. наук, проф.,
МГТУ им. Н. Э. Баумана

Исследование эффективности генетического метода с фрагментным кроссовером

Описывается генетический метод, основанный на многоточечном кроссовере, многохромосомной рекомбинации и пофрагментных макромутациях. Дано объяснение большей эффективности метода по сравнению с алгоритмами однородного, одно- или двухточечного кроссовера.

Генетические алгоритмы (ГА) относятся к средствам решения широкого круга задач оптимизации и структурного синтеза, причем для некоторых задач структурного синтеза применение ГА практически является безальтернативным. Однако эффективность ГА, определяемая точностью приближенного решения задачи и затратами вычислительных ресурсов, не всегда оказывается удовлетворительной. Поэтому продолжают появляться работы, посвященные повышению эффективности ГА.

Одним из направлений повышения эффективности является применение многоточечного кроссовера. В этом направлении следует отметить разработку алгоритма с однородным кроссовером [1], исследование вероятности разрыва шаблонов разных порядков при разных n (n — число разрывов хромосомы при кроссовере) [2], исследование поисковых и смешивающих возможностей многоточечного кроссовера [3, 4], применение кроссовера с многими родительскими хромосомами [5] и др.

Одним из способов реализации многоточечного кроссовера является генетический метод с фрагментным кроссовером, названный в [6] смешанным эволюционным методом (ММЕМ — Mixed Mode Evolution Method). Повышение эффективности генетического поиска достигается в ММЕМ за счет выбора оптимальных размеров фрагментов, на которые разделяются хромосомы при кроссовере. ММЕМ характеризуется следующими двумя особенностями:

- кроссовер многоточечный;
- рекомбинация многохромосомная, т. е. хромосома каждого потомка формируется из хромосом более двух родителей.

Целью данной публикации является, во-первых, обоснование наличия оптимальных значений числа разрывов хромосом при кроссовере, во-вторых, описание дополнительных резервов повышения эффективности генетического поиска с помощью циклических макромутаций.

Фрагментами в ММЕМ называют участки, на которые делятся хромосомы при кроссовере. В пределах каждого фрагмента выполняются свои операции выбора родительской пары и одноточечного кроссовера, но оценка целевой функции осуществляются по отношению ко всей хромосоме. Фрагментацию хромосом иллюстрирует рис. 1.

Специфика ММЕМ отражена главным образом в блоке формирования хромосом для следующего поколения. Параметрами, характеризующими особенности метода, являются длина L фрагментов, на которые разделяются родительские хромосомы, число хромосом-родителей у каждой формируемой хромосомы.

Была исследована эффективность ММЕМ следующих четырех тестовых задач:

- a — синтеза расписаний;

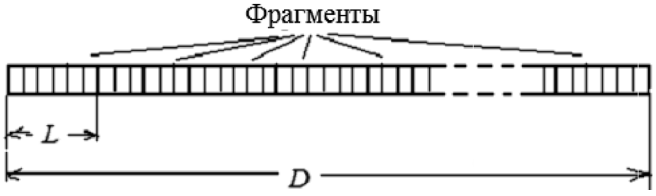


Рис. 1. Структура хромосомы в ММЕМ

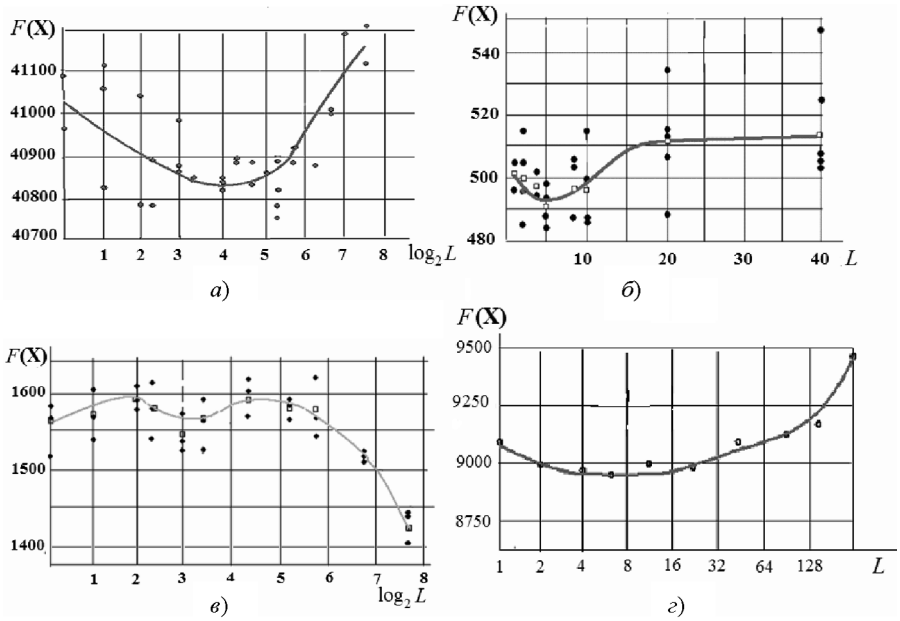


Рис. 2. Экспериментальные зависимости целевых функций от длины фрагментов L в задачах:
а — синтеза расписаний, *б* — маршрутизации транспортных средств с временными окнами, *в* — компоновки; *г* — синтеза топологии вычислительной сети

- *б* — маршрутизации транспортных средств с временными окнами;
- *в* — компоновки;
- *г* — синтеза топологии вычислительной сети.

На рис. 2 показаны результаты решения ряда тестовых задач с помощью ММЕМ при различных значениях длины фрагментов. Длины хромосом находились в диапазоне от 40 генов в задаче (*б*) до 800 в задаче синтеза расписаний (*а*). Поскольку при $L = D$ имеем генетический алгоритм (ГА) с одноточечным кроссовером, а при $L = 1$ — с однородным кроссовером (здесь L — длина фрагмента — число генов в фрагменте, D — длина хромосомы), результаты рис. 2 дают возможность сравнения точности решения задач при различных числах точек кроссовера (длин фрагментов). Во всех четырех задачах в случае ММЕМ степень приближения к экстремуму (в задачах *а*, *б*, *г* выполнялась минимизация, а в задаче *в* — максимизация целевой функции) была выше, чем при одноточечном, двухточечном или однородном кроссовере, при одинаковых вычислительных затратах.

По отношению к отдельному фрагменту выполняются условия, при которых Д. Холландом была сформулирована **теорема шаблонов** [7], ее выражение без учета мутаций и при вероятности кроссовера $p_c = 1$:

$$K(\mathbf{H}_{AB}, t+1) = \\ = K(\mathbf{H}_{AB}, t)(1 - d(\mathbf{H}_{AB})/L)F_{av}(\mathbf{H}_{AB})/F_0, \qquad (1)$$

где $K(\mathbf{H}_{AB}, t)$ — число шаблонов $\mathbf{H}_{AB}$ в поколении t; $d(\mathbf{H}_{AB})$ — длина шаблона $\mathbf{H}_{AB}$; $F_{av}(\mathbf{H}_{AB})$ — средняя полезность хромосом в поколении t, включающих в себя шаблон $\mathbf{H}_{AB}$; F_0 — средняя полезность всех хромосом в поколении t.

В формуле (1) учитываются наследование шаблонов и их возможное разрушение при кроссовере, но не учитываются процессы появления новых шаблонов. Для учета процессов как разрушения, так и генерации шаблонов необходимо дополнить выражение (1) слагаемым, учитывающим вероятность синтеза новых шаблонов.

Учет генерации новых шаблонов может быть выполнен следующим образом. Рассмотрим ситуацию разрыва хромосомы между j-м локусом с аллелем **A** и $(j + 1)$-м локусом с аллелем **B**. Обозначим вероятность появления **A** и **B** в родительских хромосомах t-го поколения соответственно через p_A и p_B, а вероятность появления **A** и **B** в составе одной и той же родительской хромосомы через p_{AB}. Число новых экземпляров шаблонов $\mathbf{H}_{AB}$ с аллелями **A** и **B**, появляющихся в хромосомах потомков в результате одного акта кроссовера, оценивается как $(p_A - p_{AB})(p_B - p_{AB})$, а вероятность сохранения схемы $\mathbf{H}_{AB}$ (компенсация разрушения) как $(p_A - p_{AB})p_{AB} + (p_B - p_{AB})p_{AB} + p_{AB}^2$. Суммируя эти вероятности, получаем

$$(p_A - p_{AB})(p_B - p_{AB}) + (p_A - p_{AB})p_{AB} + (p_B - \\ p_{AB})p_{AB} + p_{AB}^2 = p_A p_B.$$

Следовательно, в результате N актов кроссовера при $p_c = 1$ число шаблонов увеличивается по сравнению с результатом, вытекающим из теоремы шаблонов, на величину $\Delta K = Np_Ap_B/L$.

При этом вероятность p_A может быть определена как $K(A, t)F_{av}(A)/F_0$, аналогично определяются p_B и p_{AB}.

Очевидно, что полученный результат для двухпозиционных шаблонов может быть распространен и на многопозиционные шаблоны длиной d, если под **A** и **B** понимать шаблоны, находящиеся в пределах одного фрагмента по разные стороны от линии разрыва хромосом.

Из (1) и (2) непосредственно не следует рост эффективности генетического поиска с уменьшением L. Обоснованием наличия оптимальных, с точки зрения точности, значений длин фрагментов служат следующие соображения.

Рассмотрим участок хромосомы, инцидентный двум фрагментам и состоящий из левой части **A** и

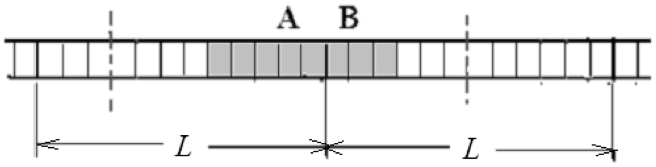


Рис. 3. Двухфрагментный участок хромосомы

правой части **B**, в условиях отсутствия мутаций и внутрифрагментного кроссовера (рис. 3).

Вероятность p_{AB} случайной генерации шаблона $\mathbf{H}_{AB}$, состоящего из левой части **A** и правой части **B** и инцидентного двум фрагментам, в исходном поколении при $N \ll s^u$ определяется выражением

$$p_{AB} = N/s^u,$$

где N — размер популяции; s — число символов в алфавите кодирования хромосом; u — порядок шаблона $\mathbf{H}_{AB}$. Вероятности генерации шаблонов **A** и **B** с порядками $u(\mathbf{A})$ и $u(\mathbf{B})$ соответственно оцениваются (при $N \ll s^{u(\mathbf{A})}$ и $N \ll s^{u(\mathbf{B})}$ как

$$p_A = N/s^{u(\mathbf{A})} \text{ и } p_B = N/s^{u(\mathbf{B})}.$$

Во многих практических задачах (синтез расписаний относится к их числу) плотные шаблоны оказывают большее влияние на функцию полезности, чем шаблоны с удаленными друг от друга локусами. Поэтому рассмотрение целесообразно провести применительно к плотному шаблону с $d(\mathbf{H}_{AB}) = u(\mathbf{H}_{AB})$.

Пусть шаблон $\mathbf{H}_{AB}$ является полезным (входит в состав оптимальной хромосомы). Существует некоторая длина шаблона $d_{пор}$, при которой появление $\mathbf{H}_{AB}$ в исходной популяции становится маловероятным, и эта вероятность есть $p_{пор}$. Например, при $p_{пор} = 0{,}02$, $N = 100$ и $s = 5$ (последнее может означать, что используются пять различных эвристик в мультиметодном алгоритме) имеем

$$d_{пор} = \lg (50N)/ \lg 5 > 5.$$

Появиться шаблон $\mathbf{H}_{AB}$ в текущей популяции при $d > d_{пор}$ может только вследствие его генерации (сохранение отсутствует). Генерация происходит с вероятностью рекомбинации частей шаблона **A** и **B**, равной $p_A p_B$. Следовательно, необходимо, чтобы вероятности p_A и p_B были больше $p_{пор}$, что, очевидно, выполняется в рассмотренном примере при $d_{пор} = 6$ или 7.

В ситуации, показанной на рис. 3, появившийся шаблон $\mathbf{H}_{AB}$ не может сохраниться и разрушается в следующем акте кроссовера. Однако высокая оценка его полезности ведет к увеличению вероятностей p_A и p_B, т. е. к усилению генерационных возможностей алгоритма. Для шаблонов меньшей полезности, чем полезность шаблона $\mathbf{H}_{AB}$, генерационные возможности составляющих компонентов усиливаются слабее или даже ослабевают. Таким образом, важно не столько сохранение полезных шаблонов в популяции, сколько усиление возможностей их генерации.

Увеличение числа позиций, в которых происходит генерация новых шаблонов, способствует появлению полезных шаблонов в большем числе частей хромосомы, что и объясняет улучшение целевой функции при уменьшении L при длинах фрагментов выше оптимальной длины $L_{опт}$. При приближении L к $L_{опт}$ начинает сказываться близость друг к другу соседних точек разрыва. Если расстояние L между этими точками становится соизмеримым или менее $d_{пор}$, интенсивность генерации полезных шаблонов падает, так как теперь шаблон формируется уже из трех участков **A**, **B**, **E**, принадлежащих каждый разным фрагментам, и вероятность его генерации определяется произведением трех вероятностей $p_A p_B p_E \ll 1$.

Характер генерации новых полезных шаблонов будет аналогичным рассмотренному, если учитывать наличие внутрифрагментного кроссовера (возможные линии разрыва фрагментов показаны на рис. 3 штриховыми линиями), при этом оптимальные длины фрагментов должны возрасти приблизительно вдвое.

В практических алгоритмах ММЕМ при использовании комбинирования эвристик оптимальные значения $L_{опт}$ находятся в диапазоне от 4 до 30. Поскольку априорно значение $L_{опт}$ неизвестно, целесообразно использовать алгоритм с автоматическим выбором L из некоторого набора, заданного в качестве исходных данных.

Генетический поиск может прекратиться вдали от искомого экстремума. Причиной этого могут быть вырождение популяции (стагнация) или застревание в локальном экстремуме.

Для предотвращения стагнации обычно используют макромутации и/или увеличение вероятности мутации. Однако по мере приближения к малым окрестностям локального экстремума (или при наступлении стагнации) обычные макромутации становятся малоэффективными. Это связано с тем, что полезность мутированной хромосомы оказывается значительно хуже, чем у других хромосом популяции, поэтому мутированная хромосома выпадает из числа возможных родителей и, следовательно, не может вывести популяцию из состояния стагнации. Если же попытаться уйти из этого состояния, выполнив мутации большинства генов у всех хромосом, то мутированные хромосомы смогут участвовать в генерации потомков, поскольку полезности всех членов популяции становятся сопоставимыми. Но при столь значительных мутациях теряется накопленный полезный генный материал и поэтому после макромутации поиск практически начинается вновь с исходных позиций.

Для того чтобы макромутации были эффективными, необходимо выполнение двух условий:

- сохранение накопленного генного материала;
- начальные полезности у всех хромосом после макромутаций должны быть соизмеримыми.

Совместить эти два противоречивых требования удалось в методе, названном *циклическим генетическим методом*. Циклический генетический метод создан в рамках генетического метода с фрагментным кроссовером. Его сущность поясняет рис. 4.

В соответствии с циклическим методом каждая хромосома текущего поколения может при макромутациях породить двух потомков. В хромосому первого потомка переписываются нечетные фрагменты, а гены четных фрагментов получают случайным образом сгенерированные аллели. В хромосоме второго потомка сохраняются фрагменты родителя с четными номерами, а нечетные фрагменты заполняются случайными значениями.

Макромутации повторяются периодически через W смен поколений, называемых циклами, где W — число смен поколений, после которых вероятно появление признаков стагнации. Поскольку за счет 50 % заполнения генов случайными значениями число хромосом после макромутаций увеличивается вдвое, в операции макромутации участвует и далее используется лишь половина хромосом предыдущего поколения. В эту половину целесообразно включать лучшие особи.

После каждой макромутации целевая функция резко ухудшается, но за счет сохранения полезных шаблонов в новом цикле наблюдается ее ускоренное улучшение, причем велика вероятность, что это улучшение приведет траекторию поиска в меньшую окрестность экстремума (или на более удачный уровень стагнации), чем это было в предыдущем цикле.

Для иллюстрации повышения эффективности циклического генетического метода (ЦГМ) были выполнены эксперименты с тестовой задачей синтеза расписаний со следующими исходными данными: число работ 105, число стадий 4, число машин на стадиях {4, 4, 4, 3}. На рис. 5 представлены результаты расчетов — зависимости целевой функции $F(X)$ от числа r поколения при применении простого генетического метода с фрагментным кроссовером (ММЕМ) и ЦГМ с макромутациями через 120 поколений. Как видно на рис. 5, на первых четырех циклах происходило улучшение целевой функции. Причем стагнация в случае применения ММЕМ наступила на уровне $F(X) = 21751$, а в случае ЦГМ при числе поколений $r = 240$, 360, 480 достигнуты значения $F(X)$, равные 21734, 21716 и 21676.

Следует отметить, что идея циклического пофрагментного обновления хромосом может оказаться полезной в многопопуляционных параллельных ГА для обмена генным материалом между популяциями.

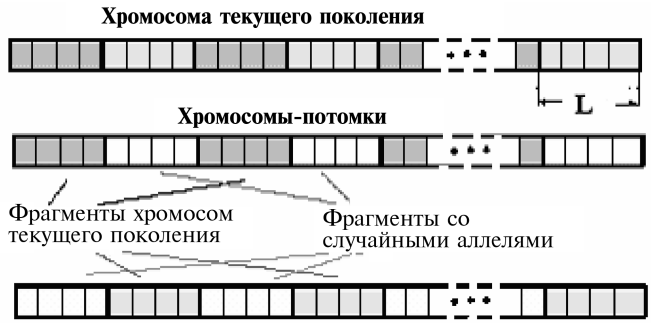


Рис. 4. Макромутации в циклическом генетическом методе

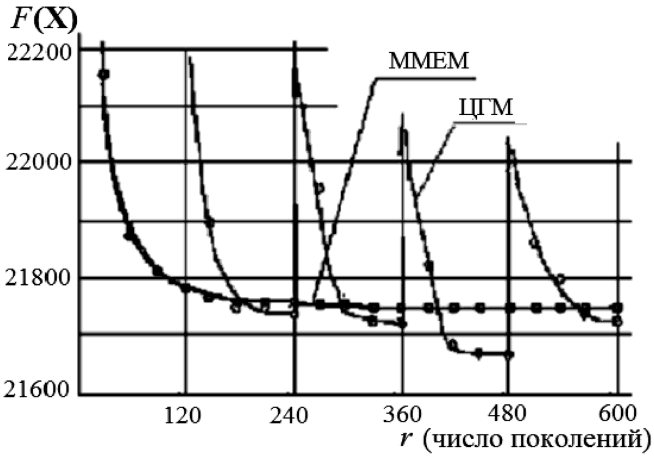


Рис. 5. Результаты применения ММЕМ и ЦГМ

Таким образом, разработан смешанный генетический метод, основанный на разделении хромосом на фрагменты и выполнении операторов выбора родителей и кроссовера отдельно для каждого фрагмента. Показано существование оптимального значения длины фрагментов, объясняемое изменением роли процессов генерации и разрушения шаблонов при изменении их длины. Выполнение циклических (пофрагментных) макромутаций обусловливает дополнительные возможности преодоления ранней стагнации и "застревания" поиска в локальных экстремумах.

Список литературы

1. **Syswerda G.** Uniform Crossover in Genetic Algorithms // Proc. 3rd Int. Conf. on Genetic Algorithms. Morgan Kaufmann Publ., 1989.
2. **De Jong K. A., Spears W. M.** A Format Analysis of the Role of Multi-point Crossover in Genetic Algorithms // Annals of Mathematics and Artificial Intelligence. 1992.
3. **Prugel-Bennett A.** The mixing rate of different crossover operators // Foundations of Genetic Algorithms 6.2001. Vol. 6.
4. **Sastry K., Goldberg D., Kendall G.** Genetic Algorithms / http://citeseer.ist.psu.edu/cache/papers/cs2/258/
5. **Eiben A. E., Raue P.-E., Ruttkay Zs.** Genetic Algorithms with Multi-parent Recombination // Proc. of the 3d Conf. on Parallel Problem Solving from Nature, Springer-Verlag. 1994.
6. **Норенков И. П., Арутюнян Н. М.** Смешанный эволюционный метод // Информационные технологии. 2007. № 1. С. 17—20.
7. **Holland J.** Adaptation in Natural and Artificial Systems. Univ. of Michigan Press, 1975.

УДК 004.89

О. А. Николайчук, канд. техн. наук, ст. науч. сотр.,
А. Ю. Юрин, канд. техн. наук, науч. сотр.,
Институт динамики систем
и теории управления СО РАН

Управление опытом при исследовании динамики технического состояния уникальных машин и конструкций: моделирование опыта

Целью данной работы является повышение эффективности повторного использования опыта при решении задач исследования технического состояния уникальных механических систем на основе применения теории управления опытом (experience management) и технологии рассуждений по аналогии (case-based reasoning). Проведено сопоставление процессов, составляющих управление опытом, и этапов цикла решения проблем на основе прецедентов (CBR-цикла). Основной результат — описание гибридной модели представления опыта. Данная модель получена в результате сочетания прецедентной, продукционной и математических моделей представления знаний. Моделирование опыта представлено на уровне моделей предметной и проблемной областей и на уровне модели приложения. Результаты моделирования позволят автоматизировать все этапы управления опытом при исследовании технического состояния уникальных механических систем.

Введение

В последнее время все большее внимание исследователей привлекает проблема повышения эффективности повторного использования опыта при решении исследовательских задач. При этом актуальной является разработка подходов к компьютерной поддержке исследователя на основе опыта или прецедентов [1, 2].

Актуальность данной темы обусловлена тем, что зачастую в областях, где требуется поддержка исследователя, к моменту возникновения проблемы уже накоплен значительный опыт решения подобных проблем. Однако невозможность или сложность повторного использования этого опыта по ряду причин (например, отсутствие экспертов или специалистов необходимой квалификации, отсутствие соответствующего программного обеспечения и т. д.) приводит к невозможности эффективного использования этих знаний и реализации их потенциала.

Задача повышения эффективности исследования динамики технического состояния уникальных машин и конструкций (например, в нефтехимии), включая решение задач определения причинно-следственного комплекса возникновения опасных ситуаций, идентификации, диагностирования и прогнозирования технического состояния, за счет *повторного использования опыта* является актуальной [3, 4]. Актуальность обусловлена как уникальностью процессов, явлений и состояний, возникающих на этапе эксплуатации оборудования, так и их *опасностью* для человека. Своевременное и полное определение параметров технического состояния, выявление, интерпретация и устранение предвестников и причин отказов и аварий позволяют принять максимум мер по их предотвращению.

Одним из способов повышения эффективности использования накопленных знаний при исследовании динамики технического состояния является применение подхода, известного как управление опытом (*experience management*) [2].

Управление опытом

Управление опытом может рассматриваться как особый вид управления знаниями. *Опыт* — это частное знание, которое было приобретено (оценено и сохранено) субъектом в процессе решения проблемы, часто его называют конкретным, фактическим знанием или фактом [2]. По этой причине опыт всегда рассматривается в некотором контексте решения проблемы. В настоящее время существует множество определений *управления знаниями* [2, 5, 6]. В данной работе мы руководствовались следующим: *управление знаниями* — это систематическое, явное и преднамеренное приобретение (сбор), создание и использование знаний в целях максимизации эффективности их применения [6]. Так как *управление опытом* является частным случаем управления знаниями, то определим его как систематическое, явное и преднамеренное приобретение (сбор) и использование опыта в целях максимизации эффективности его повторного применения.

Управление опытом включает ряд процессов созвучных процессам, составляющим управление знаниями: *моделирование, приобретение* (*сбор*), *отбор* (*оценка*), *хранение и сопровождение, применение* (*повторное использование*). Однако эти процессы отличаются от одноименных процессов в управлении знаниями ввиду специфики опыта как конкретного, фактического знания.

Коротко опишем данные процессы в контексте управления опытом.

Моделирование опыта — это процесс отыскания способов представления и формализации опыта.

Приобретение (*сбор*) опыта представляет собой процесс формализации существующих знаний,

представленных как в явной форме (в базах данных, нормативных документах), так и в неявной — в умах экспертов. Так как опыт представляет собой частные (конкретные, фактические) знания, расположенные в контексте конкретной решаемой проблемы, то он легче поддается описанию (формализации и представлению) людьми по сравнению с описанием обобщенных знаний.

Отбор (*оценка*) опытных знаний — это процесс оценки их уместности (релевантности) точности и актуальности. Ввиду того, что опыт расположен в контексте конкретной решаемой проблемы, то его уместность может быть оценена путем сравнения контекста текущей решаемой проблемы с контекстом предыдущих (уже решенных) проблем.

Хранение и сопровождение — это процесс формирования общей памяти в виде хранилища знаний и механизма поддержания ее актуального состояния. В динамических областях, таких как область исследования динамики технического состояния уникальных машин и конструкций, доступный опыт должен непрерывно обновляться, и период его актуальности может быть ограничен. Устаревший опыт (утративший свою актуальность) должен быть выявлен и удален (или обновлен) из хранилища опыта. Опыт, содержащийся в хранилище, может быть рассмотрен как некая совокупность независимых единиц опыта, при этом изменение (замена) отдельной единицы опыта обычно не влияет на поведение системы в целом, в то время как изменение отдельного фрагмента обобщенного знания может сильно повлиять на процесс решения ряда связанных проблем. Основная цель данного процесса — это сохранение опытных знаний для будущего повторного их использования.

Применение — это процесс восстановления и повторного использования знаний при принятии решений. Повторное использование опыта в ряде случаев требует разработки специальных механизмов преобразования (адаптации) опыта согласно контексту новой проблемы. Повторное использование опыта — это *основная* цель управления опытом, для достижения которой необходимо решить ряд задач:

- получить доступ к опыту и отобрать (извлечь) тот опыт, который будет повторно использован;
- оценить полезность отобранного опыта в контексте новой решаемой проблемы;
- адаптировать опыт к новому контексту (при необходимости);
- решить новую проблему (или поддержать пользователя при решении новой проблемы) путем повторного использования отобранного, оцененного и адаптированного опыта.

Отдельные процессы, входящие в управление опытом, могут быть автоматизированы с помощью технологии прецедентных экспертных систем или рассуждений на основе прецедентов (*case-based reasoning*) [2, 7], соединяющей в себе такие методы искусственного интеллекта, как системы поддержки принятия решений, экспертные системы и теорию распознавания образов.

Case-based reasoning и управление опытом

Case-based reasoning (CBR), как отдельное направление исследований в области систем, основанных на знаниях, сфокусировано, прежде всего, на решении проблем на основе опыта, что связано с разработкой и исследованием методов представления (моделирования), оценки, хранения и индексации, извлечения и адаптации опыта. Таким образом, данная технология охватывает основные процессы управления опытом и позволяет их автоматизировать (см. таблицу).

Выделяют ряд этапов, составляющих классический CBR-цикл решения проблемы, на основании имеющегося опыта [2, 7]: *извлечение* наиболее подобного прецедента(ов) (***Retrieve***), *повторное использование* информации, содержащейся в извлеченном прецеденте(ах) (***Reuse***), *просмотр* и *проверка* нового решения (***Revise***), *сохранение* нового прецедента (***Retain***).

Суть подхода к решению задач на основании CBR-цикла заключается в представлении информации о проблемной ситуации в виде некоторого образа (прецедента). Структура образа может быть произвольной, но четко выделяются две части: описание проблемы и ее решение. Накопленный опыт также должен соответствовать данному представлению и образовывать базу прецедентов. Далее, основываясь на введенном частичном описании образа новой проблемы, осуществляется *поиск*, *оценка* и *выбор* потенциально "полезных" образов в базе прецедентов — их *извлечение*. "Полезность" аппроксимируется мерой близости/подобия описаний образов, вычисляемой как расстояние между образами в многомерном признаковом пространстве. При этом делается предположение, что подобные (похожие) проблемы имеют подобные решения.

С ростом базы прецедентов в ряде случаев наблюдается уменьшение эффективности извлечения, что связано с необходимостью анализа все большего числа прецедентов. Для более эффективного извлечения предлагается применять *индексирование* прецедентов, однако формирование и обнаружение индексов зачастую является отдельной сложной задачей.

По результатам *извлечения* отбирается наиболее подобный образ, решение которого с необходимыми модификациями (преобразованиями) назначается образу новой проблемной ситуации, т. е. *по-*

Knowledge managment (обобщенные процессы)	**Experiense managment** (конкретизация процессов knowledge managment)	**Case-based reasoning cycle** (CBR-цикл)
Идентификация (Identify) — определение центральных понятий, стратегий и областей знаний	***Моделирование (Modeling)*** — отыскание способов для представления и формализации опыта; определение модели прецедента и адаптационных операторов	—
Захват (Capture) — захват; формализация существующих знаний	***Сбор, коллекционирование (Collecting)*** — формирование базы прецедентов (БП)	***Сохранение (Retain)***
Выбор (Select) — выбор; оценка релевантности, значимости и точности знаний	***Оценка (Evaluating)*** — оценка повторно используемого опыта в контексте решаемой проблемы. Оценка может быть выполнена, если приемлемы выбранный опыт или его точность и актуальность: • проверка прецендентов БП на актуальность (задача сопровождения БП); • извлечение (поиск, начальная оценка, выбор); • просмотр и проверка (оценка правильности нового решения)	***Извлечение (Retrieve); проверка (Revise)***
Сохранение (Store) — хранение знаний в некой общей памяти	***Сохранение (хранение)/сопровождение (Storing/Maintaining)*** — хранение опытных знаний в некой общей памяти, сопровождение этой памяти, в том числе и поддержание ее актуальности	***Сохранение (Retain)***
Применение (Apply) — извлечение и использование знаний при принятии решений	***Повторное использование (Reusing)*** — основная цель управления опытом, включает: • получение доступа к опыту и выбор опыта, пригодного для повторного использования; • оценку полезности отобранного опыта для решения новой проблемы; • адаптацию при необходимости к новому контексту; • повторное использование опыта	***Извлечение (Retrieve); повторное использование (Reuse)***

вторно используется при решении новой проблемы. Используемый метод *преобразования* (*адаптации*) в значительной степени влияет на качество повторного использования опыта. В настоящий момент существует несколько методов (походов) к адаптации, основное отличие между которыми заключается в применении либо *трансформационного*, либо *порождающего* [7] подхода. Как правило, затраты на адаптацию зависят от степени подобия (близости) между извлеченным и новым прецедентами: чем выше степень подобия, тем меньше затраты.

Новое решение, полученное в результате повторного использования опыта, представляется пользователю, который по возможности *проверяет* его и *пересматривает*. Данный этап слабо отражен в литературе, в большинстве случаев это связано с тем, что задача пересмотра (проверки) решения обычно не является задачей CBR-системы. Решение, определенное CBR-системой, проверяется и по возможности исправляется *человеком* — экспертом предметной области.

Пересмотренный прецедент *сохраняется* в базе прецедентов для дальнейшего повторного использования. На данном этапе происходит *обучение* CBR-системы. Типичная форма обучения в CBR-системах — это обучение путем добавления проверенного прецедента в базу прецедентов. Таким образом, опыт решения новой проблемы становится доступным для повторного использования в будущем.

Необходимо отметить, что перед непосредственным применением CBR-цикла к решению задачи необходимо провести моделирование предметной области. Именно по результатам моделирования можно выбрать (или разработать) модели представления прецедентов, способы (методы) оценки, индексирования, извлечения, адаптации и хранения прецедентов. Поэтому данная статья посвящена рассмотрению этапа моделирования — основополагающего этапа, в соответствии с результатами которого в дальнейшем будет проведена автоматизация процессов, составляющих управление опытом.

Моделирование опыта

Целью моделирования опыта (в контексте CBR) является построение моделей, обеспечивающих эффективное представление (формализацию) и обработку опытных знаний в процессе повторного использования опыта.

Моделирование опыта должно быть основано на моделях как предметной области (ПО), так и проблемной области (ПрО) (рис. 1). В данной работе рассматривается предметная область механических систем [3, 8] и проблемная область задач генезиса, прогнозирования и принятия решений.

При этом перспективным является использование объектно-ориентированного подхода к моделированию, так как данный подход позволяет исследователям оперировать естественными для них понятиями предметной и/или проблемной областей.

Модели предметной области включают предметные сущности (классы и экземпляры классов) и отношения между ними. Модели проблемной области охватывают разнообразные типы задач,

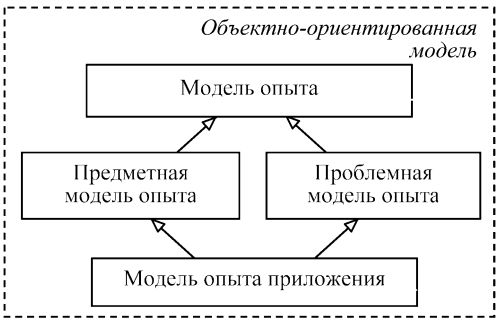


Рис. 1. Структура модели опыта

для решения которых требуются соответствующие представления данных и знаний, а также алгоритмы, реализующие решение (рис. 2).

Комплексное моделирование опыта требует гибридизации этого процесса, т. е. сочетания различных типов моделей методов решения, основанных на моделях предметной и проблемной областей (рис. 2).

Для решения задач исследования динамики технического состояния могут быть использованы разнообразные модели методов, сочетание которых призвано компенсировать недостатки их раздельного применения: *вывод по аналогии, дедуктивный вывод* (*использование продукционных правил*) и *получение решения на основе математических моделей* [4]. Первый метод позволяет получить правдоподобное решение без трудоемкого и тщательного анализа предметной области, но при наличии большого объема опытных данных, представленных в виде прецедентов. Второй метод дает возможность получить достоверное решение и может быть задействован при отсутствии опытных данных, но требует наличия информации о зависимостях между предметными сущностями, процессами, явлениями. Третий метод позволяет получать точные решения, но только при достаточно точной (полной) формализации предметной области. Ввиду того, что задача исследования динамики технического состояния является слабоформализованной и многие закономерности не имеют строгого математического выражения, то применение третьего метода возможно только для уточнения отдельных значений (промежуточных решений) в рамках первых двух подходов.

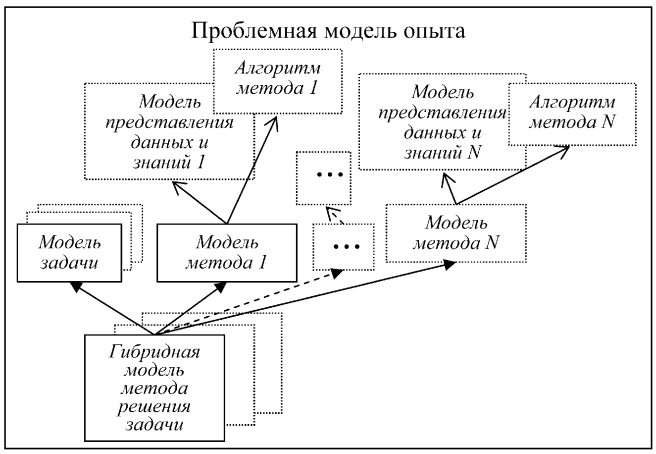


Рис. 2. Структура модели опыта проблемной области

Применение данных методов требует представления опыта в определенном виде, обеспечивающем его эффективную обработку, в частности, необходимо разработать представление опыта в виде прецедентных, продукционных и математических моделей, составляющих *гибридную* модель представления опыта.

Определим компоненты гибридной модели опыта в задаче исследования динамики технического состояния.

Прецедентная модель опыта. В общем случае (обобщенная) модель прецедента имеет следующий вид:

$$П = <Пр, Р>,$$

где П — прецедент; Пр — описание проблемы; Р — описание решения. Под проблемой понимается наличие неизвестных свойств объекта исследования и необходимость их определения по известным свойствам, т. е.

$$Пр = <О, Ц>,$$

где О — объект предметной области; Ц — цель разрешения проблемы. Объект в общем случае описывается набором свойств $О = (Св_1, ..., Св_N)$, где $Св_j$ — свойства объекта исследования, которые могут иметь различную объектно-ориентированную структуру, $j = \overline{1, N}$.

Исходя из условий (требований, ограничений) конкретной предметной области необходимо развить эту обобщенную модель до модели прецедента выбранной предметной и/или проблемной областей, и/или модели прецедента приложения (см. рис. 1).

Рассмотрим, какие ограничения на модель прецедента накладывает проблемная область. Проблемная область исследования динамики технического состояния включает решение задач генезиса, прогнозирования и принятия решения. *Генезис* — это происхождение, становление и развитие, результатом которого является определенное состояние изучаемого объекта. *Прогнозирование* — это определение тенденций динамики конкретного объекта или события на основе анализа его состояния в прошлом и настоящем. *Принятие решения* — это генерация альтернатив и выбор одного решения из множества на основе некоторого набора критериев. Прецедентный подход в общем случае представляет собой комбинирование задач, в нашем случае — генезиса и принятия решений или прогнозирования и принятия решений. Причем предметное наполнение решения будет определяться целью разрешения проблемы.

Как видно из определений, задачи генезиса и прогнозирования направлены на изучение динамики конкретного объекта или события в прошлом и будущем соответственно. В связи с этим структура прецедента должна отражать динамику изменения состояния объекта или события, при этом в описании состояния объекта может присутствовать как статическая информация, так и динамическая:

$$\text{Пр}^{\text{ПрО}} = <\text{O}_0(\text{Св}_{01}, ..., \text{Св}_{0N}), ..., \text{O}_T(\text{Св}_{\text{Т}1}, ..., \text{Св}_{TN}), \text{Ц}>,$$

где $\text{O}_t(\text{Св}_{t1}, ..., \text{Св}_{tN})$ — состояние объекта в t-й момент времени, $t = \overline{0, T}$. Целью решения проблемы является достижение некоторого "целевого" состояния $\text{Ц} = \text{O}_{\text{Ц}}(\text{Св}_{\text{Ц},1}, ..., \text{Св}_{\text{Ц},N})$.

Одним из способов повышения эффективности использования метода поиска решения по аналогии является классификация состояний объекта исследования. Введение классов состояний и явлений в ряде случаев позволяет достигнуть значительного повышения производительности программной реализации метода за счет понижения мощности пространства прецедентов:

$$\text{Пр}^{\text{ПрО}} = <\{\text{O}_{1k_1}\}_{k_1=1}^{N_1}, ..., \{\text{O}_{nk_n}\}_{k_n=1}^{N_n}, \text{Ц}>, \sum_{i=1}^{n} N_i = T,$$

где $\{\text{O}_{ik_i}\}_{k_i=1}^{N_i}$ — i-й класс состояний объекта.

Выбор основания для классификации зависит от предметной области и в зависимости от основания классификации может быть многоуровневой.

Для применения методов частичной прецедентности решения и понижения неопределенности решения задач проблемной области предлагается выделить в модели прецедента отдельный класс описания процессов, протекающих в объекте. Процесс определяется некоторым выбором свойств объекта:

$$\text{Пр}^{\text{ПрО}} = <... \text{Процесс}(\text{Св}_1, ..., \text{Св}_{N_{\text{Процесс}}}), ...>,$$

где $N_{\text{Процесс}}$ — число свойств определяющих процесс.

В данном классе необходимо описывать процессы, наиболее значимые для решения поставленных задач.

Структура прецедента, в частности, его часть, описывающая решение, должна учитывать специфику задач принятия решений. Принятие решений включает формулирование и сопоставление альтернатив, выбор, построение и корректировку гипотезы, таким образом:

$$\text{P}^{\text{ПрО}} = <\text{P}_1, ..., \text{P}_M, \text{Кр}>,$$

где $\text{P}^{\text{ПрО}}$ — модель решения, определяемая проблемной областью; P_m — возможные решения поставленной проблемы в виде объектно-ориентированной структуры (набор альтернатив), $m = \overline{1, M}$; Кр — описание критерия оценивания возможных решений (альтернатив).

При использовании вывода по аналогии возможно получение решения (альтернативы или альтернатив), не удовлетворяющего требованиям пользователя в соответствии с критерием оценивания. Наличие классов решений позволяет определить другие альтернативы, принадлежащие классу полученного решения, которые могут удовлетворить пользователя. Таким образом, предлагается ввести классификацию решений:

$$\text{P}^{\text{ПрО}} = <\{\text{P}_{1k_1}, \text{Кр}_1\}_{k_1=1}^{M_1}, ..., \{\text{P}_{sk_s}, \text{Кр}_s\}_{k_s=1}^{M_s}>, \quad \sum_{j=1}^{s} M_j = M,$$

где $\{\text{P}_{jk_j}, \text{Кр}_j\}_{k_j=1}^{M_j}$ — j-й класс решений по критерию Кр_j.

Выбор основания для классификации зависит от предметной области.

Необходимо отметить, что объект с позиций системного анализа представляет собой систему, обладающую определенной структурой, т. е. состоит из подсистем. В связи с этим необходимо учесть эту структуру путем введения разбиения полного пространства прецедентов на подпространства:

$$\Pi^{\text{ПО}} = \Pi^{\text{ПО}}(\text{O}) = \sum_{\text{Стр}} \Pi(\text{O}_{\text{Стр}}),$$

где $\Pi^{\text{ПО}}$ — пространство прецедентов предметной области; $\text{O}_{\text{Стр}}$ — подсистема или структурный элемент объекта; $\Pi(\text{O}_{\text{Стр}})$ — прецедент структурного элемента объекта; Стр — индекс структурного элемента объекта.

Далее рассмотрим компоненты модели прецедента приложения. Объектом предметной области исследования динамики технического состояния является механическая система, модель которой включает такие свойства, как технические характеристики, технические требования, критерии предельного состояния, ремонтопригодность, мероприятия по обслуживанию и ремонту и т. д. (рис. 3). Данные свойства и образуют тот набор свойств, посредством которого описывается статическое состояние объекта исследования.

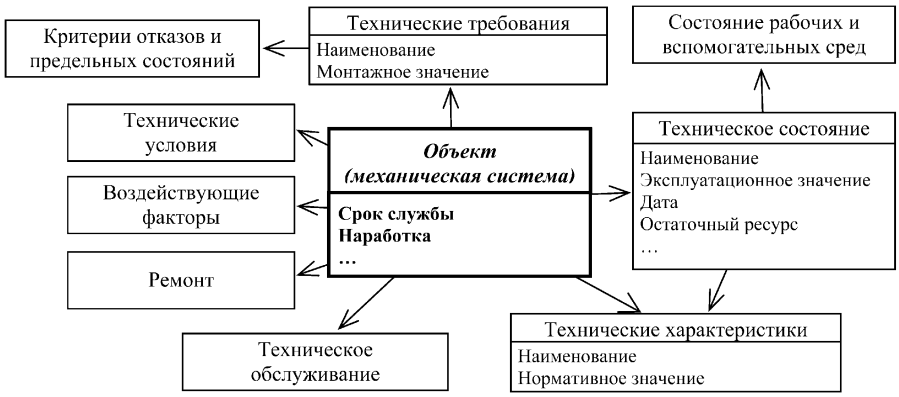


Рис. 3. Предметная модель объекта исследования на языке UML

Динамика технического состояния должна рассматриваться в течение всего "жизненного цикла" механической системы: от этапа проектирования до этапа утилизации. В связи с этим классификация состояний имеет два основания: во-первых, этапы "жизненного цикла": проектирование, изготовление, эксплуатация, утилизация; во-вторых, степень опасности (или нежелательности) состояния на стадии эксплуатации: дефект, повреждение, разрушение, отказ.

С учетом ограничений проблемной и предметной областей прецедент приложения "исследование динамики технического состояния уникальных механических систем" имеет следующий вид:

$$\Pi^{\text{УМС}} = <\{\text{МС}^{\text{Пр}}_{k_{\text{Пр}}}\}^{N_{\text{Пр}}}_{k_{\text{Пр}}=1}, \ldots,$$

$$\{\text{МС}^{\text{Э}}_{\text{Деф}k_{\text{Деф}}}\}^{N_{\text{Деф}}}_{k_{\text{Деф}}=1}, \ldots, \{\text{МС}^{\text{Э}}_{\text{От}k_{\text{От}}}\}^{N_{\text{От}}}_{k_{\text{От}}=1},$$

$$\{\text{МС}^{\text{Ут}}_{k_{\text{Ут}}}\}^{N_{\text{Ут}}}_{k_{\text{Ут}}=1}, \text{НП}, \text{Ц}, \{\text{Р}_{\text{Пр}k_{\text{Пр}}}, \text{Кр}_{\text{Пр}}\}^{M_{\text{Пр}}}_{k_{\text{Пр}}=1}, \ldots,$$

$$\{\text{Р}_{\text{Ут}k_{\text{Ут}}}, \text{Кр}_{\text{Ут}}\}^{M_{\text{Ут}}}_{k_{\text{Ут}}=1}>,$$

где $\Pi_{\text{УМС}}$ — прецедент уникальной механической системы; $\{\text{МС}^{\text{Пр}}_{k_{\text{Пр}}}\}^{N_{\text{Пр}}}_{k_{\text{Пр}}=1}$ — класс состояний при проектировании; $\{\text{МС}^{\text{Э}}_{\text{Деф}k_{\text{Деф}}}\}^{N_{\text{Деф}}}_{k_{\text{Деф}}=1}$ — класс состояний дефекта при эксплуатации; $\{\text{МС}^{\text{Э}}_{\text{От}k_{\text{От}}}\}^{N_{\text{От}}}_{k_{\text{От}}=1}$ — класс состояний отказа при эксплуатации; $\{\text{МС}^{\text{Ут}}_{k_{\text{Ут}}}\}^{N_{\text{Ут}}}_{k_{\text{Ут}}=1}$ — класс состояний при утилизации; НП — нежелательные процессы, протекающие в процессе "жизненного цикла" механической системы; Ц — цель исследования, т. е. достижение состояния механической системы приемлемого уровня надежности; $\{\text{Р}_{\text{Пр}k_{\text{Пр}}}, \text{Кр}_{\text{Пр}}\}^{M_{\text{Пр}}}_{k_{\text{Пр}}=1}$ — класс решений на этапе проектирования; $\{\text{Р}_{\text{Ут}k_{\text{Ут}}}, \text{Кр}_{\text{Ут}}\}^{M_{\text{Ут}}}_{k_{\text{Ут}}=1}$ — класс решений на этапе утилизации.

Классы решений представляют собой перечень мероприятий, осуществление которых позволит достигнуть целевого состояния. Критериями выбора возможных решений являются уровень надежности, который они обеспечивают, и стоимость мероприятий.

Механическая система представляет собой иерархический объект. Иерархия имеет следующую структуру: "деталь—сборочная единица—специфицированное изделие". Данная структура учитывается в модели прецедента путем введения разбиения полного пространства прецедентов на подпространства:

$$\Pi^{\text{ПО}} = \Pi^{\text{ПО}}(\text{О}) = \sum_{i_{\text{МС}}, i_{\text{СЕ}}, i_{\text{Д}}} \Pi(\text{О}_{i_{\text{МС}} i_{\text{СЕ}} i_{\text{Д}}}),$$

где $i_{\text{Д}}$ — индекс деталей, $i_{\text{СЕ}}$ — индекс сборочных единиц, $i_{\text{МС}}$ — индекс специфицированных изделий.

Рассуждения на основе модели (продукционная модель). В общем виде модель опыта в виде продукции можно представить следующим образом:

ЕСЛИ <условие> **ТО** <действие>.

Основываясь на модели предметной области, можно уточнить вид продукции:

ЕСЛИ <свойства объектов(ов) предметной области>
ТО <усвойства объекта(ов) предметной области>.

Исходя из требований проблемной области (см. определение задач) можно выделить следующие виды продукций:

ЕСЛИ <состояние объекта в некоторый момент времени>
ТО <состояние объекта в следующий момент времени>,
и(или)

ЕСЛИ <свойства объекта (состояние объекта в некоторый момент времени)>
ТО <процесс, протекающий на объекте>,
и(или)

ЕСЛИ <нежелательное состояние объекта в некоторый момент времени>
ТО <решение>.

Используя результаты моделирования предметной области, в частности, предложенную классификацию технических состояний объекта исследования (по этапам жизненного цикла и по степени опасности на стадии эксплуатации), определим модели продукций уровня приложения "исследование динамики технического состояния уникальных механических систем":

$$R_{\text{Пр}}: \textbf{ЕСЛИ } \text{МС}_{tik_i} \textbf{ ТО } \text{МС}_{t+1\,i+1\,k_{i+1}},$$

где МС_{tik_i} — состояние объекта из i-го класса состояний в некоторый t-момент времени; $\text{МС}_{t+1\,i+1\,k_{i+1}}$ — состояние объекта из $(i + 1)$-го класса состояний в некоторый следующий $(t + 1)$-й момент времени;

$$R_{\text{НП}}: \textbf{ЕСЛИ } \text{МС}_{tik_i} \textbf{ ТО } \text{НП}_p,$$

где НП_p — p-й нежелательный процесс;

$$R_{\text{НПР}}: \textbf{ЕСЛИ } \text{МС}^{\text{НС}}_{tik_i} \textbf{ ТО } \{\text{Р}_{\text{Пр}k_{\text{Пр}}}, \text{Кр}_{\text{Пр}}\}^{M_{\text{Пр}}}_{k_{\text{Пр}}=1},$$

$$\ldots, \{\text{Р}_{\text{Ут}k_{\text{Ут}}}, \text{Кр}_{\text{Ут}}\}^{M_{\text{Ут}}}_{k_{\text{Ут}}=1},$$

где $\text{МС}^{\text{НС}}_{tik_i}$ — нежелательное состояние в некоторый i-й момент в прошлом, являющееся причиной возникновения некоторого текущего состояния.

Продукции образуют базу знаний, концептуальная структура которой имеет следующий вид:

$$\text{БЗ} = \langle \text{МС, НП, Причины, Решения,}\ R_{\text{Пр}}, R_{\text{НП}}, R_{\text{НПР}} \rangle,$$

где МС — свойства механической системы; НП — модели нежелательных процессов; Причины — причины нежелательных процессов; Решения — решения по устранению причин возникновения нежелательных процессов; $R_{\text{Пр}}$ — причинно-следственные отношения между состояниями механической системы; $R_{\text{НП}}$ — причинно-следственные отношения между состояниями механической системы и нежелательными процессами; $R_{\text{НПР}}$ — причинно-следственные отношения между причиной нежелательных процессов и решениями.

Математические модели. Моделирование опыта невозможно без использования математических моделей процессов, явлений и событий, которые являются высшей формой обобщения опыта.

Тип математической модели определяется требованиями проблемной области, ее идентификация осуществляется на основе предметной области, а принятие конкретного решения возможно только в рамках приложения.

В случае решения задачи исследования динамики технического состояния математический аппарат может быть использован как метод определения значений отдельных свойств при описании объекта исследования, включая его статическое и динамическое состояние. Например, описание роста макротрещин, описание изменения остаточных напряжений в детали и т. д.

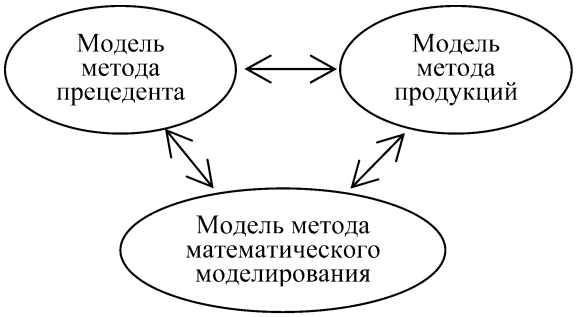


Рис. 4. Гибридная модель метода решения задачи

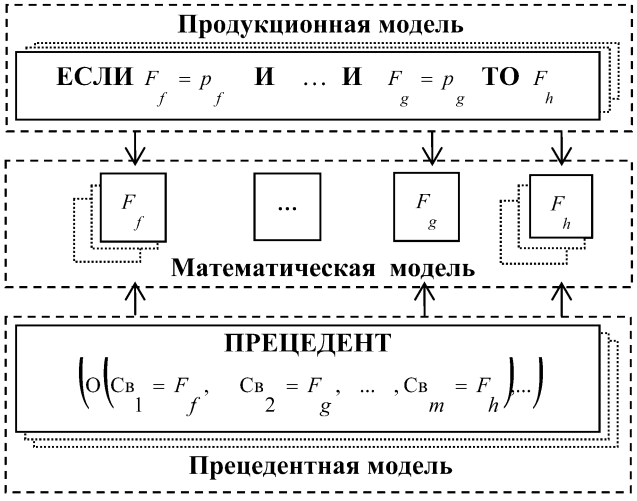


Рис. 5. Комбинация прецедентных, продукционных и математических моделей

Гибридная модель. Гибридное моделирование опыта может быть основано на трех описанных выше моделях опыта (рис. 4). Порядок и способы сочетания моделей определяются исходя из особенностей предметной и проблемной областей. В частности, на основе математических моделей могут быть определены значения свойств, составляющих как элементы прецедентной, так и продукционной моделей, в свою очередь, продукционные модели могут быть использованы при адаптации решений, полученных по аналогии (рис. 5):

$$\text{МО} = \langle \text{П, Продукция, М, } F_{\text{ПМ}}, F_{\text{Продукция М}}, F_{\text{Продукция П}} \rangle,$$

где МО — модель опыта; П — прецедентная модель; Продукция — продукционная модель; М — математическая модель; $\text{F}_{\text{ПМ}}$: П → М — множество отношений между прецедентами и математическими моделями; $F_{\text{Продукция М}}$:Продукция → М — множество отношений между продукциями и математическими моделями; $F_{\text{Продукция П}}$ → Продукция — множество отношений между продукциями и прецедентами.

Заключение

Решение проблемы управления опытом позволяет повысить эффективность труда во многих областях, в которых к настоящему моменту существует невостребованный опыт решения исследовательских задач. Особо это актуально при решении задач, связанных с исследованием опасных для человека объектов, например, с уникальными машинами и конструкциями в нефтехимии.

В статье подробно рассмотрено моделирование опыта как один из этапов, составляющих подход к повышению эффективности управления опытом при исследовании динамики технического состояния уникальных машин и конструкций. Результатом этапа моделирования является модель представления знаний, на основании которой предлагается автоматизировать процесс повторного использования опыта с помощью технологии case-based reasoning (рассуждений на основе прецедентов).

Описываемая модель является обобщением модели опыта и разработана на основе ограничений накладываемых проблемной и предметной областями. Модель является *гибридной* и получена в результате комбинирования подходов к представлению знаний в виде прецедентов, логических и математических моделей.

На основании данной *гибридной* модели опыта в дальнейшем будет проведена автоматизация всех этапов управления опытом: приобретения, хранения и сопровождения, оценки и повторного использования опыта для решения задач генезиса, прогнозирования и принятия решений при исследовании динамики технического состояния уникальных машин и конструкций.

Список литературы

1. **Люгер Дж. Ф.** Искусственный интеллект: стратегии и методы решения сложных проблем. 4-е изд.: Пер. с англ. М.: Вильямс, 2003.

2. **Bergmann R.** Experience Management // Lecture notes on artificial intelligence. 2002. Vol. 2432. P. 1—364.

3. **Берман А. Ф.** Деградация механических систем. Новосибирск: Наука, 1998.

4. **Николайчук О. А., Юрин А. Ю.** Информационно-логико-математический подход для исследования технического состояния систем // Матер. междунар. науч.-техн. конф. "Искусственный интеллект. Интеллектуальные многопроцессорные системы-2006 (ИИ — ИМС'2006)", Украина, п. Кацевели, 24—29 сентября 2006 г. Таганрог, 2006. Т. 3. С. 286—291.

5. **Попов Э. В., Фоминых И. Б., Харин Н. П., Виньков М.** М., Управление знаниями, www.rfbr.ru/default.asp?doc_id = 20742

6. **Wiig K**. Knowledge Management: An Introduction and Perspective // J. of Knowledge Management. 1997. Vol. 1. P. 6—14.

7. **Aamodt A., Plaza E.** Case-Based reasoning: Foundational issues, methodological variations, and system approaches // AI Communications. 1994. Vol. 7. N 1. P. 39—59.

8. **Берман А. Ф., Николайчук О. А.** Объектно-ориентированная модель исследования безопасности сложных технических систем // Матер. II Междунар. конф. "Проблемы управления и моделирования сложных систем". Самара, 2000. С. 366—371.

СЕТИ И СИСТЕМЫ СВЯЗИ

УДК 004.52

В. Г. Шахов, канд. техн. наук, **С. В. Нопин,** аспирант,
Омский государственный университет путей сообщения

IP-телефония в защищенном режиме: варианты реализации

Предлагается передача речевых сообщений по IP-протоколам в защищенном от подслушивания режиме по Интернет-каналам. Рассматриваются в основном варианты маскирования речи на основе общеупотребимых операционных систем. Это дает гарантированное качество защиты, поскольку используемые программы разработаны профессионалами, и обеспечивает наименьшие затраты, так как эти программные продукты дополнительно не оплачиваются. Приведены рекомендации по приемлемым вариантам конфиденциальной речевой связи.

В соответствии с предыдущей статьей [5], где сформулированы основные задачи конфиденциальной речевой связи, в данной работе освещены конкретные варианты реализации.

Существуют классические труды в области защиты речевых сообщений, из которых наиболее значимым является фундаментальная работа Л. Рабинера и Р. Шафера [2]. Классически определены три способа защиты: **преобразование спектра, скремблирование и вокодирование.**

Преобразователи спектра используют некоторые процедуры в частотной области. Простейший режим — обращение спектра, описанное, в частности, в работах

[6, 7]. С помощью специальной техники речь вначале модулируется на какой-то несущей частоте, после чего выбирается нижняя боковая полоса (НБП), которая потом возвращается на заданную частоту передачи. Метод недостаточно эффективен, так как любой нарушитель, имеющий аналогичную аппаратуру, получает прямой доступ к каналу.

Более сложный вариант — модуляция сообщений с использованием случайно выбираемой несущей. Закон изменения несущей известен только двум клиентам связи. Можно модифицировать алгоритм введением случайного интервала изменения несущей, в этом случае появляется двумерная матрица сканирования по частоте и времени. Этот метод очень эффективен при попытках несанкционированного доступа [6]. Его главный недостаток — требование широкой полосы частот. В частности, метод используется в системах сотовой связи, но с другой целью (для уплотнения каналов). Кроме того, в каждом таком приложении используются свои протоколы взаимодействия, которые непосредственно не состыковываются.

Метод ***скремблирования*** основан на перестановке во времени фрагментов речи, причем перестановки могут быть случайными, как и интервалы выборок фрагментов. Метод достаточно эффективен, но имеет два недостатка. Первый связан с задержками дуплексной связи. Скремблер должен оперировать на определенном временном интервале, причем чем больше интервал, тем мощнее защита, но тем больше задержки связи. Существуют нормативы на задержки связи [3], которые ограничивают уровень защиты, следовательно, и возможности метода.

Вторая проблема скремблирования — синхронизация передающей и приемной частей. В работе [5] достаточно подробно описана система скремблирования речевых сообщений, в том числе способ синхронизации, защищенный патентом и основанный на специальных кодах Баркера.

Вокодерные системы маскирования состоят в том, что речевое сообщение разбивается на узкополосные частотные поддиапазоны, каждый из которых может переставляться на другую частоту, в том числе по случайному закону, известному двум клиентам связи. В последнем варианте это близко к ранее описанному принципу модуляции по случайному закону. Вокодерные системы вполне могут интерпретироваться в цифровой форме, в том числе с использованием цифровой фильтрации [8].

Недостатком вокодерных систем является повышенная сложность алгоритмов, даже с использованием систем цифровой обработки сигналов.

Нам представляется наиболее перспективным метод цифрового маскирования речевых сообщений, структура которого представлена на рис. 1.

Генератор псевдослучайной последовательности (ПСП) формирует поток битов, который в простейшем случае побитно суммируется с потоком оцифрованных данных. Основным требованием к предложенной системе маскирования является синхронизация генераторов ПСП на передающей и приемных частях. Она может выполняться различными способами, в том числе известными в технике связи.

Достоинствами предложенной структуры маскирования являются следующие.

1. Предложенный вариант наиболее просто реализуется при наличии надежного генератора ПСП, поэтому предполагает сквозное (потоковое) маскирование, известное в теории секретных систем как гаммирование [9]. Кстати, такой метод маскирования относится к разряду абсолютно секретных шифров, известных как шифрование с бесконечным ключом, или с шифрблокнотом. Это "изюминка" российской школы шифрования, первой в мире по стойкости шифров. Отечественный стандарт шифрования по ГОСТ 28147—89 является лучшим шифром, до которого "не дотягивают" и последние зарубежные шифры, включая американский стандарт AES [9].

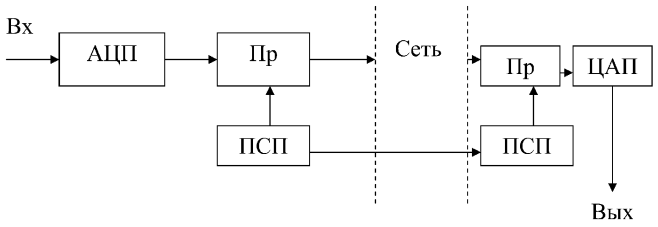


Рис. 1. Структура системы конфиденциальной связи:
Пр — процессор; ПСП — генератор псевдослучайной последовательности

2. В качестве источников гаммы могут быть использованы различные алгоритмы, в том числе встроенные в имеющиеся операционные системы, например, в *Windows.*

3. Имеющиеся в типичных ОС источники гаммы защищаются ключами, длина которых может быть выбрана такой, что за реальное время речевые сообщения не могут быть восстановлены нарушителями при перехвате цифровых кодов.

4. Образующийся после маскирования цифровой поток легко адаптируется под существующие технологии коммутации пакетов, поэтому не возникает дополнительных задержек, кроме существующих в компьютерных сетях.

Авторами предложен метод маскирования речевых сообщений с помощью встроенных средств ОС *Windows NT* и ее производных (XP, 2000, 2003, Vista).

Современные IBM-совместимые компьютеры, как правило, обладают аппаратной возможностью вводить-выводить звук с помощью стандартной звуковой карты [10]. Во всех версиях ОС *Windows* (начиная с *Windows* 95) присутствуют специальные интерфейсы, предназначенные как для ввода-вывода звука, так и для преобразования форматов звуковых данных. Одним из интерфейсов ввода-вывода звука является *DirectSound OC Windows.* Основным преимуществом данной технологии над технологиями по воспроизведению и записи звука базовыми методами является концепция абстракции и эмуляции оборудования. Ее наличие гарантирует, что вне зависимости от типа используемого оборудования все действия при работе с *DirectSound* будут иметь одинаковый результат на любой ЭВМ. Все возможности по воспроизведению или записи обязательно будут реализованы на любой ПЭВМ, оборудованной устройствами воспроизведения/записи. Это, в том числе, означает, что при использовании *DirectSound* обеспечивается максимальная производительность, которая возможна для данной ПЭВМ и максимальная вероятность того, что программа, использующая данный интерфейс, будет корректно работать на всех компьютерах с любыми звуковыми картами. Интерфейс преобразования форматов звуковых данных называется (ACM) *Audio Compression Manager*

(диспетчер сжатия звука) [10]. Интерфейс позволяет изменять частоту, разрядность, число каналов, а также тип сжатия звуковых данных. При достаточной мощности процессора преобразование может выполняться в реальном времени, что позволяет использовать эти аудиоинтерфейсы для реализации систем *IP*-телефонии. ACM включает в себя набор аудиокодеков, выполняющих необходимые преобразования. Как правило, аудиокодек позволяет осуществить не только сжатие, но и распаковку звуковых данных, т. е. восстановление исходного сигнала.

В настоящее время не существует сложившихся стандартов реализации систем *IP*-телефонии, защищенных от несанкционированного доступа с учетом особенностей и возможностей операционной системы *Windows*. В международных стандартах, с одной стороны, есть требования к системам телефонии, в том числе к компрессии звука и некоторым функциям защиты, с другой стороны, они не учитывают возможностей современных операционных систем, например ОС *Windows*, а ориентированы прежде всего на аппаратные реализации в сетях, обеспечивающих доставку данных реального времени по соответствующим протоколам.

Для оценки качества маскирования были выбраны следующие множества частных показателей:

1. Качественные показатели (M) характеризуют субъективные результаты измерения и оценивания звучания речи на основе оценок отдельных экспертов. Они характеризуют эффективность и степень соответствия систем установленным при проектировании требованиям.

2. Технологические показатели (T) характеризуют качество программных решений с общих позиций современной технологической базы. Для систем реального времени такими показателями могут быть: коэффициенты использования ЦП (T_1) и сетевых ресурсов (T_2), простота и функциональность настройки параметров системы (T_3), оперативность отображения необходимой статической и динамической информации (T_4), надежность функционирования (T_5), возможность динамического наращивания функциональных возможностей в отношении алгоритмов компрессии (T_6) и защиты информации (T_7), например, путем использования кодеков и/или криптоалгоритмов на основе аудио- и криптографических интерфейсов ОС *Windows*.

3. Показатели, характеризующие качество защиты информации (K): криптографическая стойкость защиты пакетов данных — алгоритм шифрования (K_1), совокупная длина ключа шифрования (K_2), алгоритм хеширования (K_3), длина имитовставки (K_4), простота и безопасность распространения ключей (паролей) для криптографической защиты (K_5), возможность вести контроль и настройку режимов криптографической защиты трафика *IP*-телефонии (K_6), возможность использования сертифицированных ФСБ РФ криптоалгоритмов и криптомодулей для того, чтобы официально применять систему *IP*-телефонии в органах государственной власти и силовых ведомствах (K_7).

4. Показатели, характеризующие совокупную сложность реализации программного обеспечения (Q), включающую в себя алгоритмы компрессии речи (Q_1), защиты данных (Q_2), формирования и обработки *IP*-пакетов (Q_3) и их передача/прием по *IP*-сети (Q_4), алгоритмов управления обслуживанием вызова, т. е. принятия решений о том, каким образом должно быть установлено соединение между абонентами (Q_5).

5. Экономические показатели E характеризуют затраты, связанные с обеспечением требуемых функциональных характеристик, которые складываются из стоимости проектирования и разработки ПО (E_1), минимально необходимой конфигурации аппаратных средств (E_2) и стоимости эксплуатации системы *IP*-телефонии (E_3).

В итоге критерий оценки эффективности K_{IP} системы *IP*-телефонии, функционирующей в защищенном режиме, примет вид функции пяти переменных:

$$K_{\text{IP}} = (M,\ T,\ K,\ Q,\ E).$$

Сворачивание частных показателей $T_1 \ldots T_7$, $K_1 \ldots K_7$, $Q_1 \ldots Q_5$, $E_1 \ldots E_3$ в показатели качества T, K, Q, E соответственно проводится с помощью весовых коэффициентов. Для каждой составляющей определяются (назначаются в зависимости от целей анализа) весовые коэффициенты $(t_1, t_2, t_3, t_4, t_5, t_6, t_7) = t$ (для составляющей показателя качества T), $(k_1, k_2, k_3, k_4, k_5, k_6, k_7) = k$ (для составляющей показателя качества K), $(q_1, q_2, q_3, q_4, q_5) = q$ (для составляющей показателя качества Q), $(e_1, e_2, e_3) = e$ (для составляющей показателя качества E).

Для численной оценки показателей качества M, T, K, Q, E предлагается аддитивный подход, тогда формулы оценки качества примут следующий вид:

$$T = \sum_{i=1}^{7} (t_i\ T_i), \tag{1}$$

$$K = \sum_{i=1}^{7} (k_i\ K_i), \tag{2}$$

$$Q = \sum_{i=1}^{5} (q_i\ Q_i), \tag{3}$$

$$E = \sum_{i=1}^{3} (e_i\ E_i). \tag{4}$$

Для оценки общего критерия качества также используется аддитивная методика:

$$K_{IP} = k_M M + k_T T + k_K K + k_Q Q + k_E E, \tag{5}$$

где k_M — весовой коэффициент значимости качественных показателей звучания речи для системы в целом; k_T — весовой коэффициент значимости технологических показателей для системы в целом; k_K — весовой коэффициент значимости совокупной сложности реализации программного обеспечения для системы в целом; k_Q — весовой коэффициент значимости защиты информации для системы в целом; k_E — весовой коэффициент значимости экономических показателей.

В соответствии с выбранными шкалами оценок исходных показателей лучшим вариантом системы *IP*-телефонии, функционирующей в защищенном от несанкционированного доступа режиме, может считаться тот, для которого выполняется условие максимальной численной оценки:

$$K_{IP} = K_{IPMAX}. \tag{6}$$

Кроме соответствия требованиям к качеству применяемых криптографических алгоритмов, программные средства системы *IP*-телефонии, работающей в защи-

щенном режиме, должны соответствовать некоторым требованиям безопасности своего применения, т. е. не должны содержать явных или скрытых функциональных возможностей, позволяющих:

- модифицировать алгоритмы криптографических преобразований в процессе эксплуатации ПО;
- модифицировать или изменять информационные или управляющие потоки, связанные с функционированием системы;
- осуществлять доступ посторонних лиц к ключевой или парольной информации;
- осуществлять доступ посторонних лиц к конфиденциальной информации, проходящей через модули криптографической защиты информации.

Защита от несанкционированного доступа *IP*-телефонного трафика в органах государственной власти и силовых ведомствах имеет дополнительное требование — решения *IP*-телефонии должны использовать сертифицированные ФСБ РФ криптоалгоритмы и криптомодули.

Разработка методов и алгоритмов обработки данных требует решения задачи моделирования и тестирования программных реализаций отдельных элементов системы или отдельных подсистем в целом, в том числе имитационного моделирования в среде GPSS [13].

Представленный сценарий функционирования системы передачи речи включает в себя ряд псевдопараллельных задач, осуществляющих обработку данных в режиме реального времени. Это накладывает естественные ограничения на загрузку ЦП и устройства ввода-вывода. Для анализа функционирования системы наиболее целесообразно выявить моменты максимальной загрузки. Сценарий ввода информации, ее обработки и дальнейшей передачи (вывода) можно рассмотреть с позиции системы массового обслуживания. В рамках данного подхода каждый элемент (модуль, поток) системы передачи речи может рассматриваться в качестве обслуживающего устройства, а блоки передаваемых данных от источников данных — в виде дискретных событий — заявок на обслуживание.

Основными параметрами имитационной модели являются интенсивность поступления транзактов λ или объем блока данных, передаваемых вместе с транзактом, причем λ для потоков записи/воспроизведения обратно пропорциональна объему блока данных согласно формуле

$$\lambda = \frac{\nu}{M_{\text{Б}}}, \qquad (7)$$

где ν — частота дискретизации; $M_{\text{Б}}$ — число отсчетов (объем блока с данными в байтах), передаваемых за один транзакт.

Для потоков передач/приема блоков с данными через *IP*-сеть интенсивность поступления транзактов λ определяется согласно формуле

$$\lambda = \frac{1}{M_{\text{п}}}, \qquad (8)$$

где $M_{\text{п}}$ — математическое ожидание времени передачи/приема блоков с данными.

Дополнительными ограничениями является то, что квант времени, выделяемый планировщиком ОС *Windows* на выполнение задачи, составляет от нескольких миллисекунд до нескольких десятков миллисекунд в зависимости от версии ОС *Windows* и типа ЦП, обычно около 20 мс.

На рис. 2 представлены обработанные статистические результаты проведенного моделирования в среде GPSS в виде графика, характеризующего долю отброшенных пакетов с данными, не успевшими вовремя попасть в буфер потока воспроизведения.

На рис. 3 отображена минимальная длина динамического буфера за период моделирования со следующими исходными данными: период генерации транзактов 100 ± 10, 20, 30, 40 мс; время обработки каждого транзакта 10 ± 10 мс; шаг дискретизации модельного времени 1 мс; общее время моделирования 100 000 мс. По оси абсцисс отложено число пакетов с данными, попадающих в буфер воспроизведения до непосредственного начала воспроизведения звука.

Таким образом, разработанная модель на языке имитационного моделирования GPSS позволяет определять характеристики системы *IP*-телефонии — отношение времени нормальной голосовой связи между абонентами *IP*-телефонии к времени пропадания звука и минимально необходимую длину буфера воспроизведения для поддержания заданного качества обслуживания.

Сформулируем следующие основные технические требования к системе передачи речи через вычислительные сети, работающей в защищенном режиме по сценарию "компьютер"—"компьютер":

- поддержка ввода/вывода звуковых данных с разных звуковых адаптеров;

Рис. 2. Зависимость потерь аудиоинформации от начальной длины буфера

Рис. 3. Зависимость минимального объема динамического буфера от начальной длины буфера

- поддержка ввода/вывода пакетов с данными с разных сетевых адаптеров;
- обеспечение возможности создания буфера разной длины при воспроизведении звука для регулирования качества обслуживания пользователя;
- унифицированный интерфейс управления системой;
- возможность интеграции (реинтеграции) необходимых компонентов (модулей) для изменения архитектуры системы в каждом сеансе связи;
- возможность компрессии (декомпрессии) звуковых данных разными алгоритмами;
- возможность защиты данных с помощью разных криптоалгоритмов, функционирующих в различных режимах.

На основе разработанной программной архитектуры системы *IP*-телефонии [10], защищенной от несанкционированного доступа, можно реализовать программную архитектуру системы защиты данных от несанкционированного доступа [11], которая включает в себя аналогичную структуру программных компонентов и связей этих компонентов в упрощенном и усеченном виде. Отличительной особенностью данного класса систем является отсутствие требования функционирования в режиме реального времени.

Список литературы

1. **Шахов В. Г., Нопин С. В.** Использование IP-телефонии в защищенном режиме // Информационные технологии. 2008. № 5. С. 7—11.
2. **Рабинер Л. Р.** Цифровая обработка речевых сигналов / Л. Р. Рабинер, Р. В. Шафер. — М.: Радио и связь, 1981. 454 с.
3. **Фомин В. В.** Устройство защиты радиотелефонных переговоров. / В. В. Фомин — Автореф. дис., Омск, 1998. 18 с.
4. **Фомин В. В.** Анализ системы защиты закрытой радиотелефонной связи (УЗРП для радиостанций УВД) // В. В. Фомин, В. Г. Шахов, И. М. Леонидов. — Депонированные работы. 1996. № 12, реф. № 6063-ж. д. 96. 20 с.
5. **Фомин В. В.** Синхронизация устройств защиты речевой информации в полосе канала ТЧ // В. В. Фомин. Депонированные работы. 1998. № 4, реф. № 312-ж. д. 98. 22 с.
6. **Шахов В. Г.** Безопасность информационных систем / В. Г. Шахов. М.: Транспорт, 2001. 212 с.
7. **Петраков А. В.** Утечка и защита информации в телекоммуникационных каналах / А. В. Петраков, В. С. Лагутин — М.: Энергоатомиздат, 1997. — 304 с.
8. **Рабинер Л. Р.** Цифровая обработка сигналов / Л. Р. Рабинер, Р. В. Шафер — М.: Радио и связь, 1981. — 496 с.
9. **Иванов М. А.** Криптографические методы защиты информации в компьютерных системах и сетях / М. А. Иванов — М.: КУДИЦ-ОБРАЗ, 2001. — 368 с.
10. **Гордеев О.** Программирование звука в Windows. Руководство для профессионалов / О. Гордеев — СПб.: BHV, Санкт-Петербург, 1999. — 364 с.
11. **Нопин С. В.** Защита данных с помощью встроенных криптографических интерфейсов операционной системы Windows / С. В. Нопин, В. Г. Шахов // Омский научный вестник. — 2006. — № 7(43). — С. 131—134.
12. **Нопин С. В.** Разработка защищенных от несанкционированного доступа систем IP-телефонии на основе операционной системы Windows / С. В. Нопин, В. Г. Шахов // Омский научный вестник. — 2006. — № 9(46). — С. 137—142.
13. **Шрайбер Д.** Моделирование на GPSS / Д. Шрайбер — М.: Мир, 1978. — 592 с.

УДК 004.7:519.21

А. В. Бородин,
Ю. А. Богоявленский, канд. техн. наук, доц.,
Петрозаводский государственный университет

Математическая модель беспроводного сегмента IEEE 802.11 в режиме насыщения

Рассмотрены проблемы адекватной оценки характеристик производительности сегмента беспроводной сети, работающей под управлением распределенного алгоритма доступа к среде передачи, в зависимости от числа станций в сегменте. Предложена полумарковская модель процесса захвата среды станцией в насыщенном беспроводном сегменте. Проведено сравнение значений, получаемых на модели, с экспериментальными данными.

Введение

На сегодняшний день одной из наиболее распространенных технологий локальных беспроводных сетей является Wi-Fi, определяемая семейством стандартов IEEE 802.11 [1]. В этих сетях управление захватом передающей среды проводится всеми станциями, участвующими в передаче, на основе распределенного соревновательного метода множественного доступа CSMA/CA DCF. При росте нагрузки происходит значительное падение производительности сегмента беспроводной сети в результате роста задержек, связанных с определением станции, захватившей канал на время следующей передачи.

Для анализа производительности беспроводного сегмента, работающего в режиме насыщения, нами предложена полумарковская модель процесса захвата среды станцией, управляемого счетчиком отсрочки передачи кадра и счетчиком числа повторных коллизий станции в течение соревновательного цикла. Эта модель является развитием марковской модели, предложенной Джузеппе Бьянки [2]. Расшифровки упоминаемых в статье аббревиатур см. в [1].

1. Модель процесса захвата среды станцией

1.1. Допущения

При разработке модели были сделаны следующие допущения.

- Ошибки, связанные с искажением кадров при передаче (например, по причине действия шума в канале связи), отсутствуют. Таким обра-

зом, кадр может быть принят с ошибкой только вследствие коллизии.

- В сети отсутствуют скрытые станции, задержки распространения сигнала пренебрежимо малы по сравнению с длительностью межкадровых промежутков, и сегмент беспроводной сети состоит из конечного фиксированного числа взаимодействующих станций, каждая из которых всегда имеет кадр для передачи (работает в условиях насыщения).
- Основное упрощающее допущение состоит в том, что вероятность коллизии переданного кадра постоянна и не зависит от числа попыток передачи этого кадра, предпринятых ранее. Мы предполагаем известным распределение длин передаваемых кадров. В рассмотрение вводятся также длины межкадровых промежутков (определенные стандартом).

На практике хорошим приближением такой сети будут несколько станций, размещенных в одном помещении в пределах прямой видимости при отсутствии искусственных источников помех. Допущение независимости вероятности коллизии от числа попыток будет, очевидно, тем более справедливо, чем ближе находится сеть к состоянию насыщения.

1.2. Механизм захвата среды станцией

Рассмотрим беспроводный сегмент, содержащий n станций, соревнующихся за передающую среду. В условиях насыщения MAC-протокол станции после завершения каждой успешной передачи сразу же получает очередной кадр от протокола вышестоящего уровня.

Станция, захватившая канал на время следующей передачи, определяется на основе распределенного алгоритма множественного доступа с контролем несущей и предотвращением коллизий CSMA/CA. Правило выбора счетчика отсрочки реализуется распределенной координационной функцией (РКФ), являющейся частью алгоритма CSMA/CA. Согласно алгоритму станция связывает с принятым к передаче кадром случайное целое число в интервале $[0, W-1]$, называемое *счетчиком отсрочки*. Значение W определено стандартом и называется *начальным соревновательным окном*. Работа алгоритма определяется совокупностью последовательных интервалов времени, называемых *временными слотами*. В начале каждого слота счетчик отсрочки уменьшается на единицу, когда значение счетчика достигает нуля, станция немедленно начинает передачу кадра.

Заметим, что уменьшение времени отсрочки останавливается, как только канал опознается занятым, и, таким образом, длительность временного интервала между началами двух последовательных слотов варьируется в широких пределах от размера пустого слота до размера слота, в течение которого осуществляется передача кадра максимальной длины.

В случае, когда две или несколько станций осуществляют попытку передачи кадров в течение одного слота, происходит коллизия — искажение кадров, ведущее к неверному приему. Станция, обнаружившая коллизию, выбирает новое значение счетчика отсрочки, удваивая размер соревновательного окна.

Интервал времени от момента вовлечения кадра в соревновательный процесс до успешной передачи или исчерпания максимального числа последовательных коллизий будем называть *соревновательным циклом*. Таким образом, успешной передаче кадра может предшествовать один или несколько соревновательных циклов.

В связи с трудностью распознавания коллизий в беспроводной среде посредством прослушивания собственной передачи в протоколе предусмотрена отправка специального кадра подтверждения (ACK). Станция может осуществлять захват передающей среды, выставляя в канал биты кадра данных. Такой метод называется основным (basic access). Однако, поскольку обнаружение коллизии основывается только на получении кадра подтверждения, интервал распознавания коллизии может быть достаточно большим. Поэтому введен дополнительный метод доступа (RTS/CTS), предусматривающий предварительную отправку кадра приглашения (RTS) и получение кадра подтверждения на прием (CTS). В этом случае накладные расходы на передачу данных увеличиваются, однако скорость распознавания коллизий резко возрастает.

1.3. Полумарковская модель процесса захвата среды

Следуя [2], рассмотрим $b(t)$ — случайный процесс, представляющий значение счетчика отсрочки станции. Поскольку значение счетчика отсрочки для каждой станции зависит от истории передач, этот случайный процесс — немарковский. Пусть m — максимальный уровень отсрочки, обозначим $W_i = 2^i W$, где $i \in [0, m]$, называется *уровнем отсрочки*. Рассмотрим $s(t)$ — случайный процесс, представляющий значение $(0, ..., m)$ уровня отсрочки станции в момент t.

Рассмотрим моменты времени t, соответствующие началам временных слотов. Тогда двумерная последовательность $\{s_k, b_k\} = \{s(t_k), b(t_k)\}$ является марковской цепью, граф переходов которой приведен на рис. 1. В отличие от [2] нами введены в рассмотрение два дополнительных состояния "Fail" и "Success", отражающих соответственно не-

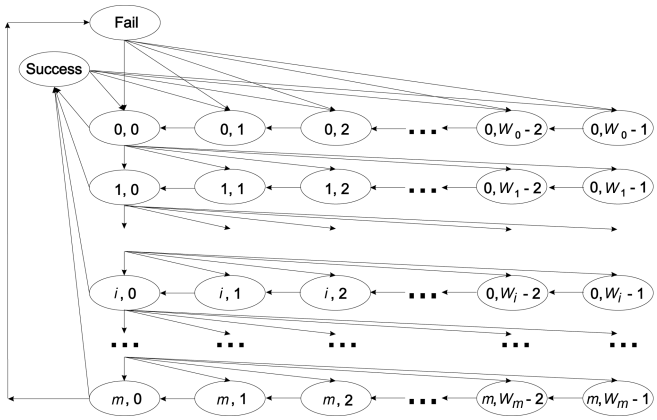


Рис. 1. Граф переходов процесса вложенной марковской цепи

удачное завершение цикла по максимальному числу коллизий и передачу кадра. Заметим, что в действительности рассматривается семейство процессов, параметризованное по числу станций.

Двумерный процесс $\{s(t), b(t)\}$ является полумарковским процессом, управляемым вложенной цепью $\{s_k, b_k\}$. Введение полумарковского процесса позволяет учесть в модели непостоянство длин временных слотов, возникающее в результате остановки уменьшения счетчика отсрочки при опознании канала как занятого. Множество всех состояний процесса $\{s(t), b(t)\}$ обозначим S.

1.4. Стационарное распределение вложенной цепи

Будем считать, что в начальный момент времени процесс находится в состоянии "Success". Обозначим $P_{\{A|B\}}$ — вероятность перехода в состояние A при условии пребывания системы в состоянии B, $\rho(n)$ — вероятность того, что станция, предпринявшая попытку передачи, терпит неудачу вследствие коллизии. Тогда ненулевые переходные вероятности цепи определяются выражениями

$$P_{\{Success|i,\,0\}} = 1 - \rho(n),\ i \in (0,\ m-1);$$

$$P_{\{0,\,k|Success\}} = \frac{1}{W},\ k \in (0,\ W-1);$$

$$P_{\{Fail|m,\,0\}} = \rho(n);$$

$$P_{\{0,\,k\}|Fail} = \frac{1}{W},\ k \in (0,\ W-1);$$

$$P_{\{i,\,k|i,\,k+1\}} = 1,\ k \in (0,\ 2^iW-2),\ i \in (0,\ m);$$

$$P_{\{i,\,k|i-1,\,0\}} = \frac{\rho(n)}{2^iW},\ k \in (1,\ 2^iW-1),\ i \in (0,\ m).$$

Марковская цепь конечна, неприводима и апериодическая, следовательно, она обладает свойством эргодичности, т. е. существует стационарное распределение цепи $\{\pi_{(i,\,k)}(n)\}$. Рассуждая по аналогии с [2] и [3], получим соотношения для стационарных вероятностей цепи:

$$\pi_{(i,\,k)}(n) = \frac{2^iW-k}{2^iW}\rho(n)^i\pi_{(0,\,0)}(n),$$

$$i \in (0,\ m),\ k \in (0,\ 2^iW).$$

Используя условие нормировки, получим

$$\frac{\pi_{(0,\,0)}(n)}{2}\sum_{i=0}^{m}\rho(n)^i(2^iW+1) = 1.$$

Пусть $\tau(n)$ — вероятность того, что станция попытается передать кадр в течение произвольно выбранного слота. На основе стационарных вероятностей получена система уравнений, связывающая $\tau(n)$ и $\rho(n)$:

$$\begin{cases} \tau(n) = \dfrac{2\sum\limits_{i=0}^{m}\rho(n)^i}{\sum\limits_{i=0}^{m}\rho(n)^i(2^iW+1)}; \\ \rho(n) = 1 - (1-\tau(n))^{n-1}. \end{cases}$$

Эта система может быть решена численно.

1.5. Стационарное распределение полумарковского процесса

Мы предполагаем известным распределение интервалов времени между переходами. Так как максимальная длина передаваемого кадра зафиксирована стандартом и в течение одного временного слота система переходит в новое состояние, то среднее время пребывания процесса в состояниях ограничено сверху. Вместе с тем, процесс регулярен в силу того, что любая пара переходов обязательно содержит переход, время которого ограничено снизу положительной константой. Вложенная цепь Маркова положительно возвратна. Следовательно, полумарковский процесс является эргодическим, и, зная распределение длительностей интервалов времени на переходах вложенной цепи, можно определить стационарные вероятности пребывания процесса на переходах.

Пусть E_s — средняя длительность слота успешной передачи, E_c — средняя длительность коллизионного слота, E_i — длительность пустого слота. Значение $E_i = \sigma$ зафиксировано стандартом и зависит от конкретной технологии, используемой на физическом уровне. Стандарт определяет также следующие параметры: H — длительность передачи заголовка кадра данных, длительности передачи кадров RTS, CTS, ACK (см. п. 1.2), длины межкадровых промежутков *SIFS, DIFS, EIFS* [1].

SIFS — короткий межкадровый промежуток, выдерживаемый станцией-получателем перед передачей кадров CTS, ACK, а также станцией-отправителем перед передачей кадра данных при получении кадра CTS.

DIFS — распределенный межкадровый промежуток, выдерживаемый каждой станцией перед возобновлением процесса изменения счетчика отсрочки после того, как канал опознается как свободный.

EIFS — расширенный межкадровый промежуток, выдерживаемый вовлеченной в коллизию станцией перед возобновлением процесса изменения счетчика отсрочки.

Пусть v_x — вероятность появления кадра с полем данных, длительность передачи которого равна x. Тогда средняя длительность слота успешной передачи для метода передачи "basic access" определяется как

$$E_s = H + \sum_{x = x_{\min}}^{x_{\max}} x v_x + \text{SIFS} + \sigma + \text{ACK} + \text{DIFS} + \sigma,$$

а для метода "RTS/CTS"

$$E_s = \text{RTS} + 3\text{SIFS} + 4\sigma + \sum_{x = x_{\min}}^{x_{\max}} x v_x + \text{SIFS} + \\ + \sigma + \text{ACK} + \text{DIFS} + \text{CTS} + \text{H}.$$

Длительность коллизионного слота для метода "basic access" вычисляется как

$$E_c = H + \sum_{x = x_{\min}}^{x_{\max}} x v_x + \text{DIFS} + \sigma,$$

а для метода "RTS/CTS"

$$E_c = \text{RTS} + \text{DIFS} + \sigma.$$

Пусть $E_b(n)$ — средняя длительность отложенного слота, в течение которого станция не предпринимает попыток передачи, поскольку связанный с ней счетчик отсрочки не равен нулю. Отложенный слот может быть пустым, если ни одна из оставшихся станций сегмента не передает, успешным, если передает только одна станция, и коллизионным, если станций, осуществляющих передачу в течение слота, несколько. Поэтому имеем

$$E_b(n) = (1 - \tau(n))^n E_i + n\tau(n)(1 - \tau(n))^{n-1} E_s + \\ + (1 - (1 - \tau(n))^n - n\tau(n)(1 - \tau(n))^{n-1}) E_c.$$

Пусть $E_t(n)$ — средняя длительность слота передачи. С вероятностью $\rho(n)$ попытка передачи приводит к коллизии, в противном случае передача осуществляется успешно, т. е.

$$E_t(n) = (1 - \rho(n)) E_s + \rho(n) E_c.$$

Пусть $g_{(i,j),(h,k)}$ — средняя длительность перехода $(i, j) \to (h, k)$, равная E_s, $E_b(n)$ или E_c в зависимости от того, является ли соответствующий слот успешным, коллизионным или отложенным. Тогда выражение для стационарных вероятностей полумарковского процесса будет иметь вид

$$f_{(i,j),(h,k)} = \frac{\pi_{i,j} P\{(h, k), (i, j)\} g_{(i,j),(h,k)}}{(1 - \tau(n)) E_b(n) + \tau(n) E_t(n)}. \qquad (1)$$

Зная долю времени, в течение которого система пребывает в состоянии (i, j) при условии, что следующий переход будет проведен в состояние (h, k), мы можем получить долю времени, в течение которого станция осуществляет передачу:

$$T_t(n) = \frac{\tau(n) E_t(n)}{(1 - \tau(n)) E_b(n) + \tau(n) E_t(n)}.$$

2. Характеристики производительности сегмента беспроводной сети

На основе стационарных вероятностей полумарковского процесса (1) получим выражения для основных характеристик производительности метода множественного доступа, а именно, для пропускной способности в состоянии насыщения и средней задержки обработки кадра, вовлеченного в соревновательный цикл.

2.1. Пропускная способность в состоянии насыщения

Нормализованной пропускной способностью в состоянии насыщения $T(n)$ назовем долю времени, в течение которой канал используется для успешной передачи полезных данных всеми станциями сегмента. (Заметим, что в данном случае под полезными данными будут пониматься сообщения вышележащих уровней, в том числе заголовки.)

Пусть E_p — средняя длина поля данных кадра. Тогда $T(n)$ может быть выражена как

$$T(n) = \frac{n\tau(n)(1 - \rho(n)) E_p}{(1 - \tau(n)) E_b(n) + \tau(n) E_t(n)}.$$

Для того чтобы сравнить результаты численных расчетов на основе полумарковской модели с полученными в [2], мы используем те же системные параметры и предполагаем одинаковый размер всех кадров (длина поля данных 8184 бит).

Рис. 2 и 3 демонстрируют нормализованную пропускную способность насыщения, достижимую в сегменте для РКФ 802.11, соответственно при использовании метода доступа "basic access" и "RTS/CTS". Заметим, что в случае метода "basic access" разница между результатами анализа марковской и полумарковской моделей составляет около 10 %.

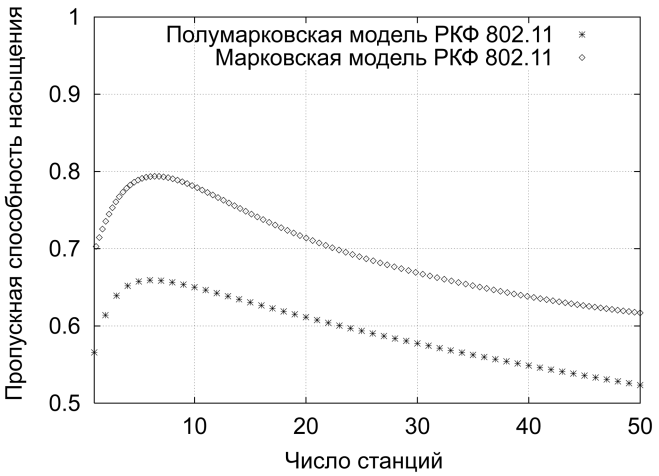


Рис. 2. Пропускная способность насыщения для метода "basic access"

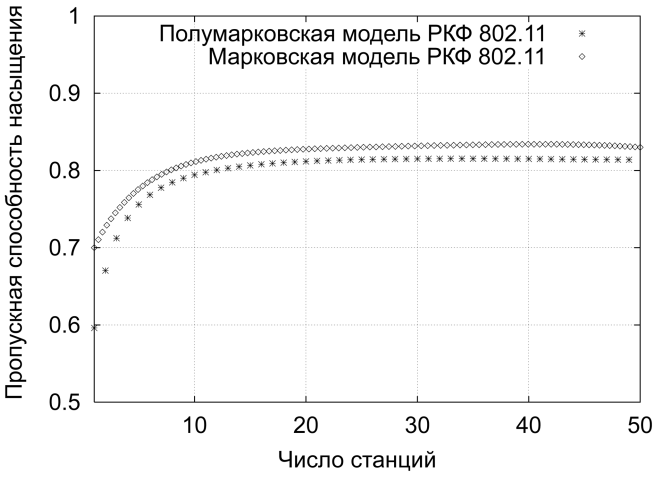


Рис. 3. Пропускная способность насыщения для метода "RTS/CTS"

2.2. Средняя задержка обработки кадра станцией

В отличие от [2] наша модель позволяет легко получить важную с практической точки зрения метрику производительности сегмента беспроводной сети — оценку задержки обработки кадра станцией.

Задержкой обработки будем считать среднюю длительность интервала времени, предшествующего успешной передаче кадра. При этом будем предполагать, что кадр, израсходовавший все попытки передачи и перешедший в состояние "Fail", немедленно переходит на новый соревновательный цикл.

Утверждение. Средняя задержка обработки кадра $E_{delay}(n)$ определяется выражением

$$E_{delay}(n) = (1 - \rho(n)^{m-1})E_s +$$

$$+ \sum_{r=1}^{m} \left(\frac{1}{2^r W} E_c + \frac{2^r W - 1}{2} E_i \right) \rho(n)^r (1 - \rho(n)^{m-r}) \times$$

$$\times \left((1 - \rho(n)^m) + m \frac{\rho(n)^{2m-1}}{1 - \rho(n)^m} \right).$$

Доказательство. Пусть $E_{cycle}(n)$ — средняя длина соревновательного цикла, а $E_{ncycles}(n)$ — среднее число циклов, предшествующих передаче. Тогда средняя задержка кадра может быть вычислена как

$$E_{delay}(n) = E_{cycle}(n) E_{n\text{cycles}}(n).$$

Средняя длина цикла $E_{cycle}(n)$ может быть вычислена как среднее время возвращения в объединенное состояние "Fail-Success".

В силу структуры полумарковского процесса $E_{cycle}(n)$ имеет вид

$$E_{cycle}(n) =$$

$$= \sum_{r=1}^{m} \left(\sum_{k-1}^{r} (\mu_{k-1,k} + E_t) \right) \rho(n)^{r-1} (1 - \rho(n)),$$

где $\mu_{k-1,k}$ — среднее время прохождения уровня k соревновательного цикла:

$$\mu_{k-1,k} = \frac{\rho(n)}{2^k W} E_c + \frac{\rho(n)}{2^k W} E_b (1 + 2 + \ldots +$$

$$+ 2^k W - 1) = \frac{\rho(n)}{2^k W} E_c + \frac{\rho(n)(2^k W - 1)}{2} E_b.$$

Возвращаясь к $E_{cycle}(n)$, найдем

$$E_{cycle}(n) = (1 - \rho(n)^{m-1}) E_s +$$

$$+ \sum_{r=1}^{m} \left(\frac{1}{2^r W} E_c + \frac{2^r W - 1}{2} E_i \right) \rho(n)^r (1 - \rho(n)^{m-r}).$$

Получим $E_{ncycles}(n)$ как среднее число возвращений в состояние "Fail", минуя состояние "Success":

$$E_{ncycles}(n) = (1 - \rho(n)^m) \left(1 + \sum_{i=1}^{\infty} i (\rho(n)^m)^i \right) =$$

$$= \left((1 - \rho(n)^m) + m \frac{\rho(n)^{2m-1}}{1 - \rho(n)^m} \right).$$

На рис. 4 и 5 проиллюстрирована зависимость задержки обработки от числа станций в сегменте в случае, когда сегмент беспроводной сети работает в режиме насыщения (при тех же системных параметрах, что и в примере для пропускной способности из п. 2.1).

3. Измерительные эксперименты

Для проверки адекватности модели были проведены измерения трафика в беспроводном сегменте. Было задействовано 12 станций, работающих под управлением ОС Linux в режиме "точка—точка". На одной из станций с помощью сервера kismet [4] фиксировались передачи всех, в том числе служебных, кадров в сегменте. Одна станция работала на прием и не формировала никакого трафика, за исключением определенного стандартом 802.11 кадра подтверждения. В серии испытаний 1—10 станций работали только на передачу, нагружая сегмент в течение пятиминутного периода. Кадры измеряемого трафика генерировались утилитой iperf [5] со

скоростью, обеспечивающей режим работы сегмента, близкий к насыщению. Системные параметры работы сегмента были установлены такими же, как и принятые в расчетах п. 2.1. В экспериментах был использован режим <basic access>. На рис. 6 представлены результаты экспериментального замера пропускной способности в сравнении с результатами моделирования.

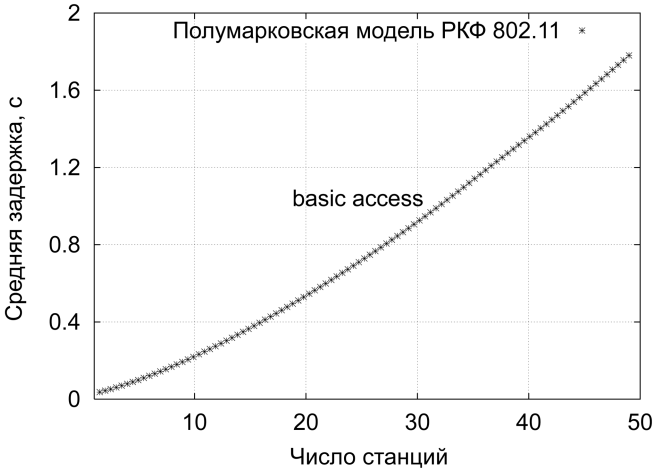


Рис. 4. Задержка обработки в режиме насыщения для метода "basic access"

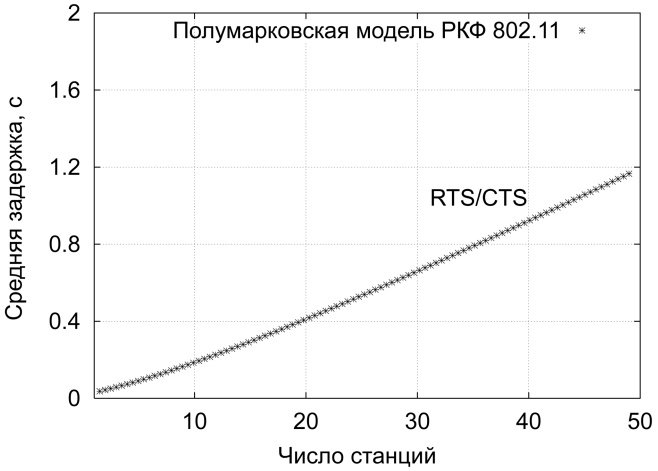


Рис. 5. Задержка обработки в режиме насыщения для метода "RTS/CTS"

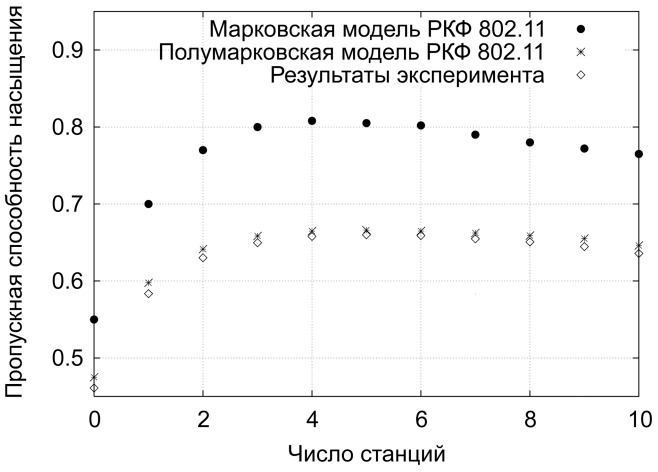


Рис. 6. Сравнение результатов исследования модели с экспериментальными данными

Анализ показал удовлетворительное совпадение полумарковской модели с экспериментальными данными (отличие в пределах 2—3 %). В то же время марковская модель [2] существенно переоценивает пропускную способность, поскольку не принимает во внимание разницу в длительности временных слотов, в частности, очень длинных коллизионных слотов, характерных для проведенных нами измерений.

В будущем для более глубокого исследования адекватности модели мы планируем провести серию измерительных экспериментов и выполнить анализ с помощью статистических критериев.

Заключение

В работе рассмотрен процесс захвата среды станцией в сегменте беспроводной сети 802.11. Построена и исследована модель процесса захвата в виде двумерного полумарковского процесса, для которого получено стационарное распределение вероятностей. На его основе получены выражения для характеристик производительности сегмента беспроводной сети: нормализованной пропускной способности и задержки обработки кадра станцией в соревновательном процессе. Использование полумарковского процесса позволило существенно уточнить модель [2] и адекватно предсказывать изменение производительности сегмента беспроводной сети в период пиковой нагрузки в зависимости от числа передающих станций. Как видно из графиков на рис. 2—5, пропускная способность насыщенного сегмента с ростом числа станций резко падает, а задержка резко возрастает.

Предлагаемый подход можно использовать для построения более общих моделей, позволяющих оценивать эффективность координационных функций, отличных от определенных стандартом. Полученная модель позволит построить практически важные алгоритмы, выявляющие приближение беспроводного сегмента к состоянию насыщения в реальном масштабе времени.

Список литературы

1. **IEEE,** wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, IEEE 802.11 standards, 1999—2003.

2. **Bianchi G.** Performance Analysis of the IEEE 802.11 Distributed Coordination Function // IEEE J. on Selected Areas in Communication. March 2000. Vol. 18. N 3. P. 535—547.

3. **Wu H., Peng Y., Long K., Cheng S., Ma J**. Performance of Reliable Transport Protocol over IEEE Wireless LAN: Analysis and Enhancement // Proc. IEEE ICC 2002. 2002. P. 599—607.

4. **Kismet** — an 802.11 layer 2 wireless network detector, sniffer, and intrusion detection system. http://www.kismetwireless.net/

5. **Iperf** — the TCP/UDP Bandwidth Measurement Tool. http://dast.nlanr.net/Projects/Iperf/

УДК 004.72

А. В. Першин,
Санкт-Петербургский государственный электротехнический университет "ЛЭТИ" имени В. И. Ульянова (Ленина)

Архитектура настраиваемого агента

Обсуждаются способы построения системы мобильных агентов для решения задачи поиска в сети, определяются компоненты, задачи и функции системы мобильных агентов. Предлагается архитектура специализированного агента с использованием абстрактных шаблонов, допускающая для увеличения быстродействия адаптацию к состоянию сети благодаря сборке агента из готовых компонентов на этапе создания.

Большое число задач, решаемых с помощью сетевых распределенных систем, связано с поиском товаров и услуг, запрашиваемых потребителями. Данную задачу удобно определять в терминах поставщик-потребитель следующим образом: существует множество поставщиков товаров или услуг и множество потребителей, где потребители, согласно своим интересам, ищут нужного поставщика. Подобная постановка задачи актуальна для таких систем, как сети продажи товаров, системы планирования перевозок, системы резервирования мест в гостиницах или бронирования билетов. Традиционная архитектура таких систем является централизованной, объединяющей данные всех поставщиков в единой базе данных, что требует значительных ресурсов. Альтернативой являются децентрализованные системы, например системы, построенные с использованием мобильных агентов [1], позволяющие достичь большей эффективности благодаря уменьшению объема передаваемой информации и увеличению степени параллелизма, что снижает требования к производительности компьютеров сети.

Для построения мобильных агентов можно использовать абстрактные шаблоны [2], классифицируемые следующим образом:

Управление перемещением [3, 4]. Паттерны перемещения позволяют при разработке агента выделить в отдельный компонент такие аспекты управления перемещением, как представление маршрута агента, правила обхода заданного маршрута, действия в случае недоступности хоста в маршруте, организовать обход одного маршрута несколькими агентами и т. д.

Обеспечение взаимодействия между агентами [3—5]. Данные шаблоны позволяют выделить различные способы доставки сообщений агенту, например, с помощью агента-курьера, с использованием "доски объявлений", организацией "встречи" агентов в определенном месте в определенное время для обмена данными и т. д.

Построение иерархии агентов [3, 6, 8]. Такие шаблоны позволяют создать иерархическую организацию агентов, где вышестоящие агенты могут управлять нижестоящими. Это дает возможность распределить подзадачи между агентами нижнего уровня и обеспечить параллельную обработку.

Организация решения задачи [2, 3, 5, 9, 10]. Данные шаблоны имеют отношение к решению конкретной задачи, поставленной перед агентом или группой агентов, описывают методы реализации бизнес-логики агента.

Обеспечение надежности [11—13]. Шаблоны обеспечивают возможность построения агентских систем с учетом повышенных требований к надежности. Позволяют механизм обеспечения надежности сделать прозрачным для агента [9] [10]. При написании алгоритма работы агента не требуется учитывать аспекты надежности. Основной задачей является обнаружение отказа агента и мониторинг его перемещения [11].

Эффективность использования шаблона для реализации функций агента зависит от специфики решаемой задачи, архитектуры и требований к системе. Выбор реализации должен выполняться на этапе создания агента с учетом особенностей решаемой задачи, загрузки сети и узлов поставщиков.

Архитектура мобильного агента зависит от способа взаимодействия агентов при выполнении поиска и архитектуры узлов поисковой системы. Возможны следующие варианты взаимодействия: 1) агенты независимы, т. е. не взаимодействуют друг с другом; 2) агенты взаимодействуют друг с другом, но иерархия между агентами отсутствует; 3) агенты разделяются на поисковых и управляющих, так называемых агентов-супервизоров, группа поисковых агентов управляется агентом-супервизором. Любая из этих архитектур возможна, и она будет определена в момент запуска агентов в зависимости от параметров запроса потребителя, ситуации в сети и возможно других факторов.

Архитектура поиска с мобильными агентами, где агенты независимы и не обмениваются сообщениями между собой, показана на рис. 1. Сплошными линиями показан обмен между потребителями и сервером-интерфейсом, штриховыми линиями — перемещения поискового агента.

При поиске в сети с использованием мобильных агентов важным является решение задачи распределения нагрузки между агентами. Эту задачу можно решить, используя агентов-маршрутизаторов. На рис. 2 приведена схема архитектуры системы мобильных агентов с использованием агента-маршрутизатора (на схеме он обозначается номером "1"). Поиск в такой системе включает следующие этапы: *а* — потребитель посылает запрос на сервер-интерфейс; *б* — сервер-интерфейс создает аген-

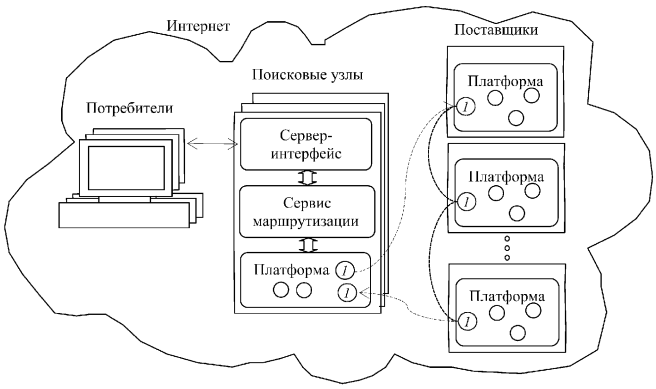


Рис. 1. Архитектура поиска с мобильными агентами

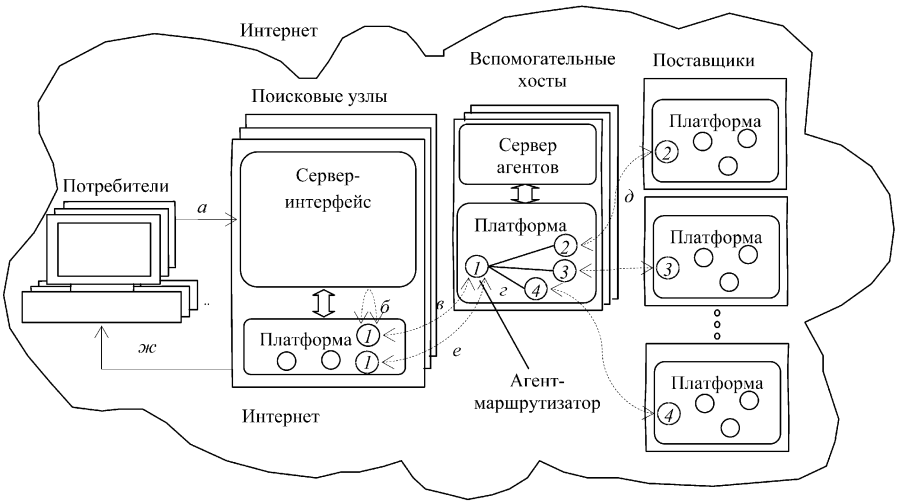


Рис. 2. Архитектура поиска с мобильными агентами-маршрутизаторами

та-маршрутизатора; *в* — агент-маршрутизатор перемещается на вспомогательный хост сети и, согласно запросу пользователя, определяет подмножество поставщиков для обхода; *г* — агент-маршрутизатор создает несколько поисковых агентов для обхода нужных поставщиков, задает им маршруты и запускает их; *д* — поисковые агенты обходят заданных поставщиков и возвращаются с результатом; *е* — агент-маршрутизатор возвращается с результатом на сервер-интерфейс; *ж* — сервер-интерфейс формирует и передает результат потребителю.

Платформа системы мобильных агентов (СМА) [1] обеспечивает взаимодействие агента и операционной системы хоста, среду выполнения агентов. Структура системы мобильных агентов для поиска (рис. 3) разбивается на ярусы согласно выполняемым функциям:

Ярус 1. Функции интерфейса с потребителями: получение запроса; выдача ответа.

Ярус 2. Создание серверов агентов и координация агентов с дополнительными службами. Ярус также включает вспомогательные службы: маршрутизацию (формирование списка узлов); мониторинг (наблюдение за статусом агентов); восстановление агентов после сбоя.

Ярус 3. Серверы агентов на вспомогательных хостах: конструирование агентов, мониторинг, управление группой поисковых агентов, обмен данными между агентами и группами агентов, балансировка нагрузки.

Ярус 4. Поисковые агенты: анализ информации поставщика.

Путь от момента получения поискового запроса до создания и запуска агента, показанный на рис. 3 стрелками:

1 — поступление поискового запроса на сервер-интерфейс;

2 — формирование списка серверов для обхода и выполнения локального поиска;

3 — получение информации о загрузке узлов и сети;

4 — создание сервера агентов и передача ему запроса;

5 — определение состава агента и его сборка на основе библиотеки компонентов, определение числа поисковых агентов. Запуск агентов.

Серверы агентов (ярус 3) и создаются с целью увеличения параллелизма для распределения нагрузки. Состав агента и тип компонентов определяются на основе данных службы мониторинга. Параметрами, определяющими состав агента, являются выбранная архитектура (группа независимых агентов, группа агентов с супервизором, иерархическая организация агентов и т. д.) и метод балансировки нагрузки [14], [15]: статическая балансировка (маршруты агентов рассчитываются перед запуском поиска), динамическая (распределение хостов по агентам происходит в процессе выполнения поиска, по мере освобождения того или иного агента). Выбор архитектуры определяется параметрами запроса (последовательный поиск, поиск первого подходящего решения и т. д.), рекомендациями и зависимостями, определенными в процессе имитационного моделирования, а также состоянием сети. Критерием выбора архитектуры и состава агента, числа поисковых агентов является минимальное время поиска.

Вспомогательные хосты служат для обеспечения вычислительными ресурсами агента-маршрутизатора. Маршрутизация проводится в два этапа: 1) определение списка поставщиков для обхода (служба маршрутизации); 2) распределение поставщиков среди поисковых агентов (сервер агентов). Сервер агентов является узким местом в системе, так как через него идет обмен сообщениями всех поисковых агентов, что может отрицательно сказаться на производительности. Для решения этой проблемы можно создать систему из нескольких

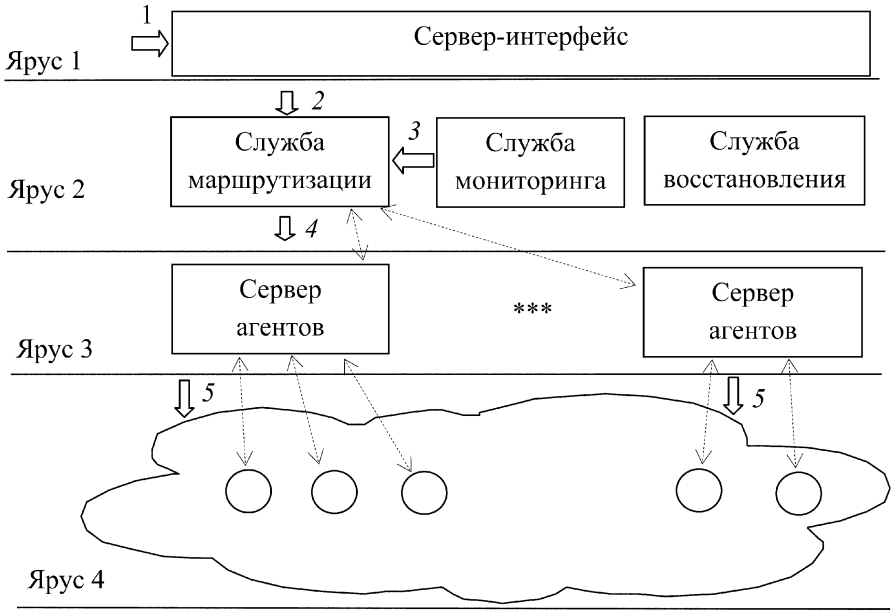


Рис. 3. Структура системы мобильных агентов для поиска

мобильных агентов-маршрутизаторов, а поисковых агентов разделить на группы. Каждому агенту-маршрутизатору делегируется управление определенной группой агентов. Это позволит перераспределить нагрузку между агентами-маршрутизаторами, так как они будут находиться на разных хостах и обслуживать только часть агентов. Это потребует дополнительной функциональности в агентах: управления иерархией, групповой конфигурации.

В результате сбоев, возникающих в сетях, возможны отказы в доступе как к отдельным серверам, так и к целым сегментам сети. В случае таких сбоев возможны потери агентов, которые могут привести к невозможности выполнить поисковый запрос. Особо чувствительным элементом является агент-маршрутизатор, поэтому стоит обратить внимание на методы, повышающие надежность системы поиска. Основным методом увеличения надежности является резервирование. Применительно к рассматриваемой задаче целесообразно резервировать агента-маршрутизатора. Для обнаружения потери агента может использоваться регулярный опрос. Эта задача возлагается на службу мониторинга и восстановления, обеспечивающую обнаружение отказов и восстановление агентов после сбоя, что потребует дополнительных функций агента, связанных с взаимодействием со службой восстановления агентов.

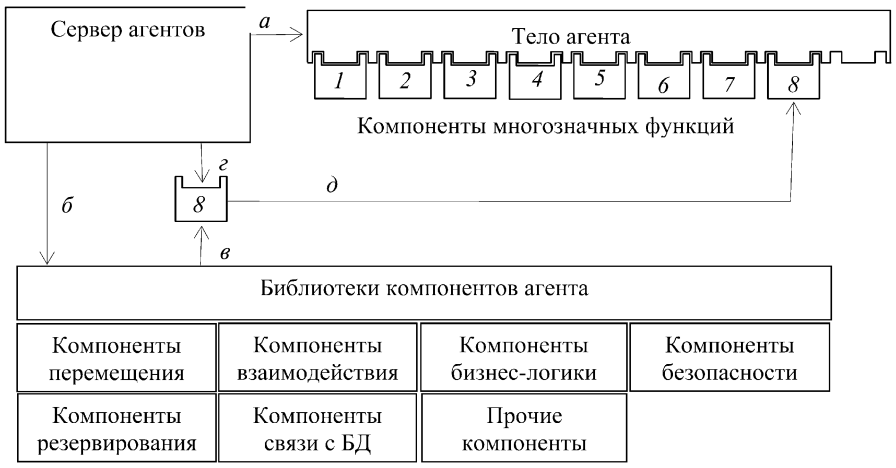


Рис. 4. Сборка агента

Набор функций агента будет определяться архитектурой системы поиска, набором служб и функций агентов. Выбор оптимальной архитектуры системы определяется взаимно влияющими факторами: интенсивностью запросов потребителей; числом поисковых агентов, создаваемых на каждый запрос пользователя; загрузкой сети, серверов и вспомогательных хостов.

Таким образом, набор функций и их реализация могут быть различными в зависимости от архитектуры. Агент, реализующий полный набор функций, неэффективен, так как он будет содержать избыточную функциональность, которая увеличивает размер агента, снижает быстродействие и увеличивает трафик в сети. В связи с этим рассмотрим архитектуру агента, позволяющую собирать специализированного агента из готовых компонентов, конфигурируемых в процессе создания. Поскольку критерием выбора архитектуры агента является текущая ситуация в сети, то архитектура агента должна позволять выбирать способы реализации требуемых функций на этапе создания или инициализации агента. В этом случае сборку агента можно представить в виде схемы (рис. 4). Агент будет состоять из "тела" и компонентов. Тело агента будет управлять инициализацией и завершением, обеспечивать возможность динамической стыковки компонентов, их взаимодействие, возможность сохранения и восстановления состояния компонентов. Сборка агентов производится с помощью сервера агентов, последовательность сборки показана стрелками: *а* — создание тела агента; *б* — запрос класса функционального элемента из библиотеки; *в* — создание компонента; *г* — инициализация и настройка компонента; *д* — интеграция компонента в тело агента.

К телу агента могут подключаться следующие компоненты:

1 — компонент перемещения — управляет перемещением агента, содержит маршрут, текущее положение агента в сети и т. д.;

2 — компонент бизнес-логики — обеспечивает поведение агента, определяемое решаемой задачей, определяет функции поиска агента (бизнес-функции агента);

3 — компонент взаимодействия с другими агентами — определяет протокол взаимодействия между агентами в группе (ACL, KQML, бинарный формат);

4 — компонент взаимодействия с системой локального поиска (компонент связи с БД) — определяет протокол взаимодействия с базой данных, т. е. является интерфейсом к базе данных;

5 — компонент взаимодействия с сервером-интерфейсом — определяет протокол взаимодействия между сервером-интерфейсом и агентом для обеспечения возможности передачи результатов работы и получения команд управления; взаимодействие может быть как локальным, так и удаленным;

6 — компонент резервирования — позволяет создать резервных агентов, обнаружить потерю резервируемого агента и заменить его резервным в случае потери. Для обеспечения этих функций осуществляется информационный обмен между резервируемым и резервными агентами, в результате чего повышается надежность агента-маршрутизатора в ненадежной сетевой среде.

7 — компонент безопасности — отслеживает изменение состояния агента, подлинность сертификатов, цифровых подписей от внутренних данных и кода агента, а также сообщений, принимаемых и отсылаемых агентом, взаимодействует с компонентами безопасности платформы СМА;

8 — компонент взаимодействия с платформой СМА — обеспечивает возможность взаимодействия агента и платформы СМА. Внутреннее состояние агента и его компонентов сохраняются с помощью платформы СМА, платформа также обеспечивает передачу сообщений между агентами, управление инициализацией и завершением агента, поэтому следует выделить отдельный интерфейс для взаимодействия агента и платформы.

Все компоненты могут настраиваться, т. е. могут быть заменены на другой компонент, иначе реализующий ту же функциональность.

Базовый агент для выполнения операций сетевого поиска должен иметь по крайней мере четыре компонента: 1) компонент перемещения, чтобы перемещаться на сервер поставщика; 2) компонент взаимодействия с БД поставщика, чтобы проводить локальный поиск; 3) компонент взаимодействия с маршрутизатором; 4) компонент взаимодействия с платформой СМА.

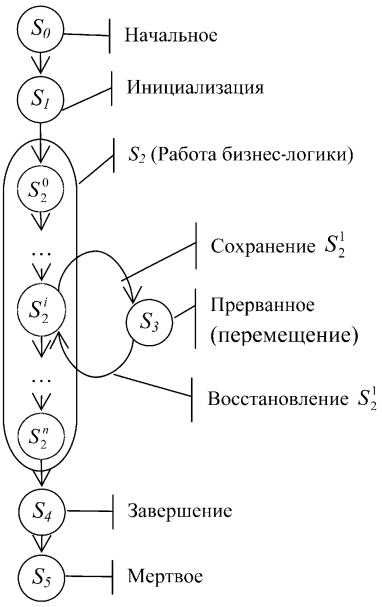


Рис. 5. Состояния мобильного агента

Тело агента должно обеспечивать возможность взаимодействия между компонентами произвольной функциональности. Для реализации такой архитектуры потребуется общий интерфейс для передачи межкомпонентных сообщений и обеспечения возможности сохранения и восстановления состояния компонента в процессе перемещения. Этот процесс можно пояснить с помощью обобщенной диаграммы состояний мобильного агента (рис. 5), где выделены следующие основные состояния: инициализация – S_1, работа – S_2, перемещение – S_3, завершение — S_4. Когда агент находится в рабочем состоянии, он выполняет поиск на хосте поставщика. Рабочее состояние агента представляется подсостоянием его бизнес-логики $\{S_2^0, S_2^1, ..., S_2^n\}$, которое должно сохраняться и восстанавливаться после перемещения, этот процесс обозначается событиями: сохранение S_2^i, восстановление S_2^i, где i — означает i-е состояние бизнес-логики.

При перемещении агента должны передаваться не только данные о состоянии, но и код агента. Под состоянием агента подразумевается совокупность атрибутов агента, определяющих его дальнейшее поведение (например, информация об индексе текущего поставщика в маршруте, адреса сервера агентов, истории перемещений и др.). В данной архитектуре состояние агента складывается из совокупности атрибутов модулей. При сборке агента из компонентов состав агента меняется, следовательно, сервер агентов должен отбирать компоненты, которые нужно присоединять к агенту. Классы агента одинаковой функциональности, обеспечивающие разную реализацию, могут находиться в разных каталогах, разных пакетах, загружаться удаленно по сети — любой из этих подходов реализуем в современных языках программирования, например в языке Java.

На рис. 6 показана UML-диаграмма, обеспечивающая подключение компонентов перестраиваемого агента. Состояние агента и его компонентов сохраняется в контейнере **Storage**, который может быть сохранен на диске или передан по сети. Класс **AgentServer** является сервером, конструирующим агентов. Класс **Agent** определяет базовую функциональность тела агента. Интерфейс **ComponentsInterract** обеспечивает функциональность для конструирования агентов из компонентов. Класс **Component** описывает универсальный интерфейс компонента агента, который позволяет обмениваться сообщениями между компонентами.

На базе классов **Agent** и **Component** должны быть созданы классы агентов (**ConcreteAgent**) и компонентов (**ConcreteComponent**), содержащие функциональность для решения конкретных задач путем реализации перечисленных абстрактных методов. Последовательность операций при создании и подключении компонента к телу агента показана на рис. 7.

В процессе создания агента происходит создание и добавление компонента в тело агента — addComponents. При инициализации агента его атрибуты устанавливаются в начальное состояние — Agent.init. Перед перемещением атрибуты агента сохраняются в контейнере Storage, а после перемещения восстанавливаются — Agent.load, затем происходит инициализация компонентов, заключающаяся в связи агента и его компонентов, — setAgent, получение компонентами информации друг о друге — getComponentsInfo, восстановление состояний компонентов — Component.load.

Формальная спецификация взаимодействия компонентов обеспечивает эффективное повторное использование уже имеющейся функциональности благодаря разделению функций агента по компонентам, что позволяет конструировать агентов разной функциональности из готовых блоков. Взаимодействие между компонентами и телом агента осуществляется только с помощью множества заранее заданных сообщений, которые определяют межкомпонентный интерфейс высокого уровня.

В заключение отметим, что ключевыми моментами предлагаемой архитектуры являются:

- использование иерархической многоуровневой системы, что обеспечивает высокую степень параллелизма;
- применение абстрактных шаблонов для построения мобильных агентов;
- разделение базовой функциональности и компонентов расширения агента;
- адаптация архитектуры агентов к ситуации в сети благодаря сборке агента из готовых компонентов на

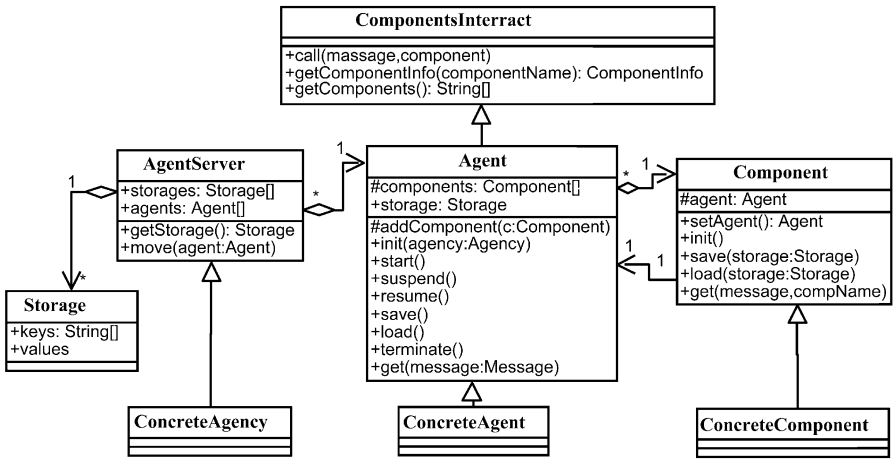


Рис. 6. Диаграмма классов перестраиваемого агента

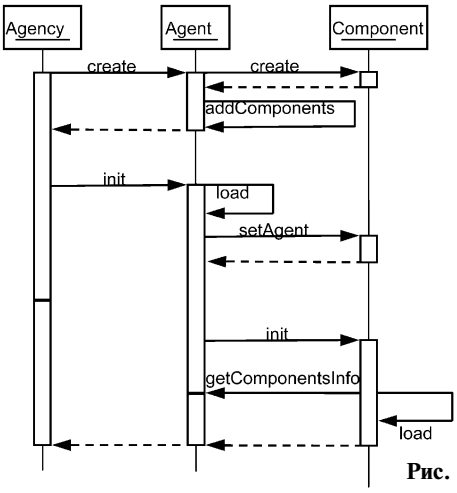


Рис. 7. Создание агента

этапе создания, что обеспечивает возможность изменения организации взаимодействия поисковых агентов для увеличения производительности.

Рассмотренная система имеет высокую степень параллелизма, которая должна обеспечивать сокращение времени поиска. Однако параллелизм не всегда приводит к сокращению времени поиска в загруженной сети с большим числом сообщений и не позволяет прогнозировать эту величину. Для получения приблизительных оценок целесообразно использовать имитационное моделирование работы системы мобильных агентов.

Список литературы

1. **CORBA** Facilities: Mobile Agent System Interoperability Facilities Submission. MASIF Revision. Object Management Group Framingham Corporate Center. 03/1998.

2 **Першин А. В.** Классификация шаблонов проектирования систем мобильных агентов // Сб. докл. Междунар. конф. по мягким вычислениям и измерениям. Санкт-Петербург, 17-19 июня 2004. СПб.: Изд-во СПбГЭТУ "ЛЭТИ", 2004. Т. 1. С. 238—243.

3. **Weiss M.** Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Cambridge, Massachusetts: MIT Press, 1999.

4. **Deugo D., Weiss M.** A Case for Mobile Agent Patterns // Workshop on Mobile Agents in the Context of Competition and Cooperation (MAC3). Autonomous Agents. 1999. P. 19—23.

5. **Emerson Ferreira de Araujo Lima.** Implementing Mobile Agent Design Patterns in the JADE framework, Software-Practice and Experience, 2003.

6. **Deugo D., Weiss M., Kendall E.** Reusable Patterns for Agent Coordination. Coordination of Internet Agents, Springer, 2001.

7. **Satoh I.** Hierarchically Structured Mobile Agents and their Migration. Department of Information Sciences, Ochanomizu University, Japan, 2000.

8. **Parunak H. Van Dyke, Odell J.** Representing Social Stuctures in UML // Autonomous Agents. 2001. Montreal, Canada, 2001.

9. **Jiang H., Vidal J. M.** From Rational to Emotional Agents // In AAAI Workshop on Auction Mechanism for Robot Coordination. 2006. July.

10. **Jiang H., Vidal J. M., Huhns M. N**. EBDI: An Architecture for Emotional Agents // In Proc. of the Autonomous Agents and Multi-Agent Systems Conf. 2007.

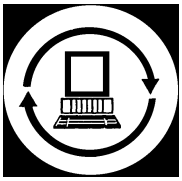
11. **Lyu Michael R., Wong Tsz Yeung.** A Progressive Pault Tolerant Mechanism in Mobile Agent Systems // Proc of 7th World Multiconference on Systemics, Cybernetics and Infomatics. SCI2003. 2003.

12. **Pleisch S., Schiper A.** Fault-Tolerant Mobile Agent Execution // IEEE Transactions on Computers. 2003. Vol. 52. N 2. P. 209—222.

13. **Pleisch S., Schiper A.** Modeling Fault-Tolerant Mobile Agent Execution as a Sequence of Agreement Problems // Proc. of 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00). 2000.

14. **New** Challenges in Dynamic Load Balancing / K. D. Devine, E. G. Boman, R. T. Heaphy et al. // Appl. Numer. Maths. 2005. Vol. 52. Issues 2—3. P. 133—152.

15. **Nehra N., Patel R. B., Bhat V. K.** A Framework for Distributed Dynamic Load Balancing in Heterogeneous Cluster // Journ. of Computer Science. 2007. P. 14—24.



ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ И УСТРОЙСТВА

УДК 004.272.42

Р. Н. Федюнин, ассистент,
Пензенский государственный университет

Устройство деления с настраиваемой логикой на базе однородных модулярно-конвейерных структур

Все позиционные системы счисления обладают одним существенным недостатком – выполнение арифметических операций зависит от скорости распространения межразрядных переносов. Этот недостаток отсутствует в системе счисления в остаточных классах. Представлен способ реализации конвейерного устройства деления в системе остаточных классов на базе однородных вычислительных структур, позволяющий более эффективно по сравнению с позиционными аналогами обрабатывать числа большой разрядности.

Существенным недостатком всех позиционных систем счисления является зависимость выполнения арифметических операций от скорости распространения межразрядных переносов. Этот недостаток отсутствует в системе счисления в остаточных классах (СОК) [1] К достоинствам СОК следует отнести:

- независимость образования разрядов числа, в силу чего каждый разряд несет информацию обо всем исходном числе, а не о промежуточ-

ном числе, получающемся в результате образования младших разрядов (как это имеет место в позиционной системе). В связи с этим появляется возможность независимой параллельной обработки разрядов, что позволяет использовать принципиально новые методы параллельно-конвейерной обработки информации;

- малоразрядность остатков, представляющих число, открывает возможности использования табличной арифметики.

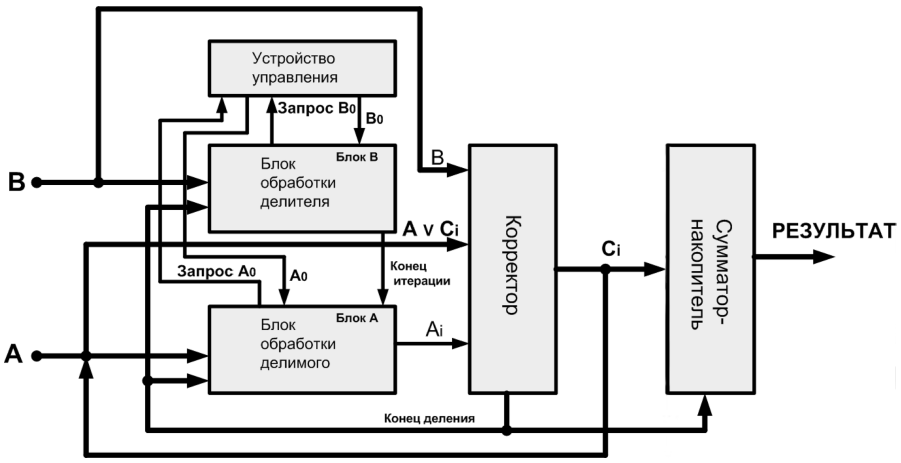


Рис. 1. Структурная схема устройства деления в СОК

В данной статье представлен способ реализации конвейерного устройства деления в системе остаточных классов на базе однородных вычислительных структур (ОВС) [2], позволяющий более эффективно обрабатывать числа большой разрядности (свыше 64 разрядов) по сравнению с позиционными аналогами [3].

Классический алгоритм общего деления в СОК [1] возможно реализовать в вычислительной структуре, обобщенная схема которой представлена на рис. 1.

В представленном на рис. 1 устройстве вычислительные блоки (*Блок A* и *Блок B*) являются набором матриц с клеточной топологией (рис. 2), в которых выполняется процесс обработки делимого и делителя по заданным модулям. Данные в блоки поступают по шинам данных A и B, разрядность которых определяется модулями, по которым работают вычислительные матрицы. Управляющий сигнал "Конец итерации" определяет зависимость вычислительного процесса в блоке A от результата вычислений в блоке B по алгоритму общего деления в системе остаточных классов [1]: как только результат в блоке B станет равным 1, в блок A требуется подать сигнал "Конец итерации".

Устройство управления предназначено для контроля вычислительного процесса в блоке B и блоке A и однозначного определения промежуточного результата деления [1], так как операция деления требует дополнительных характеристик числа для однозначного определения частного — *характер числа* и *след числа*. Данные характеристики в данном примере подробно не рассматриваются, предполагается, что они вычисляются устройством управления.

Корректор — устройство формирования промежуточного частного и анализа продолжения или окончания процесса деления. Как видно на рис. 1, шина данных промежуточного частного при условии, что деление не завершено, передает промежуточное частное в блок A для выполнения следующей итерации деления. Функционально корректор представляет собой набор сумматора-вычитателя и устройства умножения, которые подробно рассмотрены в [2—4].

Сумматор-накопитель складывает поступающие к нему промежуточные частные, тем самым накапливая конечный результат, который поступает на шину данных "РЕЗУЛЬТАТ" после подачи управляющего сигнала "Конец деления". Функционально сумматор-накопитель представляет собой матричный сумматор [3].

Блок *A* и блок *B* представляют собой наборы вычислительных модулей (рис. 2).

Каждый функциональный элемент однородной структуры представляет собой устройство, схема которого представлена на рис. 3.

Ячейка операционной части устройства деления (ЯОЧД) реализует следующую систему логических функций:

$$L\_OUT = L;$$

$$Q_1(t+1) = Q_1(t)\cdot \overline{L} \vee Q_1(t)\cdot DIV \vee A\cdot L\cdot \overline{DIV};$$

$$SH\_OUT = DIV\cdot A;$$

$$\begin{aligned} R = {} & CORR\cdot(\overline{A}\cdot Q_1(t)\cdot \overline{C} \vee A\cdot \overline{Q_1(t)}\cdot \overline{C} \vee \\ & \vee \overline{A}\cdot \overline{Q_1(t)}\cdot C \vee A\cdot Q_1(t)\cdot C) \vee CU\cdot(\overline{A}\cdot B\cdot \overline{C} \vee \\ & \vee A\cdot \overline{B}\cdot \overline{C} \vee \overline{A}\cdot \overline{B}\cdot C \vee A\cdot B\cdot C); \end{aligned} \quad (1)$$

$$\begin{aligned} Z = {} & CU\cdot(A\cdot B \vee A\cdot C \vee B\cdot C) \vee \\ & \vee CORR\cdot(\overline{A}\cdot B \vee \overline{A}\cdot C \vee B\cdot C); \end{aligned}$$

$$SO = R\cdot CORR \vee R\cdot CU \vee SH\cdot DIV;$$

$$P = Z,$$

где R и Z — соответственно значения сигналов, формируемых на выходах суммы и переноса сумматора; A, B, C — соответственно сигналы, подаваемые на одноименные A, B, C информационные входы ячейки ЯОЧД; L, SH, $CORR$, DIV, CU — управ-

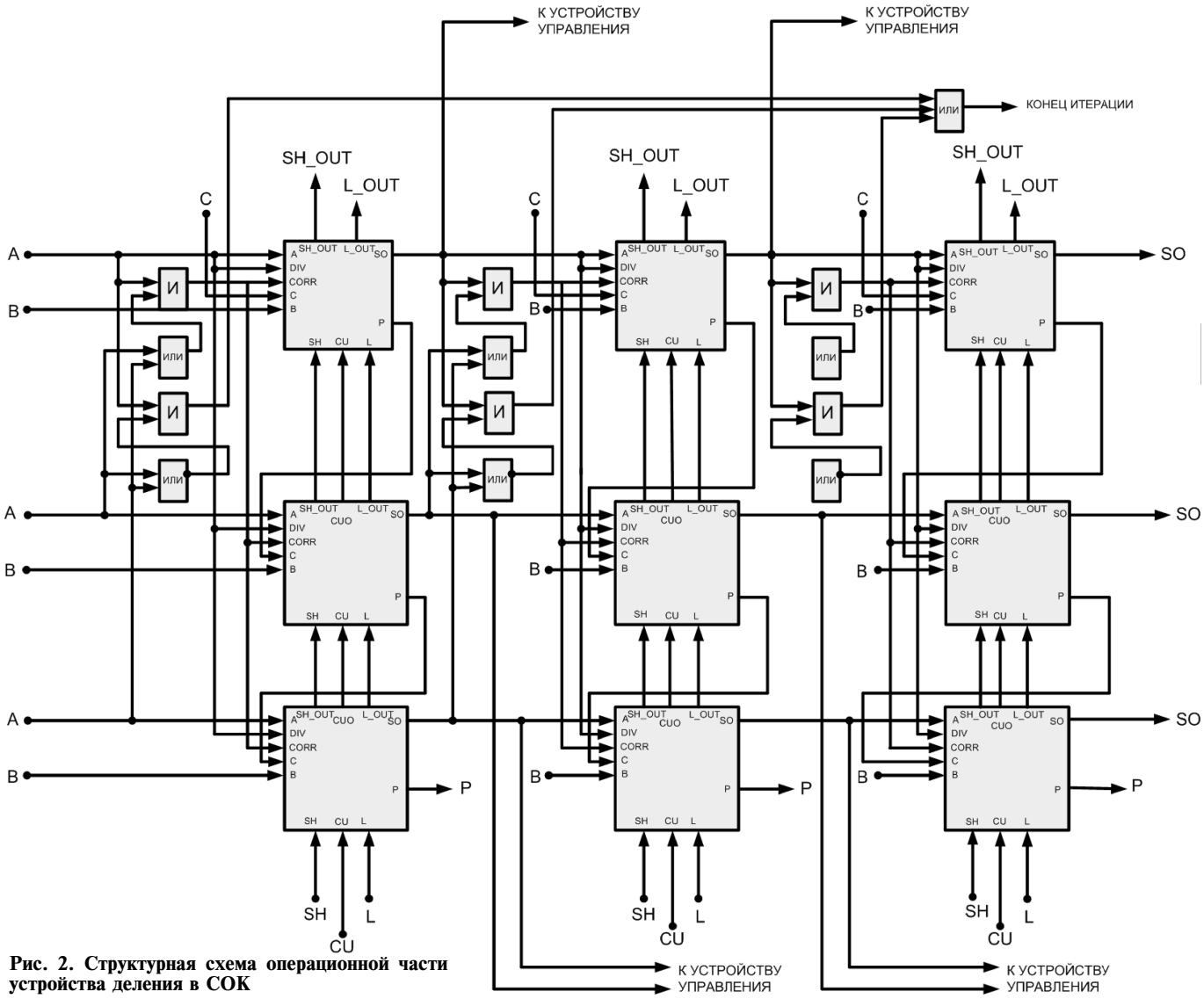


Рис. 2. Структурная схема операционной части устройства деления в СОК

ляющие сигналы, подаваемые на одноименные входы ячейки устройства деления, для выполнения соответствующих операций настройки и выполнения арифметических действий; $Q_1(t+1)$ — соответственно состояние выхода триггера для фиксации базового модуля; SO и P — информационные выходы ячейки устройства деления.

Основными режимами работы ОЧД структуры являются режим загрузки и режим выполнения арифметических операций.

Режим загрузки решает задачу загрузки в столбцы вычислительной структуры числа, биты которого размещаются в тригге-

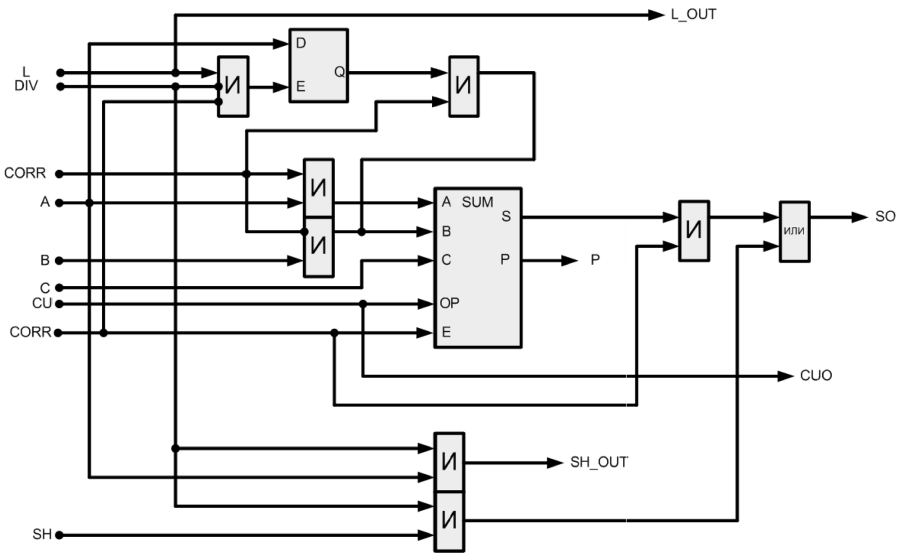


Рис. 3. Ячейка операционной части устройства деления в СОК

Таблица 1

Система логических функций, реализуемая ячейкой вычислительной структуры в ОВС

Код команды управления загрузкой			Система логических функций	
DIV	*CORR*	*L*	$Q_1(t+1)$	*L_OUT*
0	0	0	$Q_1(t)$	*L*
0	0	1	*A*	*L*
0	1	0	$Q_1(t)$	*L*
0	1	1	$Q_1(t)$	*L*
1	0	0	$Q_1(t)$	*L*
1	0	1	$Q_1(t)$	*L*
1	1	0	$Q_1(t)$	*L*
1	1	1	$Q_1(t)$	*L*

рах ячеек каждого из столбцов операционной части устройства деления (ОЧД). В рассматриваемом режиме работы на управляющие входы L_1—L_n ОЧД, где n — число столбцов в ОЧД, подается активный сигнал, равный "1". Данные, которые требуется загрузить в триггеры ячеек структуры, подаются на информационные входы A_1—A_n первой строки, причем самый младший бит подается на вход A первой ячейки первого столбца первой строки. В результате данные, поданные на группу входов A_1—A_n первой строки, считываются в триггеры ячеек структуры. Система логических функций, реализуемая каждой ячейкой вычислительной структуры в данном режиме, приведена в табл. 1.

Пример выполнения операции загрузки данных в триггеры ячеек ОЧД представлен на рис. 4.

В данном примере выполняется загрузка вектора 111_2 в ячейки ОЧД, для чего на входы A_1—A_n первой строки подается вектор данных, а на управляющих входах L_1—L_n сформировали активный сигнал "1", в результате данные с группы входов A_1—A_n первой строки были записаны в ячейки ОЧД.

В режиме реализации арифметических операций выполняется операция арифметического деления чисел по заданному модулю, при этом сле-

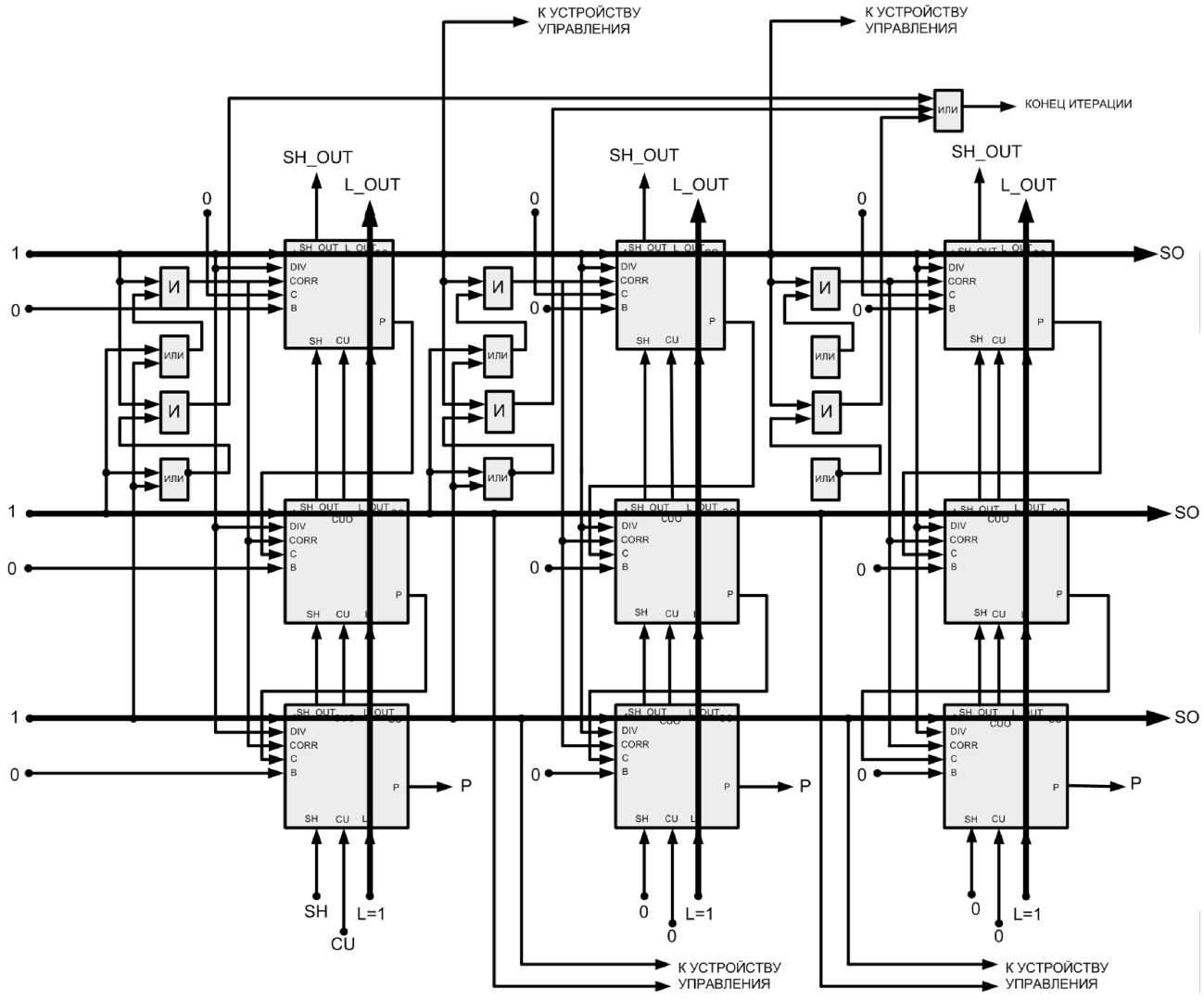


Рис. 4. Пример реализации операции загрузки данных в ОЧД

дует учесть, что предварительно ОЧД была настроена на работу по заданному модулю предыдущим режимом работы среды. Иными словами, проведена загрузка кода модуля во все ячейки ОЧД, с помощью которого будет осуществляться коррекция делимого. Специфика выполнения деления в СОК в том, что числа по конкретному модулю обрабатываются параллельно и отчасти независимо блоками ОЧД, описанными выше. Таким образом, вся операционная часть устройства деления в СОК представляет собой набор ОЧД, как показано на рис. 5.

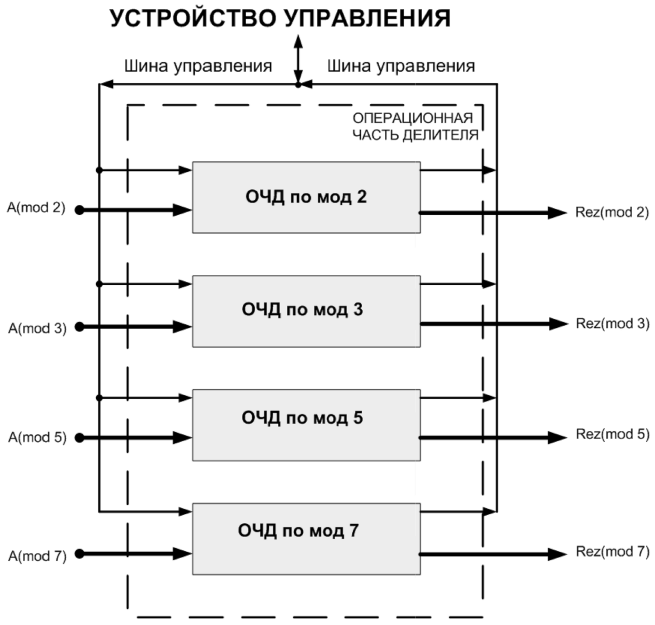


Рис. 5. Операционная часть устройства деления — Блок А и Блок В

Для работы в данном режиме на информационные входы A_1—A_n первого столбца ОЧД подается вычет [1] по соответствующему модулю. Если вычет по младшему модулю (в данном примере — по модулю два) равен 0, то в целом рассматриваемое число — четное и деление возможно, а значит, на управляющий вход DIV рассматриваемой ОЧД подается активный сигнал "1". Далее, логикой И-ИЛИ на входе каждого столбца ОЧД осуществляется проверка на четность конкретного вычета. Если он нечетный, о чем свидетельствует единица в самом младшем разряде вычета, подаваемого на вход *A* первой ячейки первого столбца первой строки, то вырабатывается управляющий сигнал коррекции *CORR* и данный вычет складывается с рабочим модулем, который хранится в триггерах столбца ОЧД. Полученные данные снова проходят проверку и, если они четные, то выполняется деление на два. В рассматриваемой реализации это осуществляется сдвигом по информационным каналам A-SH_OUT и SH-SO.

Полученный результат деления отправляется на входы *A* очередного столбца ОЧД и в устройство управления, которое по полученному результату определяет его характер [1] и след [1], а ис-

Таблица 2

Пример выполнения операции деления в СОК

Операция	Mod 2	Mod 3	Mod 5	Mod 7	Комментарий
Ввод *B* и анализ числа — коррекция	0 —	0010 —	001**1** 0011 + 0101	000**1** 0001 + 0111	Наличие нечетных чисел указывает на необходимость коррекции
Деление	0	0010	**1**000	**1**000	Число четное, среди вычетов по модулям нечетных чисел нет, выполняется операция деления
Коррекция	0	0001 0001 + 0011	0100	0100	
Деление	0	0100	0100	0100	
Деление	0	0010	0010	0010	
Конец операции деления *B*	**1**	**0001**	**0001**	**0001**	Все числа в *B* равны "1" — операция деления *B* завершена
Ввод *A* и анализ числа — коррекция	0	000**1** 0001 + 0011	0100	000**1** 0001 + 0111	Наличие нечетных чисел указывает на необходимость коррекции
Деление	0	0100	0100	1000	
Деление	0	0010	0010	0100	
Коррекция	0	000**1** 0001 + 0011	000**1** 0001 + 0101	0010	
Деление	0	0100	0110	0010	
Операция завершена	**0**	**0010**	**0011**	**0001**	Результат A_3 направляется в корректор

ходя из данных параметров — и четность результата. Если значение по модулю 2 будет равно 0, результат четный, иначе если результат по модулю 2 равен 1, то результат нечетный. Если результат четный, повторяем процедуры, описанные выше, иначе осуществляется коррекция числа — из него вычитается 1 для получения четного числа. Единица коррекции подается на входы B соответствующего столбца. Операция деления в блоке B продолжается до получения "1" в ОЧД по каждому модулю, что формирует управляющий сигнал "Конец итерации" и завершает очередной процесс деления в блоке A. Итерации деления в блоках A и B продолжаются до получения конечного результата, о чем сигнализирует сигнал управления корректора "Конец деления".

Рассмотрим пример реализации арифметического деления в системе остаточных классов. Модули, по которым выполняется операция деления, в данном случае соответственно равны $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$. Делимое $A = 64_{10} = (0, 1, 4, 1)$, делитель $B = 8_{10} = (0, 2, 3, 1)$. Поток данных и операций над данными в однородной среде приведен в табл. 2. Несмотря на то, что в таблице сначала рассматривается обработка числа B, а затем числа A, в устройстве, схема которого приведена на рис. 1, данные числа обрабатываются в блоке B и блоке A параллельно.

В корректоре по формуле $C_1 = A - B \cdot A_3$ [1] вычисляется промежуточное частное и, если оно равно нулю, то операция деления завершается формированием управляющего сигнала "Конец деления" и A_3 является конечным результатом, иначе C_1 по шине данных направляется на вход блока A для выполнения следующей итерации операции деления.

В данном примере $C_1 = A - B \cdot A_3 = 0,1,4,1, - - 0,2,3,1 \cdot 0,2,3,1 = 0,1,4,1 - 0,1,4,1 = 0$, операция завершена, $A_3 = 0,2,3,1 = 8_{10}$ является результатом деления.

Реализация экспериментального макетирования позиционной ОВС [6] и ОВС СОК [3] проводится на базе ПЛИС XILINX SPARTAN3 на частоте 200 МГц.

Как показывают результаты экспериментального исследования (табл. 3) преимущества позиционных ОВС наблюдаются только при обработке чисел малой разрядности (8—64 разряда). А так как обработка чисел большой разрядности (128 и более разрядов) в современных процессорах актуальна, то можно сделать вывод, что использование конвейерных арифметических устройств в нетрадиционной арифметике — перспективное направление науки.

Таблица 3

Временные и аппаратные затраты на реализацию ОВС

Разрядность данных	Устройство деления в СОК		Устройство деления в позиционной системе счисления	
	Временная задержка результата относительно данных, нс	Аппаратные затраты, число логических ячеек ПЛИС (LC)	Временная задержка результата относительно данных, нс	Аппаратные затраты, число логических ячеек ПЛИС (LC)
8	135	639	37,5	273
16	195	834	52,5	1092
32	195	3015	97,5	4368
64	195	6100	195	17 472
128	195	11 205	390	69 888

В результате теоретической и практической работы автором установлено, что использование в параллельно-конвейерных устройствах деления с настраиваемой логикой алгоритмов арифметического деления чисел с фиксированной точкой, реализованных в базисе систем счисления в остаточных классах, обеспечивает повышение скорости вычислений в среднем в 2 раза при использовании форматов данных более 128 разрядов, а при форматах менее 64 разрядов — неэффективен по сравнению с аналогичными решениями в позиционной системе счисления и понижает производительность вычислительной системы в среднем в 3 раза.

Показано, что использование базиса модулярной арифметики при построении конвейерных устройств деления обеспечивает сокращение аппаратных затрат в 6 раз при форматах более 128 разрядов, в среднем в 2 раза — при использовании форматов 32...64 разряда, а при использовании форматов менее 16 разрядов увеличение аппаратных затрат по сравнению с аналогами в позиционной системе счисления составляет 1,5...3 раза.

Список литературы

1. **Акушский И. Я., Юдицкий Д. И.** Машинная арифметика в остаточных классах. М.: Сов. радио, 1968. 439 с.

2. **Федюнин Р. Н.** Ячейка однородной вычислительной среды / В. С. Князьков, Р. Н. Федюнин // Патент РФ № 2004136518/09 от 12 мая 2006 г.

3. **Федюнин Р. Н.** Ячейка однородной среды / В. С. Князьков, Р. Н. Федюнин // Патент РФ № 2004136518/09 от 15 мая 2006 г.

4. **Федюнин Р. Н.** Способы организации и сложность массовых вычислений в конвейерных вычислительных системах // Научное обозрение. 2006. № 3. С. 89—100.

5. **Федюнин Р. Н.** Оценка пространственно-временной сложности и способы повышения скорости двоичных арифметических операций // Научное обозрение. 2006. № 3. С. 100—111.

6. **Федюнин Р. Н.** Ячейка однородной структуры / Р. Н. Федюнин, В. С. Князьков // Патент РФ 2295147. Опубл. 10.03.2007. Бюл. № 7.

УДК 519.2.512:681.3.06

Р. Р. Сулейманов, канд. пед. наук, доц.,
Центр информатизации образования,
Башкирский институт развития образования

Делимость в системах счисления. Модуль устройства определения кратности дискретного сигнала

Рассматриваются признаки делимости в системах счисления. Приводится общий признак делимости в различных системах счисления. Рассмотрены некоторые признаки делимости и алгоритмы, определяющие делимость в двоичной системе счисления. В качестве практического применения приводится принципиальная схема модуля устройства определения кратности дискретного сигнала.

Сумма, разность и произведение двух целых чисел — всегда целые числа. Этот факт принято называть замкнутостью множества целых чисел по отношению к действиям сложения, вычитания и умножения.

По отношению к действию деления множество всех целых чисел не является замкнутым: частное от деления одного целого числа на другое может и не быть целым. Поэтому при изучении обстоятельств, связанных с делением целых чисел, одним из первых встает вопрос о выполнимости этого действия на множестве целых чисел, т. е. о делимости этих чисел. Под "числом" далее будем понимать целое число, если не оговорено противное.

Для выяснения делимости числа a на число b имеется довольно много разнообразных способов. Один из них состоит в непосредственном делении числа a на число b. Однако такое деление часто оказывается долгим и утомительным занятием, и, естественно, появляется желание установить истинность интересующей нас делимости, не выполняя фактического деления. Способы, которые позволяют установить факт делимости чисел, не проводя грубое деление a на число b, называются признаками делимости. По существу эти способы должны установить признак делимости более коротким, экономным путем.

Некоторые признаки делимости целых чисел общеизвестны, например, на 2, 3, 5, 9 и т. п.:

- число делится на 2, если на 2 делится число единиц его последнего разряда;
- число делится на 3, если сумма его цифр делится на 3;
- число делится на 5, если его последняя цифра есть 5 или 0;
- число делится на 9, если сумма его цифр делится на 9.

Перечисленные признаки связаны с представлением чисел в десятичной системе счисления. Эти признаки не применимы, если пользоваться системой счисления с каким-либо другим основанием, отличным от 10. Например, 86 в восьмеричной системе записывается в виде $(126)^8$ (так как $86 = 1 \cdot 8^2 + 2 \cdot 8 + 6$). Сумма цифр равна 9, но 86 не делится ни на 9, ни на 3.

Однако для каждой позиционной системы счисления можно сформулировать свои признаки делимости на то или иное число.

Так, например, в двенадцатеричной системе можно сформулировать следующие признаки делимости:

- число $A = (a_n a_{n-1} \ldots a_1 a_0)_{12}$ делится на 8, если на 8 делится число $(a_1 a_0)_{12}$, образованное его двумя последними цифрами;
- число $A = (a_n a_{n-1} \ldots a_1 a_0)_{12}$ делится на 9, если на 9 делится число $(a_1 a_0)_{12}$, образованное его двумя последними цифрами;
- число $A = (a_n a_{n-1} \ldots a_1 a_0)_{12}$ делится на 11, если на 11 делится сумма его цифр, т. е. число $a_n + a_{n-1} + \ldots + a_1 + a_0$.

Но нас в данном случае будут интересовать не признаки делимости в каждой системе счисления, а признаки делимости, существующие в десятичной системе счисления.

Системы счисления с различными основаниями инвариантны, значит, можно полагать, что признаки делимости в системе счисления с одним основанием существуют в системах счисления с другим основанием, но примут другой вид.

Приведем общий признак делимости в различных системах счисления.

Признак делимости на *d* в системе счисления с основанием *a*

Запись числа x в системе счисления с основанием a группируется по k цифры в каждой, начиная справа, где k удовлетворяет следующим условиям:

$(a^k - 1) \bmod d = 0$ [$a^k - 1$ кратно d], $x \geqslant a^k$, k — натуральное число.

Складываем образованные группы. Если получившаяся сумма делится на d, то исходное число x делится на d.

Для определения числа цифр группировки для определения признака делимости на d в системе счисления с основанием a приведем следующую программу:

Программа

```
program delimost;
var a,n,d,v:integer;
    st:longint;
procedure stepen(var x,y; z:longint); {возведение x в
                степень y, st — результат возве-
                дения в степень}
```

```
var i:integer;
begin
   st : = 1;
      for i : = 1 to n do
      st : = st*a;
end;
begin
   readln(a, d); { a — основание системы счисления, d — делимость на число, n — количество цифр группировки}
     for n : = 1 to 20 do {20 — максимальный показатель степени}
      begin stepen (a, n, st);
        for v : = 1 to st do
        if st − 1 = d*v then writeln('a = ',
        a,'','n = ',n,'','d = ', d);
   end;
end.
```

Так, в системе счисления с основанием 2 для определения делимости на 3 группируем на группы по $k_{\min} = 2$, так как $(2^2 - 1) \bmod 3 = 0$; для определения делимости на 5 группируем на группы по $k_{\min} = 4$, так как $(2^4 - 1) \bmod 5 = 0$; для определения делимости на 7 группируем на группы по $k_{\min} = 4$, так как $(2^3 - 1) \bmod 7 = 0$.

Рассмотрим некоторые признаки делимости и алгоритмы, определяющие делимость в двоичной системе счисления.

Признак делимости на 3 в двоичной системе счисления. Запись числа в двоичной системе счисления разделим на группы по две цифры в каждой, начиная справа. Складываем образованные группы. Если получившаяся сумма делится на 3, то исходное двоичное число делится на 3.

Докажем приведенное утверждение. Запишем двоичное число в виде суммы ряда, сгруппируем по два справа и вынесем в образовавших группах общий множитель:

$$a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \ldots + a_7 \cdot 2^7 + a_6 \cdot 2^6 + a_5 \cdot 2^5 + a_4 \cdot 2^4 + a_3 \cdot 2^3 + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0 = \ldots + (a_7 \cdot 2^7 + a_6 \cdot 2^6) + (a_5 \cdot 2^5 + a_4 \cdot 2^4) + (a_3 \cdot 2^3 + a_2 \cdot 2^2) + (a_1 \cdot 2^1 + a_0 \cdot 2^0) = \ldots + 2^6 \cdot (a_7 \cdot 2 + a_6) + 2^4 \cdot (a_5 \cdot 2 + a_4) + 2^2 \cdot (a_3 \cdot 2 + a_2) + 2^0 \cdot (a_1 \cdot 2 + a_0).$$

Общий вид слагаемых $2^{2k} \cdot (a_{k+1} \cdot 2 + a_k)$. Докажем, что множитель всегда равен $2^{2k} = 3 \cdot N + 1$, где N — натуральное число.

$$2^{2k} = 3 \cdot N + 1;$$

$$2^{2k} - 1^2 = 3 \cdot N;$$

$$(2^k - 1) \cdot (2^k + 1) = 3 \cdot N.$$

На 3 делится либо $2^k - 1$, либо $2^k + 1$, так как 2^k не делится на 3 (три последовательных натуральных числа). Запишем остатки при делении нашего числа на 3:

$$(a_1 \cdot 2 + a_0) + (a_3 \cdot 2 + a_2) + (a_5 \cdot 2 + a_4) + \ldots = (a_1 a_0)_2 + (a_3 a_2)_2 + (a_5 a_4)_2 + \ldots$$

Слева сумма образовавшихся групп. Если эта сумма делится на 3, то само двоичное число делится на 3. Утверждение доказано.

Алгоритм определения делимости на 3 двоичного числа

а. Запись числа в двоичной системе счисления разделить на группы по две цифры в каждой, начиная справа. Сложить образованные группы.

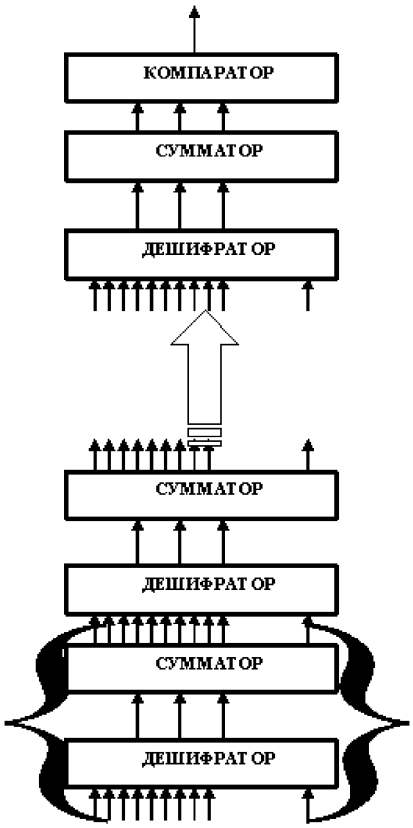
б. Если число цифр суммы больше 2, то перейти к п. а).

в. Если сумма равна 11_2, то двоичное число делится на 3.

Признак делимости на 5 в двоичной системе счисления. Запись числа в двоичной системе счисления разделим на группы по четыре цифры в каждой, начиная справа. Складываем образованные группы. Если получившаяся сумма делится на 5, то исходное двоичное число делится на 5.

Алгоритм определения делимости на 5 двоичного числа

а. Запись числа в двоичной системе счисления разделить на группы по четыре цифры в каждой, начиная справа. Сложить образованные группы.



б. Если число цифр суммы больше 4, то перейти к п. а).

в. Если сумма равна 1111_2, 1010_2 или 101_2, то исходное двоичное число делится на 5.

Признак делимости на 7 в двоичной системе счисления. Запись числа в двоичной системе счисления разделим на группы по три цифры в каждой, начиная справа. Складываем образованные группы. Если получившиеся сумма делится на 7, то исходное двоичное число делится на 7.

Алгоритм определения делимости на 7 двоичного числа

а. Запись числа в двоичной системе счисления разделить на группы по три цифры, начиная справа. Сложить образованные группы.

б. Если число цифр суммы больше 3, то перейти к п. а).

в. Если сумма равна 111_2, то исходное двоичное число делится на 7.

Алгоритм определения делимости на 15 двоичного числа

а. Запись числа в двоичной системе счисления разделить на группы по четыре цифры справа. Сложить образованные группы.

б. Если число цифр суммы больше 4, то перейти к п. а).

в. Если сумма равна 1111_2, то исходное двоичное число делится на 15 .

Модуль устройства определения кратности дискретного сигнала

Одним из областей применения признаков делимости в двоичной системе счисления является определение кратных сигналов.

Можно предложить модуль устройства, выделяющий дискретный сигнал кратный *d*. Модуль состоит из дешифраторов, сумматоров и компаратора. Назначение дешифратора: входной параллельный сигнал разрядности *n* группируется по *k* (значение *k* определяется по признаку делимости на *d* в двоичной системе счисления) и подается группами к входу сумматора. Сумматор складывает и выводит результат сложения в виде *n*-разрядного двоичного числа (сигнала), который вновь подается на дешифратор, далее на сумматор и т. д.

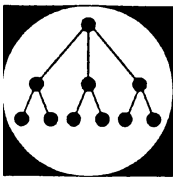
На выходе сигнал подается на компаратор для сравнения со значением *d*. В случае равенства значений исходный сигнал кратен *d*.

Число блоков "дешифратор—компаратор" можно взять $m_{\min}$, такой, что $n \leqslant 2^m$.

Принципиальная схема описанного устройства представлена на рисунке.

Литература

Сулейманов Р. Р. Признаки делимости в двоичной системе счисления // Информатика и образование. 2001. № 9.



КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

УДК 004.658:007.51

М. А. Аникин, А. Д. Брейман, канд. техн. наук, доц.,
Московский государственный университет приборостроения и информатики

Гибридная модель интеграции информации для корпоративных информационных систем с сервисно-ориентированной архитектурой

На основе анализа ключевых задач, проблем и тенденций развития средств интеграции информации в современных корпоративных информационных системах предложен подход к автоматизации интеграции информации на основе децентрализованной распределенной кэш-памяти. Предложена гибридная модель интеграции на основе комбинирования подходов кэш-памяти и интеграции корпоративной информации.

Введение

Важным фактором успешной работы предприятия является его способность к быстрому реагированию на изменяющуюся ситуацию, к организации оперативной работы своих сотрудников и подразделений. При этом постоянно возрастают как степень автономности производственного задания, так и объем разнородной разрозненной информации, связанной с ним. Для организации интегрированной корпоративной информационной системы (КИС) часто требуется реализовать выполнение "сквозных" процессов, затрагивающих множество слабо связанных автономных информационных подсистем.

Таким образом, становится актуальной проблема разработки интегрированных средств управления корпоративной информацией в условиях неопределенности и динамичности информационной среды.

Современные подходы к интеграции данных

Средства интеграции данных из разных источников в современных КИС, как правило, основаны на организации хранилища данных, наполняемого процессами извлечения, преобразования и загрузки данных (*Extract-Transform-Load, ETL*). Хотя такие системы предоставляют высокий уровень управления интегрированной информацией, позволяют выделять скрытые закономерности за счет глубокого анализа данных, но при этом отсутствует возможность выполнять такой анализ в реальном времени, а также эффективно работать с распределенными данными. Кроме того, использование ETL-подхода требует существенных затрат ресурсов как на этапе внедрения, так и в процессе эксплуатации КИС.

Сервисно-ориентированная архитектура. Для того чтобы получить возможность управлять данными в реальном времени и проводить интеграцию постепенно был предложен подход интеграции корпоративных приложений (*Enterprise Application Integration, EAI*). При следовании этому подходу обмен данными между отдельными их источниками выполняется посредством передачи сообщений между приложениями уже внедренных информационных систем.

Подход EAI получил свое дальнейшее развитие в идеях сервисно-ориентированной архитектуры (*Service-Oriented Architecture, SOA*), которая подразумевает, что при выполнении сквозных бизнес-процессов взаимодействуют слабо связанные сервисы [3]. SOA позволяет создать пространство процессов, скрывая от приложений реализации программ, предоставляя унифицированный язык сервисов, например, WSDL.

Несмотря на высокий уровень абстракции и интероперабельности сервисов для эффективного применения SOA требуется адаптация интегрируемых источников данных. Клиент-серверные архитектуры не рассчитаны на то, что в качестве источников данных используются сервисы SOA, данные для которых могут быть заранее неизвестны.

С появлением новых сервисов и источников данных растет избыточность информации. Для приемлемой производительности систем, построенных на SOA, необходима репликация больших объемов данных. Кроме того, использование сервисов осложнено из-за возросшего количества слабоструктурированной информации, составляющей более 80 % для западных и 65 % для отечественных компаний [4].

Для слабоструктурированной информации характерно наличие описания лишь локального контекста их использования, что не позволяет определять их место в глобальном интегрированном представлении. Пользователи могут иметь о ней различающиеся представления, что приводит к возрастанию числа сервисов и их связей. Теряются такие важные качества, как повторное использование, абстрактность представления сервисов, возможность постепенной модификации. Система, основанная на SOA, тогда функционирует как обычная EAI.

Интеграция информации предприятия. Для преодоления ограничений SOA была выдвинута стратегия интеграции корпоративной информации (*Enterprise Information Integration, EII*) [1]. Подход EII, как и SOA, отвечает за создание абстрактного слоя, но он нацелен на данные, с которыми работает система, а не для программных приложений, ее составляющих. EII позволяет представить всю интегрированную информацию вне зависимости от ее местонахождения в виде распределенной виртуальной глобальной схемы. Это дает возможность выполнять запросы к системе как к централизованной базе данных. Все EII-системы имеют ряд общих свойств, таких как использование принципов федеративных баз данных, распределенных запросов, создание или повторное применение метадепозитария семантических данных. EII-решения различаются по степени автоматизации создания метадепозитариев, получению схем из источников данных, степенью интероперабельности [2].

Проблемы интеграции EII и их решения. EII-интеграция подразумевает хранение только семантической схемы источников информации. Это позволяет повторно использовать готовые фрагменты схемы для управления разными источниками данных, уменьшать объем метадепозитария и сложность интеграции. Однако часто данные, затребованные федеративным запросом, в сильной степени распределены, а объем их может оказаться значительным. Поэтому доступ к ним большого числа участников в реальном времени может быть затруднен.

Повысить скорость доступа можно, применив распределенное кэширование непосредственно в памяти источников информации. Эта надстройка над существующей системой интеграции получила название информационной матрицы предприятия (*Enterprise Data Fabric, EDF*) [5]. Для ее организации создается виртуальная сеть, состоящая из секторов, на основе которых реализовано разделение информации на части, дающие равномерное распределение по объему и нагрузке [6]. Работа сервисов SOA становится независимой от физического расположения источников данных, так как они дублируются в их виртуальном представлении.

При применении EDF как дополнение к существующей интеграции, даже в случае использования децентрализованного подхода, основанного на распределенных хеш-таблицах (*Distributed Hash Tables, DHT* [11]), сохраняется потребность в глобальном представлении информации [12].

Получить глобальную информационную схему без привлечения экспертов достаточно сложно по причине потенциально слабой связанности данных и обусловленных этим неизбежных ошибок. Если бы создание индексов EDF было возможно без заранее выполненной интеграции с глобальной схемой, это позволило бы получить еще более унифицированное представление информации. Возможности по управлению таким децентрализованным слабоструктурированным виртуальным представлением будут напрямую зависеть от наличия знаний о структуре информационной среды.

Если таких знаний нет, то можно реализовать требуемую для SOA глобальную схему представления информации не на уровне стратегии интеграции, а лишь на уровне универсального сервиса, который будет создаваться на базе федеративного запроса к информационной матрице. В этом состоит одно из важных преимуществ SOA: интеграция будет заключаться в составлении правильных запросов и бизнес-правил, а не в написании

программного кода. Автоматизация интеграции позволит снизить трудозатраты на внедрение и сопровождение системы интеграции, а также даст возможность работать с данными в реальном времени.

Существующие EDF-решения не позволяют автоматизировать процесс создания виртуального сектора и требуют наличия предварительного выполнения интеграции данных, либо явного указания источников информации и распределения данных между ними.

Автоматизация интеграции данных на основе EDF

В целом средства EDF не ставят перед собой задачу автоматизации выделения семантических связей между источниками данных. Это вызвано прежде всего тем, что они изначально рассчитаны только на создание высокопроизводительной виртуальной сети доступа в реальном времени к критическим данным. Необходимый анализ информационной среды перекладывается на компоненты EII или ETL. Четкое разграничение этих задач позволяет отделить уровень данных от уровня их семантики.

Для самой EDF не требуется первичное выделение всех метаданных информационной структуры, которые в данном случае нужны только для создания уникальных пар "ключ—значение", идентифицирующих данные. Так как EDF работает с записями "ключ—значение", задача ее автоматизации может быть сведена к получению уникальных ключей записей, что обеспечит возможности их чтения/записи, причем эти ключи не должны обязательно обладать какой-либо структурной связью между собой.

Средства EDF. В основе подхода к интеграции данных на основе EDF лежит использование децентрализованной глобальной схемы метаданных. Этот подход тесно связан с одноранговыми системами управления данными (ОСУД), активно исследуемыми в последние годы множеством научных коллективов [7]. Отличие подхода EDF состоит в большей адаптации под практические задачи SOA-интеграции, в том числе — в наличии реестра метаданных, неприемлемого в "академических" ОСУД.

Идея хранения метаданных непосредственно в распределенном виртуальном представлении EDF была реализована в системе архивации данных BitVault компании Microsoft [10]. В отличие от рассмотренных ранее EDF в ней используется составной ключ, один из компонентов которого представляет собой результат вычисления хеш-функции от хранимых данных, а второй компонент − хеш-функция от их описания.

Система BitVault ориентирована на централизованный каталог ресурсов и не работает со слабоструктурированными данными. Хранящаяся в ней информация не может обладать иерархическим онтологическим описанием. Для работы со структурированными данными в корпорации Microsoft разработана система Z-Ring, обладающая тем же недостатком — централизованным управлением [9].

В неструктурированных одноранговых сетях предлагалось использовать фильтры Блума и проводить кластеризацию информации по обобщенному семантическому содержанию [7]. Однако для SOA прежде всего важно частичное онтологическое представление информации. Можно использовать глубину дерева фрагмента XML, как это было предложено в ОСУД XP2P, хэшируя отдельные его части, а затем хранить и извлекать интегрированные документы по этому пути. Но это ограничивает масштабируемость, производительность и надежность [7].

Выводы. Пространство ключей EDF для индексации слабоструктурированных в глобальном контексте локальных XML-схем и реляционных табличных структур должно обладать иерархией, представимой в виде цепочки из связей родитель—потомок. Создание иерархии на основе маршрутизации между узлами не является приемлемым для автоматизации EDF. Структура данных должна быть отражена в самом распределенном индексе.

Необходимо, чтобы виртуальное пространство делилось не по отношению к распределению данных по секторам, а относительно децентрализованно хранимых метаданных. Это приведет к пересечению пространств виртуальных секторов, так как без наличия глобальной схемы нельзя получить непротиворечивое их размещение. Это пересечение не представляет проблемы, если остается возможность адресации к каждому из секторов. Неравномерность нагрузки может быть сбалансирована путем расширения подпространства и репликации.

EDF-компонент гибридной интеграции

Ниже предложен способ организации EDF, обладающий способностью к автоматизированному децентрализованному объединению информации слабосвязанных источников, предложена модель гибридной интеграционной системы, основанной на взаимодействии созданной EDF и EII.

Гибридный способ организации EDF. В основе предлагаемого способа организации EDF лежит алгоритм распределенных хэш-таблиц, аналогичный применяемому в одноранговой системе Chord [11]. В подобных системах отсутствует глобальный каталог ресурсов, так что невозможно явно указать местонахождение запрашиваемой информации. Его заменяет виртуальное пространство идентификаторов, создаваемое за счет общей для всех узлов хеш-функции. Каждый пользователь, являющийся кэширующим узлом, самостоятельно генерирует уникальный идентификатор узла как значение хеш-функции от своего IP-адреса. Для создания уникальных идентификаторов информационных объектов (например, файлов или записей), на вход той же хеш-функции передаются текстовые представления частей информационного объекта. Идентификаторы данных и узлов имеют одинаковую разрядность и их можно сравнивать друг с другом. В узлах, чей идентификатор наиболее близок к идентификатору данных, хранится ссылка на источник этих данных или сами кэшированные данные.

Способ организации пространства идентификаторов. В предлагаемом способе организации EDF хранимые данные разделяются по виртуальным секторам, соответствующим уникальным онтологиям. Относительно локальных информационных источников интегрируемые данные структурированы изначально — онтологиями. В этом случае задача организации EDF заключается в назначении каждой онтологии некоторого набора узлов. Идентификатор состоит из двух компонентов: адреса

сектора и уникального номера в этом секторе. При управлении данными в качестве ключей будут использоваться как составные ключи, состоящие из описания онтологии и описываемого параметра информационного объекта, так и простые ключи.

Информационным объектом могут быть XML-файл, реляционная база данных, таблица или любое другое представление информации, различающее данные и метаданные. Если данные от метаданных не различаются, то невозможно создать составной ключ и интегрировать эти данные. Параметром информационного объекта могут быть текстовые значения тегов XML, ячейки таблиц реляционной базы данных. Для этих параметров информационных объектов онтологиями будут названия тегов и имена столбцов таблиц.

Пусть используемая хеш-функция выдает N-разрядные двоичные значения. Значение параметра K, задающего соотношение чисел двоичных разрядов, отводимых в ключе под данные и метаданные, определяется исходя из желаемого коэффициента распределения в виртуальном секторе и его размера. Ключ составляется по следующей схеме:

1. Получить X — значение хеш-функции от текстового параметра информационного объекта, который описан онтологией.

2. Получить Y — значение хеш-функции от названия онтологии, описывающей этот параметр.

3. Ключ составляется из K старших разрядов значения X и $(N - K)$ младших разрядов значения Y. Двухсоставный ключ по разрядности совпадает с односоставными ключами (X).

Группы организуются за счет уникальности текстовых представлений параметров и онтологий информационного объекта. Каждая из них однозначно определит набор содержащихся в ней идентификаторов.

Для конкретной онтологии $(N - K)$ младших разрядов ее идентификатора соответствуют сектору виртуального кольца с угловой мерой $\pi/2K$. Неуникальные текстовые представления параметров информационных объектов порождают асимптотически равномерное распределение их идентификаторов между $2K$ секторами. Для ресурсов, описанных общей онтологией, имеется до $2K$ вариантов местоположения хеш-значений.

Методы управления ключами. Публикация, удаление и изменение идентификаторов основаны на разделении задачи по поиску записи в EDF между несколькими кэширующими узлами. При этом принимают участие три стороны: сервер, клиент и промежуточное звено, в виде одного или нескольких кэширующих узлов. Пользователю, желающему опубликовать данные, требуется определить уникальные коэффициенты K их метаданных, число разрядов идентификатора, используемых для сектора. Для этого кэширующий узел, чей идентификатор наиболее близок к простому ключу данной онтологии, получает роль промежуточного звена по хранению ее идентификатора и коэффициента K. Также промежуточное звено ведет статистику по числу обратившихся пользователей. Получив требуемое значение K, узел может обращаться к нужному сектору, чтобы публиковать, удалять, искать или изменять данные по алгоритму DHT.

Серверная часть проводит регистрацию онтологий — их добавления и удаления. При принятии роли по хранению K узел должен сообщить серверной части этот параметр и текстовое представление онтологии. Если ни один узел не использует данную онтологию, информация о ней удаляется. Сервер EDF выполняет роль справочника онтологий: так как для хеш-функции используются только локальные онтологии, у разных информационных объектов они могут не совпадать.

При регистрации новой онтологии желательно, чтобы узел либо задал более общую онтологию, которую могли бы принять другие пользователи, либо сам согласился с уже существующей. Например, для локальных онтологий price и cost более общей и понятной всем онтологией будет являться метаонтология "цена". Сервером может быть любой узел системы, он может иметь свой справочник глобальных онтологий, пользоваться чужими справочниками и предоставлять свои другим узлам для обеспечения интероперабельности.

Узлы также могут инициировать асинхронное увеличение и уменьшение коэффициента K в зависимости от нагрузки на определенные узлы кэширующего сектора, репликацию данных для повышения надежности. Пользователь может выполнять масштабируемые по отношению к числу идентификаторов и узлов запросы по области значений при неизвестном параметре информационного ресурса, так как ему требуется опросить не более, чем $2K$ узлов, а K система стремится минимизировать. Для XML-представлений возможно выполнение запроса по области значений при неизвестной онтологии. В целом реализация запросов, репликации зависит от избыточности, с которой будет опубликована информация.

Схемы интеграции для источников информации. На рис. 1 (см. четвертую сторону обложки) приведен пример описания структуры фрагмента XML-документа.

Ключ имеет следующую структуру:

СТРУКТУРА/ПОТОМОК[АТРИБУТ = = ЗНАЧЕНИЕ]/ЗНАЧЕНИЕ.

Для публикации предлагается схема, использующая хеш-функцию SHA-1 (табл. 1). Представление табличных данных может быть рассмотрено как частный случай XML данных. Это дает унифицированное представление для гетерогенных источников данных и обеспечивает высокую интероперабельность между ними (табл. 2).

При имитационном моделировании с использованием прототипа системы для приведенного выше примера

Таблица 1

Формирование ключей для XML-представлений

XML-тег	Обозначение	Способ организации
Родительский узел	struct	Имя узла схемы [160]
Атрибут родителя	atr*/struct	Атрибут* [1..k] + узел [160 − k]
Значение атрибута	val*/atr*	Знач. атрибута* [0..m] + атрибут [160 − m]
Потомок родителя	node/struct	Узел [1..k] + структура [160 − k]
Значение потомка	val**/node	Значение узла** [0..p] + узел [260 − p]

* Используются инвертированные значения хэш-функции.
** Хеш-ключ записывается в обратном порядке.

Таблица 2

Формирование ключей для реляционных БД

Часть таблицы	Обозначение	Способ организации
Первичный ключ	struct**	Поле первичного ключа [160]
Имя столбца	atr*/struct	Столбец* [1..k] + перв. ключ [160 − k]
Ячейка столбца	val*/atr*	Ячейка столбца* [0..m] + + столбец [160 − m]
Ячейка первичного ключа	node/struct	Ячейка перв. ключа [1..k] + + первичный ключ [160 − k]

* Используются инвертированные значения хэш-функции.
** Если есть первичный ключ или таблица в реляционной БД.

Таблица 3

Характеристики системы

Характеристики EDF-системы интеграции	Значение
Узлы в сети EDF	20
Пары ключ — значение	1914
Начальный параметр K	3
Диапазон распределения пар среди узлов	1,5—13,6 %
Среднее число ключей на участника	96
Среднее квадратическое отклонение	55

были получены характеристики системы, представленные в табл. 3 и на рис. 2 (см. четвертую сторону обложки).

Гибридная виртуальная динамическая интеграция информации. Рассмотренная EDF может быть совмещена с существующими EII-решениями. Сервисы SOA могут работать одновременно как с EDF-, так и с EII-представлениями в зависимости от имеющихся у них семантических описаний, предъявляемых требований к полноте и скорости выполнения для различных частей сквозного бизнес-процесса, являются гибридными по отношению к интеграции информационной среды (рис. 3, см. четвертую сторону обложки).

В этом случае системы EII получают части глобальной схемы с локальных источников и создают непротиворечивую информационную картину. Параллельно локальные источники в автоматическом режиме осуществляют загрузку своей информации в EDF без использования глобальных описаний EII. Если запрос был составлен правильно, вне зависимости от того известны семантические связи или нет, он будет выполнен. Зная глобальную схему, можно все равно обращаться к EDF-компоненту, так как его скорость обработки запроса практически всегда выше. Справочник используемых онтологий матрицы при необходимости может быть применен и в EII-интеграции для формирования глобальной схемы. Регистрация онтологий на серверной части EDF позволяет вести контроль над уникальностью используемых терминов.

Выводы

Вместо установления связей между всеми параметрами информационного объекта и его онтологиями предлагается хранить несвязанные данные. Единственным структурированным представлением данных является двухуровневая организация, где низший уровень — это параметр, а верхний — его онтология.

Эффективность гибридного решения зависит от правильно составленных запросов SOA- и EDF-справочников онтологий. Справочник онтологий, в отличие от глобальной схемы EII, не требует описания физического расположения данных и их семантических связей друг с другом. Он может заполняться пользователями, так как требует только интуитивно понятного описания имеющихся у них данных. Это позволяет пользователям удобно управлять своими данными. При этом обеспечивается высокий уровень взаимодействия между ними, например, в сетях групповой работы [8], таких как Microsoft Groove networks.

Сквозные бизнес-процессы, организованные на базе EDF, не привязаны к конкретным метаданным информационных источников, за счет чего становится возможным эффективное применение SOA независимо от наличия унаследованных систем и их состояния.

Список литературы

1. **Guide** to Enterprise Information Integration (EII). 2006. Ipedo, www.ipedo.com/downloads/Ipedo_EII_Guide.pdf.

2. **Madsen M.** Enterprise Information Integration Technology: Architectures, Uses and Abuses // TDWI EI Series, Feb 2007. www.clickstream.blogspot.com/2007/08/slides-for-data-federationeii-uses-and.html

3. **Bringing** SOA Value Patterns to Life. An Oracle White Paper. June 2006. www.oracle.com/technologies/soa/soa-value-patterns.pdf

4. **Интеграция** клиентских данных. Этапы управления нормативно справочной информацией. — Intersoft Lab, 2006. www.iso.ru/cgibin/main/news_tech.cgi?id=288.

5. **Черняк Л.** EDF — корпоративная кэш-память // Открытые системы. 2006. № 8. www.osp.ru/os/2006/08/3584557/

6. **Powering** the Next Generation Data Infrastucture: GemFire on Intel for Financial Services (GIFS). White paper. GemStone Systems, Intel Corporation, 2006. www.gemstone.com/pdf/GIFS_Reference_Architecture_Grid_Data_Management_low.pdf

7. **Koloniari G., Pitoura E.** Peer to Peer Management of XML Data: Issues and Research Challenges // SIGMOD Record. June 2005. Vol. 34. N. 2. http://www.cs.uoi.gr/~pitoura/distribution/sr05.pdf

8. **New Collaborative** Working Environments 2020. Report on industry-led FP7 consultations 3rd Report of the Experts Group on Collaboration@Work.2006. http://ec.europa.eu/information_society/activities/atwork/hot_news/publications/documents/new_collab_environments_2020.pdf

9. **Qiao L., Wei C., Zheng Z., Shaomei W.** Z-Ring: Fast Prefix Routing via a Low Maintenance Membership Protocol. Microsoft Research.2005. http://www.cs.cornell.edu/~sw475/publications/zring.pdf

10. **Zheng Z., Qiao L., Shiding L., Wei C., Yu C., Chao J.** BitVault: a Highly Reliable Distributed Data Retntion Platform. 2005. www.research.microsoft.com/research/pubs/view.aspx?tr_id = 1035

11. **Stoica I., Morris R., Karger D., Kaashoek M., Balakrishnan H.** Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. SIGCOMM'0l, 2001. www.pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf

12. **Achieving** the Impossible: Unlimited Application Scalability. Oracle White Paper, 2007. http://www.bizreport.com/whitepapers/achieving_the_impossible_unlimited_application.html

УДК 004.3:004.9

И. И. Исмагилов, д-р техн. наук,
П. А. Зиновьев, канд. техн. наук,
В. А. Зинкин,
Институт проблем информатики АН РТ,
г. Казань

Оценка уровня развития сложных технических систем: методика и ее приложение к корпоративным информационным системам

Рассмотрены вопросы оценки уровня развития сложных технических систем. Описывается предложенная методика решения рассматриваемой задачи. Обсуждается ее конкретизация применительно к оценке уровня развития корпоративных информационных систем.

Введение

Перспективы решения проблем создания и развития сложных технических систем (СТС), в том числе и корпоративных информационных систем (КИС), в значительной мере связаны с разработкой теоретических основ системотехники и состоянием соответствующего научно-методического обеспечения всех этапов их жизненного цикла (ЖЦ).

Одним из важных этапов ЖЦ является этап управляемого развития (совершенствования, модернизации) СТС. Под *развитием* в контексте данной статьи понимается целенаправленный процесс изменения сложности и эффективности системы. Понятие развития отражает тип динамики систем, и, следовательно, можно говорить как о прогрессивном, так и регрессивном развитии в зависимости от характера изменений в системе.

Отметим, что в ряде случаев регрессивное развитие может привести к повышению эффективности системы. Например, если под влиянием внешних факторов резко уменьшился масштаб бизнеса предприятия, то существующая его информационная система в новых условиях может оказаться избыточной, а значит, неэффективной. Процесс ее приведения в соответствие с новыми условиями функционирования предприятия, очевидно, будет регрессивным развитием.

Таким образом, цель этапа управляемого развития СТС заключается в построении долгосрочного плана (траектории) развития системы в заданном направлении и проведении комплекса мероприятий по его практической реализации.

Как правило, создание проектов развития СТС связано с решением ряда сложных, трудоемких и слабоформализуемых задач. В настоящее время они в основном решаются эвристическими методами на основе опыта и интуиции проектировщиков. Среди таких задач важную роль играет прогнозирование направлений и периодичности проведения модернизаций системы. Здесь возникает задача сравнительной оценки уровня развития СТС в рамках рассматриваемых альтернатив плана развития. Очевидно, что эта задача должна рассматриваться в многокритериальной постановке с учетом объективно существующих факторов неопределенности и недостаточного уровня знаний о показателях качества системы.

Следует отметить, что повышению качества проектов может способствовать расширенное применение компьютерных средств и технологий поддержки решения слабоформализуемых задач. Создание таких средств требует использования формальных методов обработки экспертной информации, на основе которой определяются многие ключевые проектные решения.

Анализ работ в области системологии и системотехники [1—9] показал, что на данный момент отсутствуют общепринятые методики оценки уровня развития СТС. Вследствие этого актуальность проблемы оценки уровня их развития не вызывает сомнений. Решение этой проблемы предполагает разработку соответствующих научно обоснованных, практически реализуемых методик.

Методика оценки уровня развития сложных технических систем

Приведем краткое описание методики оценки уровня развития СТС с использованием гипервекторного критерия, в которой используются векторные оценки уровней и устойчивости отдельных видов развития системы [10]. В рассматриваемой методике в зависимости от целей исследования систем при оценивании могут быть предусмотрены такие виды их развития, как физическое, интеллектуальное, экологическое, технологическое, экономическое и т. д. Под в*идом развития* СТС в данной работе понимается результат классификации процесса развития по характерным особенностям изменения ее состава, структуры, функций или целей. При этом основу классификации составляют определенные авторами признаки (критерии оценки уровня развития систем).

Гипервекторный критерий оценки уровня развития СТС в общем случае может быть представлен так:

$$\mathbf{HVC} = \{\mathbf{VC}_1, \mathbf{VC}_2, ..., \mathbf{VC}_n\},$$

где $\mathbf{VC}_i$, $i = \overline{1, n}$ — соответственно векторные критерии оценки уровня i-го вида развития.

Предлагается векторный критерий оценки уровня i-го вида развития формировать двухкомпонентным в следующем виде:

$$\mathbf{VC}_i = \{DI_i, SD_i\},$$

где DI_i, SD_i — соответственно индекс уровня и коэффициент устойчивости i-го вида развития системы.

При этом индекс уровня развития будет характеризовать относительный уровень изменений в системе, а коэффициент устойчивости — относительную величину позитивных изменений в условиях воздействия как прогрессивно влияющих, так и дестабилизирующих факторов.

Для оценки уровня развития СТС достаточно использовать двухвекторный критерий, в котором использованы векторные оценки уровней и устойчивости физического и интеллектуального развития системы. При этом под физическим развитием СТС понимается расширение ее масштаба и возрастание объемов используемых физических ресурсов, а под интеллектуальным — расширение функциональных возможностей системы и повышение эффективности решаемых ею задач. По мнению авторов, указанные направления развития в случае СТС достаточно полно и адекватно характеризуют процессы развития таких систем в целом.

В данном случае предлагаемый двухвекторный критерий оценки уровня развития СТС имеет следующую структуру:

$$\mathbf{HVC} = \{\mathbf{PDC}, \mathbf{IDC}\},$$

где $\mathbf{PDC} = \{PDI, SPD\}$, $\mathbf{IDC} = \{IDI, SID\}$ — соответственно векторные критерии физического и интеллектуального развития.

Для элементов этих векторных критериев приняты следующие названия:

- PDI — индекс физического развития;
- SPD — коэффициент устойчивости физического развития;
- IDI — индекс интеллектуального развития;
- SID — коэффициент устойчивости интеллектуального развития.

Предложенные векторные критерии оценки уровня развития СТС можно рассматривать в двух вариантах в зависимости от системы, принятой в качестве базисной, относительно которой определяется сравнительный уровень развития СТС. Для элементов этих двух вариантов векторных критериев оценки уровня развития СТС введем следующие названия:

- индексы и коэффициенты устойчивости внутреннего развития;
- индексы и коэффициенты устойчивости внешнего развития.

При использовании первого варианта определения векторных критериев проводится сравнительный анализ состояний одной и той же системы, а второй вариант связан со сравнительным анализом состояний системы по отношению к состояниям другой однотипной системы.

Опишем методику определения элементов векторных критериев оценки уровня развития на примере индексов и коэффициентов устойчивости внутреннего развития СТС.

Рассмотрим два анализируемых состояния системы (АСС) в начале этапа ее развития и после его завершения, для которых введем соответственно обозначения S_0 и S_1. Для оценки развития системы на физическом и интеллектуальном уровнях проведем формирование критериального описания этих АСС с использованием двух групп частных критериев. В целях вычисления соответствующих индексов развития применим метод анализа иерархий (МАИ) [11]. В связи с этим в дальнейшем будем придерживаться терминологии, принятой в этом методе.

На основе МАИ индексы физического и интеллектуального развития определяются следующим образом:

- индекс физического развития

$$PDI = k_{pd}(GPP_1 - GPP_0),$$

где GPP_i — глобальный приоритет S_i по группе критериев оценки физического развития;

- индекс интеллектуального развития

$$IDI = k_{id}(GPI_1 - GPI_0),$$

где GPI_i — глобальный приоритет S_i по группе критериев оценки интеллектуального развития.

Отметим, что глобальный приоритет АСС — это его (состояния системы) агрегированная характеристика (относительная важность, вес).

Здесь в соответствии с МАИ глобальные приоритеты АСС определяются как линейные свертки их локальных приоритетов по частным критериям. Величины k_{pd} и k_{id} представляют собой нормировочные множители, использование которых приводит к определению множества значений соответствующих индексов развития на интервале $[-1, 1]$. Значения нормировочных множителей зависят от используемой шкалы оценок при парных сравнениях АСС по частным критериям и будут определены ниже. Отметим, что при использовании одной и той же шкалы оценок для обеих групп критериев $k_{pd} = k_{id}$.

Коэффициенты SPD и SID, характеризующие степень устойчивости физического и интеллектуального развития системы, определяются следующим образом:

$$SPD = \begin{cases} n_{pd}/N_{pd},\ n_{pd} > 0; \\ -m_{pd}/N_{pd},\ n_{pd} = 0. \end{cases}$$

$$SID = \begin{cases} n_{id}/N_{id},\ n_{id} > 0; \\ -m_{id}/N_{id},\ n_{id} = 0, \end{cases}$$

где n_{pd}, n_{id} (m_{pd}, m_{id}) — число частных критериев соответствующих групп, оценки по которым улучшились (ухудшились) в результате реализации исследуемого этапа развития; N_{pd}, N_{id} — общее число критериев в группах.

С учетом изложенного выше методика многокритериальной оценки развития СТС заключается в реализации следующих этапов.

1. Формирование векторов частных критериев для оценки физического и интеллектуального развития СТС.

2. Выбор шкал оценок парных сравнений критериев и АСС.

3. Построение матриц парных сравнений (МПС) критериев по степени важности и вычисление их приоритетов.

4. Построение МПС АСС S_0, S_1 по частным критериям и вычисление их локальных приоритетов.

5. Вычисление глобальных приоритетов АСС путем линейной свертки локальных приоритетов.

6. Вычисление индексов и коэффициентов устойчивости физического и интеллектуального развития.

7. Определение вида развития.

При оценке уровня внешнего развития сравнение анализируемой системы целесообразно проводить относительно системы, представляющей глобально-идеальную альтернативу развития, которая обеспечивает стратегическую конкурентоспособность на рынке. Нами она названа *конкурентоспособной стратегической альтернативой*. Эта прогнозная альтернатива определяется на уровне критериального описания системы экспертными методами на основе результатов анализа и прогнозирования развития рынка и конкурирующих систем, достижений соответствующих областей науки, техники и технологий. Значения критериев оценки уровня развития такой альтернативы считаются экспертами предельно достижимыми на анализируемом периоде развития СТС.

Классификация видов развития сложных технических систем

Предварительно остановимся на вопросе реализации отдельных этапов решения рассматриваемой задачи. С учетом особенностей полученных результатов рассмотрим вопрос классификации процессов развития и его описания на критериальном уровне.

Определение приоритетов критериев (коэффициентов относительной важности) проводится по МАИ [11]. Построение обратносимметричной МПС критериев $\mathbf{A}_N$ осуществляется с использованием вербальной шкалы из девяти градаций (шкалы Саати). На основе $\mathbf{A}_N$ решением задачи нахождения нормированного собственного вектора, соответствующего максимальному собственному значению, находятся приоритеты критериев оценки уровня развития СТС. При этом вычисление приоритетов следует проводить лишь в случае приемлемой согласованности МПС, которая в МАИ проверяется с использованием отношения согласованности матрицы. В случае плохой согласованности МПС целесообразно пересмотреть оценки на триадах критериев, в которых наблюдается нарушение транзитивности суждений экспертов.

При формировании МПС АСС по частным критериям путем опроса экспертов возможно использование усеченных вариантов шкалы Саати. На наш взгляд, целесообразно использование шкалы оценок, представленной в табл. 1.

Сформированные совокупности МПС АСС по двум группам частных критериев позволяют вычислить локальные приоритеты состояний системы S_0,S_1. При этом отметим, что оценки парных сравнений согласованы, так как порядки МПС АСС равны 2. Имеются также аналитические выражения для вычисления приоритетов [10]:

$$p_0 = a_{01}/(a_{01} + 1),\ p_1 = 1/(a_{01} + 1),$$

где a_{01} — оценка парного сравнения S_0 и S_1.

Вычисление глобальных приоритетов АСС GPP_l, GPI_l, $l = 0, 1$, проводится с использованием формул

$$GPP_l = \sum_{j=1}^{N_{pd}} WP_j \times PP_{lj};$$

Таблица 1

Интенсивность относительной важности	Определение	Объяснение
1	На уровне	Критериальные свойства объектов на одном уровне
3	Выше	Слабое превосходство одного объекта над другим
5	Намного выше	Сильное превосходство одного объекта над другим
2, 4	Промежуточные решения между соседними суждениями	Применяется в компромиссных случаях

$$GPI_l = \sum_{j=1}^{N_{id}} WI_j \times PI_{lj},$$

где WP_j, WI_j — приоритеты j-х критериев групп, характеризующих соответственно физическое и интеллектуальное развитие системы; PP_{lj}, PI_{lj} — локальные приоритеты l-й альтернативы по j-му критерию соответствующей группы.

Определение на основе этих глобальных приоритетов АСС индексов физического и интеллектуального развития требует вычисления нормировочных коэффициентов. Они могут быть вычислены следующим образом:

$$k_{pd} = k_{id} = 1/(p_{\max} - p_{\min}),$$

где $p_{\max}$, $p_{\min}$ — максимальное и минимальное возможные значения локальных приоритетов S_0, S_1.

С учетом выражений для определения приоритетов по МПС порядка 2 и с использованием максимального значения балльной оценки выбранной шкалы $a_{\max}$ величины $p_{\max}$, $p_{\min}$ вычисляются по формулам

$$p_{\max} = a_{\max}/(a_{\max} + 1);$$
$$p_{\min} = 1/(a_{\max} + 1).$$

Тогда нормировочные множители можно определить по выражению

$$k_{pd} = k_{id} = (a_{\max} + 1)/(a_{\max} - 1).$$

Значения нормировочных множителей для шкалы Саати и ее усеченных вариантов представлены в табл. 2.

С учетом этих данных при рекомендуемой шкале оценок ($a_{\max} = 5$) выражения для вычисления индексов физического и интеллектуального развития примут следующий вид:

$$PDI = 1{,}5(GPP_1 - GPP_0);\ IDI = 1{,}5(GPI_1 - GPI_0).$$

Анализ этих индексов позволяет определить вид развития системы. Предлагаемые виды развития и правила принятия решений о них приведены в табл. 3.

При визуальном представлении результатов многокритериальной оценки развития СТС предлагаемое графическое отображение строится в квадратной области двумерной числовой плоскости, где каждая координата изменяется от −1 до 1. При этом по оси абсцисс откладывается значение индекса физического развития, а по оси ординат — значение индекса интеллектуального развития как наиболее информативные компоненты векторных критериев развития системы.

Схематические позиции видов развития СТС представлены на рис. 1.

Более углубленный анализ ситуаций развития позволяет выделить ряд подвидов развития СТС.

Таблица 2

$a_{\max}$ k_{pd}, k_{id}	9 1,25	7 1,33	5 1,5	3 2

Таблица 3

№	Значения индексов развития	Вид развития
1	$PDI > 0$, $IDI > 0$	Прогрессивное
2	$PDI \leqslant 0$, $IDI \leqslant 0$, кроме случая $PDI = IDI = 0$	Регрессивное
3	$PDI = IDI = 0$	Нейтральное
4	$PDI > 0$, $IDI \leqslant 0$	Физическое
5	$PDI \leqslant 0$, $IDI > 0$	Интеллектуальное

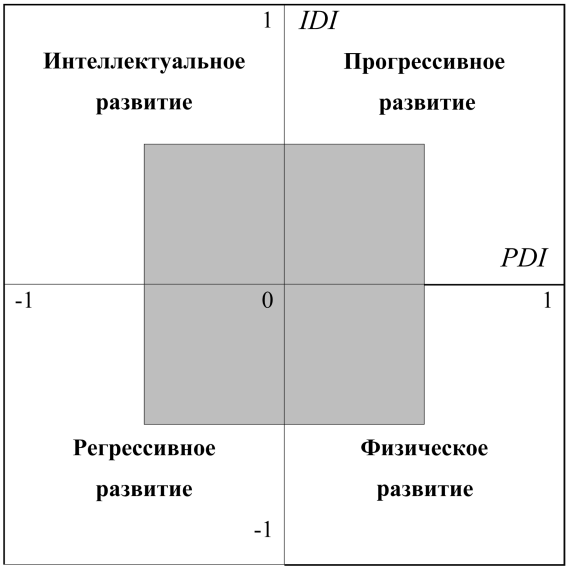


Рис. 1. Схематические позиции видов развития СТС

С этой целью предварительно разобьем опорный квадрат числовой плоскости на две непересекающиеся области неустойчивого и устойчивого развития. При этом область неустойчивого развития определим на основе значений индексов развития, вычисленных с использованием соответствующих оценок парных сравнений по частным критериям. Эти оценки представляют компромиссные решения экспертов относительно критериальных свойств между двумя первыми основными градациями шкалы оценок. При использовании шкалы Саати и ее усеченных вариантов таким значением оценки является 2. Учет этого факта и привлечение выражений для расчета приоритетов альтернатив для МПС порядка 2 позволяют определить границы области неустойчивого развития. Она является прямоугольной и задается следующими неравенствами:

$$|PDI| \leqslant k_{pd}/3;\ |IDI| \leqslant k_{id}/3,$$

где k_{pd}, k_{id} — нормировочные коэффициенты, определяемые из табл. 2.

Таблица 4

№	Значение коэффициентов устойчивости развития	Подвид развития
1	$SPD = 1$, $SID = 1$	Глобально-устойчивое прогрессивное
2	$0 < SPD < 1$, $0 < SID < 1$	Локально-устойчивое прогрессивное
3	$SPD = -1$, $SID = -1$	Глобально-устойчивое регрессивное
4	$-1 < SPD < 0$, $-1 < SID < 0$	Локально-устойчивое регрессивное
5	$SPD = 1$	Глобально-устойчивое физическое
6	$0 < SPD < 1$	Локально-устойчивое физическое
7	$SID = 1$	Глобально-устойчивое интеллектуальное
8	$0 < SID < 1$	Локально-устойчивое интеллектуальное

При использовании рекомендованной нами шкалы эта область определяется так: $|PDI| \leqslant 0{,}5$, $|IDI| \leqslant 0{,}5$. На рис. 1 область неустойчивого развития выделена фоновой заливкой.

С учетом области неустойчивого развития систем можно провести бинарную классификацию основных видов развития, выделив соответствующие их неустойчивые и устойчивые подвиды. В случае нейтрального развития правило классификации имеет отличительные особенности по отношению к другим видам развития и заключается в следующем:

- $SPD = 0$ и $SID = 0$ — устойчивое нейтральное
- $SPD \neq 0$ или $SID \neq 0$ — неустойчивое нейтральное.

Анализ ситуаций развития с учетом введенных коэффициентов устойчивости также позволяет выделить ряд подвидов устойчивого развития СТС. Предлагаемые подвиды развития и правила принятия решений о них приведены в табл. 4.

Следует отметить, что при практическом применении предложенной методики оценки уровня развития СТС необходимо конкретизировать отдельные ее этапы с учетом функционального назначения системы. При этом особое внимание должно быть уделено формированию соответствующих критериальных пространств, так как от корректности решения этой задачи зависит качество оценки уровня развития СТС.

Динамический портрет сложной технической системы

Анализ ЖЦ развивающихся СТС показывает, что после создания и введения в эксплуатацию система может находиться в двух состояниях: развития и стабилизации. Состояние стабилизации наступает после завершения очередного этапа развития системы и продолжается до начала следующего. После некоторого числа чередования этих этапов жизненного цикла возможен демонтаж системы или, при определенных условиях, ее полное уничтожение.

Для облегчения визуального анализа результатов исследования характеристик процесса развития СТС целесообразно их представление в виде кусочно-блочного псевдоцветного изображения, названного *динамическим портретом развития системы*. При его построении предлагается использовать графические примитивы, представленные в табл. 5.

В целях повышения информативности динамического портрета следует также отобразить на нем присущий системе подвид развития (нейтральное, глобально-устойчивое или локально-устойчивое), выделив цветовой заливкой соответствующий графический примитив. Целесообразно также введение условных обозначений для базовых видов развития СТС:

- ν_0 — регрессивное развитие;
- ν_1 — нейтральное развитие;
- ν_2 — физическое развитие;
- ν_3 — интеллектуальное развитие;
- ν_4 — прогрессивное развитие.

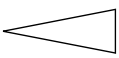
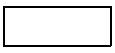
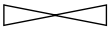
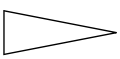
Пример построения динамического портрета СТС представлен на рис. 2.

В представленном динамическом портрете СТС фоновой заливкой выделены этапы глобально-устойчивого развития и наступившие после них этапы стабилизации.

Введем также символьное описание траектории развития СТС (код развития). При этом будем пользоваться следующими обозначениями для основных этапов жизненного цикла системы:

- S_t — создание;
- $\tilde{V}_t^i$ — неустойчивое развитие i-го вида;
- ν_t^i — локально-устойчивое развитие i-го вида;
- V_t^i — глобально-устойчивое развитие i-го вида;
- C_t — стабилизация;
- F_{t_1, t_2} — демонтаж/уничтожение.

Таблица 5

Условное графическое обозначение	Этап жизненного цикла
	Создание
	Стабилизация
	Развитие
	Демонтаж/уничтожение

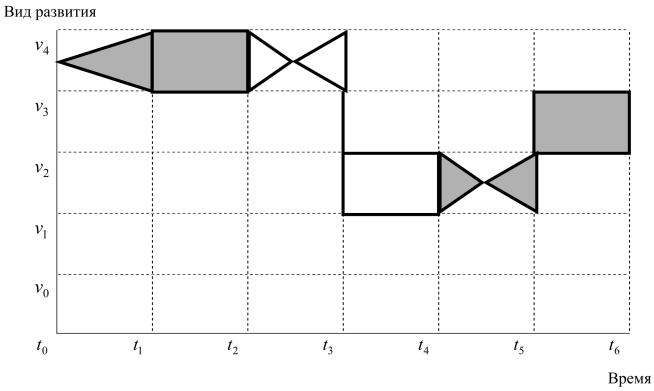


Рис. 2. Динамический портрет СТС

Здесь индексы t, t_1 — время начала реализации соответствующих этапов, t_2 — конечное время реализации этапа демонтажа/уничтожения. Значения индекса i соответствуют базовым видам развития при их условном обозначении вида ν_i, принятом при построении динамического портрета системы.

С использованием предложенных обозначений код развития СТС, динамический портрет которой представлен на рис. 2, можно записать в виде

$$S_{t_0} C_{t_1} \tilde{V}^2_{t_2} C_{t_3} V^3_{t_4} C_{t_5}.$$

По мнению авторов, код развития СТС достаточно информативен и в краткой форме представляет основные характеристики этапов ее ЖЦ.

Критерии и пример оценки уровня развития корпоративных информационных систем

В предложенной методике оценке уровня развития СТС важная роль отводится формированию векторов частных критериев для оценки физического и интеллектуального развития системы. Рассмотрим решение этой задачи применительно к корпоративным информационным системам (КИС). На наш взгляд, при оценивании уровня развития систем этого класса целесообразно использование следующих компонентов векторных критериев.

I. Критерии физического развития

1. Масштаб КИС.
2. Число локальных сетей, входящих в состав КИС.
3. Общее число рабочих мест.
4. Общая длина каналов внутрикорпоративного межсетевого взаимодействия.
5. Число Центров обработки данных.
6. Суммарная производительность Центров обработки данных.
7. Число Центров хранения данных.
8. Суммарная емкость Центров хранения данных.
9. Средняя пропускная способность локальных сетей.
10. Средняя пропускная способность каналов внутрикорпоративного межсетевого взаимодействия.
11. Число шлюзов к внешним сетям.
12. Суммарная пропускная способность каналов доступа к внешним сетям.

II. Критерии интеллектуального развития

1. Число предоставляемых ИТ-услуг (общесистемных и прикладных сервисов).
2. Качество предоставляемых ИТ-услуг.
3. Общий объем накопленных корпоративных информационных ресурсов.
4. Объем мультимедийных информационных ресурсов.
5. Средняя интенсивность внутреннего трафика.
6. Средняя интенсивность внешнего трафика.
7. Уровень интеллектуальности платформы общесистемного управления.
8. Уровень интеллектуальности информационно-вычислительной среды.
9. Уровень интеллектуальности интерфейса с пользователями.
10. Уровень обеспечения информационной безопасности.
11. Степень открытости и масштабируемости информационно-вычислительной среды.
12. Уровень надежности и отказоустойчивости.
13. Интероперабельность информационно-вычислительной среды.

Представляется, что приведенные перечни критериев физического и интеллектуального развития КИС достаточно полно характеризуют основные его направления развития и могут быть рекомендованы для практического использования.

Практическая апробация предложенной методики проведена на примере оценки уровня развития КИС Академии наук Республики Татарстан (АН РТ) после реализации проекта ее развития (2001—2005 гг.). Прежде чем привести результаты анализа, представим краткую характеристику рассматриваемой КИС, в основном, на уровне ее телекоммуникационной инфраструктуры.

Корпоративная сеть телекоммуникаций учреждений АН РТ является составной частью Единой сети научных и научно-образовательных учреждений РТ "SENet-Tatarstan". Она создавалась в 2000—2003 гг. по проекту Института проблем информатики АН РТ при поддержке гранта АН РТ.

Первый этап создания корпоративной сети АН РТ, реализованный в 2000 г., характеризовался подключением к глобальной информационной сети Интернет по выделенным линиям связи с помощью HDSL-модемов трех подразделений АН РТ (Президиум АН РТ, Институт Татарской энциклопедии, Центр травматологии и ортопедии).

Для подключения сети к распределительному хосту Интернет-провайдера на одной из автоматических телефонных станций г. Казани был смонтирован Центральный коммуникационный узел (ЦКУ) на базе маршрутизатора Cisco-3640, который одновременно выполнял роль межсетевого экрана. В зданиях указанных структурных подразделений были реализованы локальные вычислительные сети (ЛВС), технологическую основу которых составили Ethernet-коммутаторы. Общее число автоматизированных рабочих мест (АРМ), подключенных к сети Интернет на первом этапе, составило 25 единиц. Кроме того, порядка 30 персональных компьютеров (ПК) были подключены в режиме коммутируемого доступа "dial-up".

В 2001—2005 гг. сеть развивалась и модернизировалась. В частности, еще несколько институтов были подключены к ЦКУ по выделенным линиям связи, а остальные — с использованием технологии ISDN. В этот период произошел рост числа АРМ в сети до 110 единиц. В составе сети был смонтирован серверный кластер IBM Netfinity, на котором был реализован собственный Web-узел Академии наук РТ (домен antat.ru). На его базе создан Интернет-портал АН РТ, содержащий комплекс сайтов научного и культурного характера. Проводились работы по усилению информационной безопасности всей сети АН РТ, в частности, реализация служб динамической конфигурации хоста (DHCP) и внутренней системы имен доменов (DNS), установки IP-шлюзов, средств обнаружения несанкционированных вторжений и др.

По состоянию на конец 2005 г. корпоративная сеть АН РТ имела следующую структурную организацию.

1. Сеть построена по топологии "распределенная звезда", в которой каждое крупное подразделение АН РТ (Институты и научные Центры, а также Президиум АН РТ) представлены собственными развитыми ЛВС, связанными с ЦКУ с помощью маршрутизаторов.

2. В центре топологии расположен ЦКУ на базе маршрутизаторов Cisco-3640 и Cisco-2610, к которому подключены как подразделения АН РТ, так и модемные пулы для связи по коммутируемым каналам.

3. Основные подразделения АН РТ подключены к ЦКУ по выделенным линиям связи с помощью HDSL-модемов, обеспечивающих пропускную способность канала передачи данных до 2 Мбит/с. Президиум АН РТ подключен к ЦКУ по оптоволоконному каналу связи.

4. Для обеспечения доступа к сети Интернет центральный маршрутизатор ЦКУ подключен к маршрутизатору сети SENet отдельным каналом связи с пропускной способностью не менее 10 Мбит/с.

5. В одном из базовых узлов сети функционирует серверный кластер IBM Netfinity, обеспечивающий поддержку в сети Интернет официального сервера АН РТ.

6. В сети АН РТ установлены несколько серверов баз данных, развернуты средства разработки приложений, включая инструментальные CASE-среды.

В настоящее время в составе ЛВС Институтов и научных Центров АН РТ функционирует 12 серверов и свыше 150 АРМ различной производительности на базе ПК. Кроме того, более 100 персональных рабочих мест академиков и членов-корреспондентов АН РТ на базе домашних ПК имеют доступ в сеть Интернет по коммутируемым каналам.

Экспертная группа, состоящая из основных разработчиков КИС АН РТ, провела анализ оценки уровня ее развития на конец 2005 г. относительно базового исходного состояния (начало 2000 г.). Ниже представлены результаты обработки экспертных оценок по предложенной методике оценки уровня развития СТС.

Результаты вычислений приоритетов частных критериев оценки физического и интеллектуального развития представлены в табл. 6.

Таблица 6

Наименование критерия	Приоритет критерия
Группа критериев физического развития	
Масштаб корпоративной информационной системы	0,23
Число локальных сетей, входящих в ее состав	0,11
Общее число рабочих мест	0,07
Общая длина каналов внутрикорпоративного межсетевого взаимодействия	0,03
Число центров обработки данных	0,1
Суммарная производительность центров обработки данных	0,14
Число центров хранения данных	0,03
Суммарная емкость центров хранения данных	0,03
Средняя пропускная способность локальных сетей	0,03
Средняя пропускная способность каналов внутрикорпоративного межсетевого взаимодействия	0,09
Число шлюзов к внешним сетям	0,03
Суммарная пропускная способность каналов доступа к внешним сетям	0,11
Группа критериев интеллектуального развития	
Число предоставляемых ИТ-услуг (общесистемных и прикладных сервисов)	0,05
Качество предоставляемых ИТ-услуг	0,13
Общий объем накопленных корпоративных информационных ресурсов	0,11
Объем мультимедийных информационных ресурсов	0,03
Средняя интенсивность внутреннего трафика	0,03
Средняя интенсивность внешнего трафика	0,09
Уровень интеллектуальности платформы общесистемного управления	0,02
Уровень интеллектуальности информационно-вычислительной среды (ИВТ)	0,02
Уровень интеллектуальности интерфейса с пользователями	0,06
Уровень обеспечения информационной безопасности	0,16
Степень открытости и масштабируемости ИВС	0,04
Уровень надежности и отказоустойчивости	0,25
Интероперабельность ИВС	0,01

В результате проведенного анализа были получены числовые значения индексов физического (*PDI*) и интеллектуального (*IDI*) развития, а также значения коэффициентов устойчивости физического (*SPD*) и интеллектуального (*SID*) развития, которые равны соответственно:

$$PDI = 0{,}629;\ SPD = 1;$$

$$IDI = 0{,}659;\ SID = 1.$$

В соответствии с полученными значениями вид развития системы определяется как глобально-устойчивое прогрессивное.

Анализ процесса модернизации КИС АН РТ за 2001—2005 гг. на основе оценки уровня ее развития показал, что наибольшие изменения произошли по следующим основным направлениям:

- повышение производительности центров обработки данных;
- рост емкости памяти центров хранения и управления данными;
- увеличение общего числа АРМ;
- увеличение объема накопленных информационных ресурсов;
- возрастание интенсивности внешнего сетевого трафика.

Проведенный анализ позволил выявить также те направления развития КИС, которые достаточно значимы для повышения ее эффективности и которым было уделено недостаточное внимание в ранее реализованных проектах. При разработке предложений по развитию КИС АН РТ было рекомендовано уделить особое внимание таким направлениям ее совершенствования, как повышение качества предоставляемых ИТ-услуг, увеличение надежности и отказоустойчивости, повышение уровня информационной безопасности.

Полученные результаты анализа КИС АН РТ позволили повысить уровень обоснованности принимаемых проектных решений за счет выбора наиболее приоритетных направлений ее развития на ближайшую перспективу.

Заключение

Предложенная методика оценки уровня развития СТС носит достаточно универсальный характер и характеризуется надежными способами получения экспертной информации. Она позволяет получить количественную оценку уровня развития любой СТС, отнести процесс развития к определенному виду и может быть полезна при разработке и исследовании эффективных механизмов управления целенаправленным развитием сложной системы.

Список литературы

1. **Дружинин В. В., Конторов Д. С.** Проблемы системологии (проблемы теории сложных систем). М.: Радио и связь, 1976.
2. **Флейшман Б. С.** Основы системологии. М.: Радио и связь, 1982.
3. **Мартин Дж.** Планирование развития автоматизированных систем. М.: Финансы и статистика, 1984.
4. **Дружинин В. В., Конторов Д. С.** Системотехника. М.: Радио и связь, 1985.
5. **Половинкин А. И.** Основы инженерного творчества. М.: Машиностроение, 1988.
6. **Клир Дж.** Системология. Автоматизация решения системных задач. М.: Радио и связь, 1990.
7. **Волкова В. И., Денисов А. А.** Основы теории систем и системного анализа. СПб.: Изд. СПбГТУ, 2001.
8. **Мелешко В. Н.** Математическое моделирование процессов сбалансированного развития сложных систем // Информационные системы. 1999. № 10. С. 9—13.
9. **Воронцов В. Л.** Подход к планированию процесса усовершенствования технической системы по экономическим показателям // Приборы и системы. Управление, контроль, диагностика. 2000. № 5. С. 76—80.
10. **Исмагилов И. И., Зиновьев П. А.** Многокритериальная оценка уровня развития сложных технических систем // Исследования по информатике. Вып. 8. Казань: Отечество, 2004. С. 25—32.
11. **Саати Т.** Принятие решений. Метод анализа иерархий: Пер. с англ. М.: Радио и связь, 1993.

УДК 004.02

Д. М. Жук, канд. техн. наук, доц.,
А. В. Андронов,
МГТУ им. Н. Э. Баумана

Задачи автоматизации управления жизненным циклом изделия в рамках процессного подхода к управлению

Рассмотрены задачи автоматизации управления жизненным циклом изделия, актуальные для российских средних и малых предприятий. Освещены предпосылки комплексной автоматизации, особенности развития предприятий. Проанализирована взаимосвязь между основными и управляющими бизнес-процессами жизненного цикла изделия. Приведены принципы формирования законченных продуктов управляющих бизнес-процессов и их роль в общем процессе создания изделия. Сформулированы основные задачи автоматизации управления жизненным циклом изделия, в качестве базы для итеративного развития автоматизации принята модель технологической зрелости предприятия. Рассмотрены пути решения поставленных задач автоматизации управления жизненным циклом изделия.

Введение

Информационные технологии развиваются очень быстро. Повсеместная автоматизация, коснувшаяся в первую очередь финансовых структур, актуальна и для производственных предприятий. Развитие CAD/CAM/CAE спровоцировало скачкообразный рост эффективности труда, сейчас автоматизированные системы используются при изготовлении широкого спектра технически сложных промышленных изделий. Многие российские средние и малые предприятия, выпускающие такие изделия, сталкиваются с необходимостью уменьшения времени выхода на рынок и сокращения издержек. Для них характерны очаговая автоматизация и отсутствие ресурсов для полной реорганизации.

Развитие идей комплексной автоматизации

Традиционно проводилась автоматизация отдельных подразделений предприятия без планирования последующей интеграции. В связи с этим типичной картиной является наличие систем класса ERP, CAD, CRM и других, функционирующих независимо [7]. Это ведет к дублированию и искажению информации, увеличивает время поиска информации и принятия решений. В настоящее время многие предприятия заинтересованы в переходе от очаговой автоматизации к комплексной [9]. Для производственных предприятий большой интерес представляет построение единой информационной системы на основе автоматизации жизненного цикла изделия (ЖЦИ).

Теоретически комплексная автоматизация хорошо обоснована, существуют международные и российские разработки в этой области. Основными принципами создания автоматизированной системы обработки информации (АСОИ) определены системное единство, развитие, совместимость и стандартизация. Эти принципы можно распространить и на системы обработки информации, охватывающие все этапы жизненного цикла изделия, называемые системами CALS (*Continuous Acquisition and Lifecycle Support* — непрерывная информационная поддержка поставок и жизненного цикла), или ИПИ (информационная поддержка изделий) [8]. Кроме того, существуют стандарты для обмена данными, такие как STEP для обмена CAD-данными [4]. CALS-технологии в первую очередь связаны с электронным документооборотом и опираются на следование взаимодействующих систем стандартам обмена электронными документами [6]. В большинстве случаев они применяются на этапе проектирования.

Практической реализацией идей интеграции стало появление PDM-систем (*Product Data Management* — управление данными об изделии). Этот класс систем охватил электронный документооборот преимущественно на стадии проектирования, объединяя и упорядочивая CAD-данные [12]. Следующим шагом развития PDM явилась связь проектирования с производством. Поскольку автоматизированная система вышла за пределы одного подразделения, возникли проблемы, связанные с взаимодействием: новые технологии не учитывались в стандартных процедурах обмена. Успешное внедрение PDM-системы оказалось напрямую связано с реорганизацией документооборота и пересмотром рабочих процессов, вовлеченных во взаимодействие [18].

Развитие идей автоматизации связей между различными этапами ЖЦИ привело к появлению систем PLM (*Product Lifecycle Management* — управление жизненным циклом изделия). По сути PLM-система является объединяющим решением для существующих на предприятии систем (например, CAD + CAM + CAE + PDM + ERP), формируя единое информационное пространство и обеспечивая обмен данными. Ведущие производители формируют полностью интегрированное PLM-решение, в которое входят CAD/CAM/CAE-компоненты и собственная PDM-система. Внедрение такой системы затрагивает все основные информационные потоки предприятия, что приводит к

необходимости пересмотра деятельности предприятия, связанной с выпуском изделия. Сопряжение внедряемых и действующих систем является серьезной проблемой, во многих случаях возникает необходимость в отказе от использовавшихся ранее решений из-за проблем совместимости [14].

Переход от очаговой автоматизации к комплексной

Комплексная автоматизация жизненного цикла на сегодняшний день предполагает формирование единого информационного поля, использование совместимых подсистем и реорганизацию существующих процессов, в первую очередь в области документооборота. Внедрение сопряжено с решением ряда задач, актуальных практически для любого производственного предприятия с очаговой автоматизацией:

1. Оценка существующей инфраструктуры, выбор имеющихся и новых систем, которые войдут в единое информационное пространство, определение границ автоматизации;
2. Пересмотр процессов документооборота и рабочих процессов, связанных с использованием автоматизированных систем.
3. Внедрение, сопряженное с установкой и настройкой систем, переносом информации.
4. Обучение персонала использованию новых систем и процессов.

Решение перечисленных задач позволяет полностью перейти к использованию новой системы. К сожалению, на многих предприятиях выполняют только п. 3, иногда поверхностно затрагивая п. 1. Это связано в первую очередь с тем, что поставщики PLM-систем предлагают типовые комплексные решения и не заинтересованы в сохранении особенностей деятельности конкретного предприятия. Пересмотр рабочих процессов входит в задачи реинжиниринга, выполняемого консалтинговыми компаниями. Обучение персонала (п. 4) традиционно считается неперспективным вложением несмотря на большой объем публикаций и успешных решений, восходящих к принципам качества Э. Деминга, на основе которых японская промышленность заняла лидирующие позиции на рынке [1].

Автоматизация для среднего и малого бизнеса

Решение задачи комплексной автоматизации сопряжено с существенными затратами. Крупный производственный бизнес (в первую очередь авиа- и автопроизводители) в состоянии оплатить внедрение полностью. Производители программного обеспечения (ПО) учитывают их потребности, кроме того, крупным бизнесом управляют достаточно компетентные руководители, осознающие сложность проблем перехода.

Иначе обстоит дело у представителей среднего и малого бизнеса. Обладая ограниченными ресурсами, они принимают решение об автоматизации самостоятельно, без всесторонней оценки его эффективности [15]. Производители PLM-систем активно продвигают свои продукты на рынок SMB (*Small and Midsize Business* — малый и средний бизнес), хотя и признают, что большинство предприятий SMB не осознает своих реальных потребностей в области PLM [17]. В результате анализ и реинжиниринг предприятия откладываются, и осуществляется попытка внедрить новую систему в условиях старой организации работ, что практически неосуществимо. Проблемы, связанные с организационной сферой, возникают уже после начала внедрения, что приводит к непредсказуемым последствиям [2, 13].

Примеры успешных внедрений напрямую связаны с корректной оценкой потребностей предприятия и выбором функций PLM-системы, которые необходимы в первую очередь. Использование этих функций результативно только в случае их полной интеграции в ЖЦИ. Более того, предприятия сначала изменяют ход работ, используя различные средства, а затем уже переходят к использованию функций PLM-системы [19].

Современное успешное предприятие должно инвестировать в развитие и переход на новые технологии, чтобы оставаться успешным в будущем [1]. Для производственного предприятия развитие в первую очередь сопряжено с управлением ЖЦИ. В рамках долгосрочного управления вносятся изменения в состав и характер работ в рамках ЖЦИ, изменения носят существенный характер (например, внедрение системы контроля качества) [1, 5]. Текущее управление связано, прежде всего, с планированием, сбором и оценкой актуальной информации и коррекцией [1].

Для предприятий, выпускающих продукцию мелкой серией или несерийно, сложно полностью автоматизировать ЖЦИ, поскольку состав работ может изменяться от изделия к изделию. Вместо этого целесообразно автоматизировать управление ЖЦИ, включая в это понятие не только обмен информацией, но и изменение состава работ в рамках этапов ЖЦИ.

Этапы ЖЦИ

Процесс создания конечного изделия разбивается на этапы, в рамках которых выполняются отдельные виды работ. На многих предприятиях традиционно сложившаяся организационная структура соответствует классическому ЖЦИ:

каждый этап выполняется в рамках самостоятельного подразделения.

В соответствии с [20] ЖЦИ разбивается на следующие этапы: маркетинговые исследования, составление технического задания, проектирование, технологическая подготовка производства, изготовление, поставка, эксплуатация, ремонт, утилизация. В состав каждого из этапов ЖЦИ входит определенная последовательность операций, которая может существенно различаться для различных предприятий и изделий. Кроме того, одни и те же операции на различных предприятиях могут относиться к различным этапам, причем реально выполняемые работы могут отличаться от планируемых.

Очаговая автоматизация в большинстве случаев не выходит за рамки этапа ЖЦИ. Изменяются время выполнения отдельных работ и форма представления результатов. Переход на использование автоматизированных систем без изменения принципов документооборота и оценки трудозатрат вносит путаницу в ход работ ЖЦИ. Руководители подразделений постепенно корректируют свою деятельность, отказываясь от части функционала системы для решения проблем неавтоматизированных подразделений.

Принцип "черного ящика" не срабатывает, поскольку между этапами ЖЦИ существует множество связей, выстраиваемых "по месту" так, как это удобно исполнителям, без учета общей эффективности работ. Необходимо управлять связями между этапами ЖЦИ, а это требует изменения подхода к управлению.

Процессный подход к управлению

Современным подходом к организации деятельности предприятия является процессный подход к управлению. ***Бизнес-процесс*** (***процесс***) — устойчивая, целенаправленная совокупность видов деятельности, которая по определенной технологии преобразует сырье (исходные данные и ресурсы) в продукты (генерируемые информационные и материальные ресурсы), представляющие ценность для клиента. ***Процессный подход к управлению*** — управление организацией путем построения системы процессов, управления ими, осуществления деятельности по улучшению процессов [10].

Группы консультантов или внутренние подразделения формируют схемы работы предприятия, предлагая на их основе усовершенствование отдельных этапов или пересмотр деятельности предприятия в целом. В большинстве случаев оказывается, что формальное описание существующих бизнес-процессов сильно затруднено, а на основе неформальной модели сложно проводить реорганизацию. Во многих случаях попытки описания текущей структуры (*as is*) и формирования на ее основе будущей (*to be*) заканчиваются неудачей, особенно в случае использования идеологии "сквозных" бизнес-процессов [1].

Исследования в области реорганизации бизнес-процессов выявили типичную ошибку: процессы рассматриваются в отрыве от управления. Причин несколько: во-первых, традиционные подходы к организации ЖЦИ не рассматривают управление в принципе, оно лежит за пределами состава работ по выпуску изделия и, следовательно, не попадает в сферу рассмотрения; во-вторых, описать процессы управления существенно сложнее, чем процессы преобразования материальных ресурсов. Отрицательный результат дает попытка наложить обновленный процесс создания продукта на существующую систему управления [10].

Управление процессом — определение системы целей и показателей, необходимых для измерения достижений [10]. Таким образом, управление определяется информацией, поступающей в процесс вместе с исходными данными и сырьем. Например, график производства является информацией, с помощью которой управляют производством.

ЖЦИ в терминах процессного подхода

Цепочка создания ценности (ЦСЦ) — организованный и взаимосвязанный набор бизнес-процессов, создающих ценность для клиентов [10]. Поскольку ЖЦИ рассматривает создание изделия с низкой степенью детализации, можно воспользоваться схемами ЦСЦ для изображения процессов, входящих в этот цикл. Для определенности рассмотрим выпуск нового изделия, для которого применимы перечисленные этапы ЖЦИ.

Если предположить, что каждый этап жизненного цикла соответствует бизнес-процессу, то для первых двух этапов (исследование и разработка) схема будет выглядеть, как показано на рис. 1.

Как показано на схеме, бизнес-процессы преобразуют исходные данные в значимые результаты, однако такая схема не отражает информационные потоки, управляющие преобразованием.

Чтобы учесть управление, требуется описать дополнительные бизнес-процессы, поставляющие управляющую информацию для основных процессов. Управляющей будет являться информация, на основании которой основной бизнес-процесс может быть запущен, остановлен или скорректирован. Например, на этапе проектирования такой управляющей информацией будет план выпуска технического проекта (рис. 2).

Таким образом, управление, абстрактное в терминах классического жизненного цикла, в рамках описания бизнес-процессов приобретает вполне

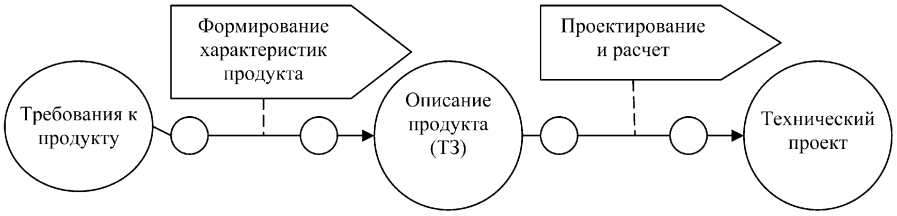


Рис. 1. Исследование и разработка

определенный характер: процесс "проектирование и расчет" управляется с помощью информации "план выпуска технического проекта". Управляющий процесс, в свою очередь, может получать информацию от вышестоящих процессов, такую как "общий план выпуска изделия". Отличительными чертами управляющего процесса является создание управляющей информации в качестве основного результата и периодичность выполнения.

Формальное изображение управляющей связи ничего не говорит о реальном взаимодействии (формы документов, периодичность обмена и т. д.) — эта информация может быть зафиксирована на детализированных схемах, например, на схеме потока работ. Вместе с тем, это дает возможность изменять способы взаимодействия, не нарушая логики взаимодействия процессов.

Исходя из принципов процессного подхода основной бизнес-процесс должен выполняться без постоянного вмешательства извне, только на основе данных, производимых управляющим процессом. Следовательно, эти данные должны быть непротиворечивы и достаточны для эффективного выполнения процесса. Теоретически можно выявить все основные информационные связи, управляющие выпуском изделия. На практике возможна идентификация только основных связей, непосредственно используемых в повседневной деятельности. Определяя зависимости бизнес-процессов, соответствующих отдельным этапам ЖЦИ, можно выявить все основные информационные связи, управляющие выпуском изделия.

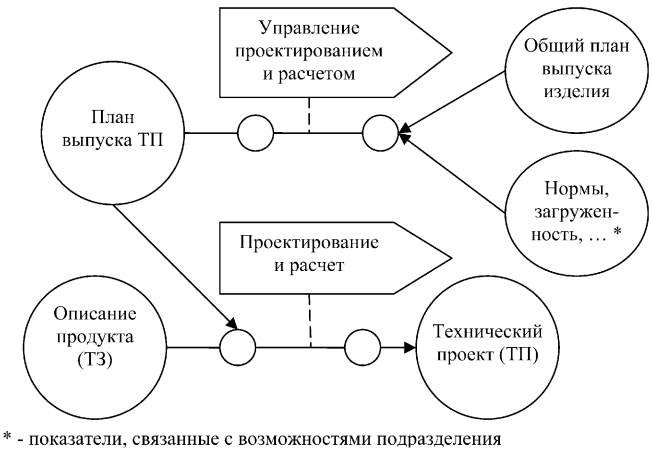


Рис. 2. Управление проектированием и расчетом

Дробление продуктов бизнес-процесса

При мелкосерийном и несерийном производстве значительная часть информации, поступающей на вход управляющих процессов, определяется по ходу работы. Например, сроки поставки материалов в общем случае могут быть определены только после проектирования и расчета. Как правило, общие сроки не позволяют выполнять работы последовательно, поэтому важно определять минимально необходимый набор информации, который дает возможность запустить следующий процесс с приемлемой степенью риска. При этом основной бизнес-процесс должен выполняться итерационно, производя составные части общего выходного продукта в виде законченных элементов. Критически важным в данном случае становится принцип закрытости бизнес-процесса от внешнего вмешательства со стороны процесса-потребителя: наружу должны выдаваться только законченные и проверенные результаты, за которые участники производящего процесса несут ответственность. Отсутствие промежуточных результатов и наличие ряда законченных составных частей продукта позволяют существенно снизить риск неверной работы последующих процессов, что в большинстве случаев предпочтительней незначительного увеличения скорости выполнения.

Формально дробление основного продукта бизнес-процесса удобно описывать в виде таблицы, в которой определены потребители составных частей продукта. Такие таблицы должны учитывать специфику рассматриваемого предприятия и не могут носить абстрактный характер. Например, для предприятия, занимающегося остеклением, будет актуально разбиение результата проектирования и расчета (технического проекта) приведенных в табл. 1.

Продукты представлены в порядке появления на выходе процесса. Важно отметить зависимость между ними: если результаты расчетов не согласованы, последующие продукты не могут быть утверждены и использованы в работе.

Продукты 2—4 используются одним потребителем. Для управления поставками важно как можно раньше сделать заказ большей части номенклатуры, от поставки которой зависит производство. Раннее утверждение основной номенклатуры накладывает существенные огра-

Таблица 1

Результаты проектирования и расчета

№	Продукт	Потребитель
1	Результаты расчетов	Согласование проекта
2	Основная номенклатура профилей	Управление поставками
3	Формула стеклопакетов	Управление поставками
4	Полная номенклатура	Управление поставками
5	Комплект чертежей	Подготовка производства

ничения на последующую разработку документации, однако позволяет соблюсти сроки производства.

Задачи автоматизации управления ЖЦИ

Проанализировав совокупность управляющих бизнес-процессов и информационных связей, можно сформулировать основные задачи, решение которых направлено на автоматизацию управления ЖЦИ. Эти задачи должны решаться последовательно, предыдущая должна формировать базу для последующих. Развитие автоматизации соотносится с принципами модели технологической зрелости предприятия (*Capability Maturity Model, CMM*). CMM была разработана для оценки степени зрелости процессов разработки ПО и охватывает весь жизненный цикл создания программ. В настоящее время модель зрелости трактуется достаточно широко, в частности, основные понятия CMM используются в рамках менеджмента качества для оценки направленности изменений.

CMM описывает пять уровней зрелости: начальный, воспроизводимый, определенный, управляемый, оптимизированный [11].

Эти уровни можно взять в качестве базы для разделения задач автоматизации управлением ЖЦИ (табл. 2).

Очаговая автоматизация. Этот уровень работы непредсказуемый, с высоким риском. Ему также соответствуют предприятия, не проводящие автоматизацию или находящиеся на начальных стадиях очаговой автоматизации. Каждое предприятие изначально находится на подобном уровне, выявление реальных потребностей в комплексной автоматизации управления ЖЦИ возможно только после начала реальной работы с автоматизированными системами.

Таблица 2

Соответствие уровней CMM задачам автоматизации управления ЖЦИ

№ уровня	CMM	Задачи автоматизации управлением ЖЦИ
1	Начальный	Очаговая автоматизация
2	Воспроизводимый	Построение единого информационного поля
3	Определенный	Формализация и регламентация управляющих связей
4	Управляемый	Автоматизированное выполнение задач в рамках управляющих бизнес-процессов
5	Оптимизированный	Автоматизированное управление бизнес-процессами

Рассмотрим последующие уровни подробнее.

Построение единого информационного поля. Объединение информации, получаемой в результате выполнения различных бизнес-процессов, позволяет сократить дублирование и повысить достоверность принимаемых решений. Как правило, единое информационное поле также обеспечивает группировку документов и совместную работу.

Задача объединения информации связана не только со сбором и обработкой существующих документов, но и с переходом на новые принципы создания и хранения данных. В первую очередь изменения касаются исполнителей, разрабатывающих и использующих документацию. От них требуется корректное использование единого хранилища документов, контроля версий, цепочек согласования и других функций, обеспечивающих достоверность информации.

Важной составляющей единого информационного поля является наличие стандартов, регламентирующих порядок именования документов и содержащейся в них ключевой информации. Разработка таких стандартов — трудоемкий итерационный процесс, практически невозможно получить идеальную систему с первого раза.

Технической реализацией единого информационного поля является развертывание системы документооборота. Предпочтительным можно назвать внедрение одной из систем PDM, хотя многие предприятия не используют функции, отличающие PDM от других систем документооборота, такие как формирование сборок и работа с цифровой моделью. В ряде случаев целесообразно использование временной системы, простой в использовании и обладающей только базовыми возможностями, поскольку требуется длительное время на обучение и внедрение стандартов, а дорогостоящие функции начнут использоваться много позже.

Формализация и регламентация управляющих связей. В рамках данной задачи должны быть описаны основные бизнес-процессы предприятия и определено управляющее воздействие. Модель должна быть построена на достаточно высоком уровне, главная ее цель — выявить ключевые зависимости между бизнес-процессами и корректно определить потребителей управляющей информации.

Построение моделей и попытки описания текущей деятельности предприятия в большинстве

случаев выявляют организационные проблемы и "узкие места" при взаимодействии различных подразделений. С позиций построения идеальной системы можно исправлять все эти несоответствия, однако такая работа может занять длительное время и не всегда заканчивается успехом. Для большинства средних и малых предприятий достаточно фиксации текущего состояния и исправления проблем, непосредственно связанных с управляющими информационными потоками.

Основным процессам ЖЦИ должна регулярно поступать достоверная управляющая информация. Несмотря на первоначальную трудоемкость должна быть выработана базовая схема взаимодействия, которая позволяет четко отслеживать любые отклонения от нормы и не вмешиваться в обычный ход работ.

Постепенная оптимизация управляющих связей должна основываться на практическом опыте управления основными бизнес-процессами, а не на оценках эффективности, проводимых в сжатые сроки и, как правило, поверхностно.

Автоматизированное выполнение задач в рамках управляющих бизнес-процессов. Такие задачи, как планирование сроков и стоимости, контроль исполнения, сбор информации и ряд других могут решать специализированные программные системы.

Автоматизация должна учитывать наличие связей между различными управляющими процессами и поддерживать единое информационное поле. Кроме того, схожие задачи в рамках различных процессов должны решаться схожими методами. Модель управления процессами должна быть пересмотрена с точки зрения автоматизации: переход к использованию компьютерных систем существенно изменяет сроки и состав работ, добавляет новые возможности. Изменения в модели управления должны основываться на принципах постоянного улучшения, например, на основе системы менеджмента качества ИСО 9000. В рамках управляющих процессов не должны появляться задачи, источником которых является новое программное обеспечение. Вместе с тем, не должно быть дублирующих задач, выполняемых "для проверки". После завершения опытной эксплуатации должен осуществляться полный переход на новые системы: если персонал не доверяет системе или не может в полной мере перейти на ее использование, значит, должны быть пересмотрены принципы ее использования или проведены дополнительные настройка и обучение.

В рамках одного процесса могут присутствовать как автоматизированные, так и неавтоматизированные задачи. Важно формализовать переход между ними и исключить дублирование работ: дублирование информации в данном случае неизбежно. Для таких переходов необходимо предусмотреть быстрый перенос данных и процедуру итеративного обновления. Детальное рассмотрение таких случаев может послужить причиной для расширения границ автоматизации: например, вместо перевыпуска полного комплекта бумажных чертежей для производства в случае внесения изменений, можно поставить рабочие места для просмотра конструкторской документации в электронном виде и перевыпускать только те чертежи, которые запросит производство.

Для автоматизации управляющих бизнес-процессов используются в первую очередь системы календарного планирования и документооборота. Исходя из специфики рассматриваемых предприятий такими системами можно считать Microsoft Office Project и систему PDM, обеспечивающую базовые функции прохождения документов.

Потенциально решение задачи автоматизации отдельных работ ведет к решению задачи автоматизации бизнес-процесса в целом. Тем не менее, именно рассмотрение отдельных задач дает возможность оптимизировать взаимодействие и использовать преимущества автоматизирующих систем для улучшения модели управления бизнес-процессами предприятия.

Автоматизированное управление бизнес-процессами. Для управляющих бизнес-процессов с устоявшейся структурой возможна полная автоматизация, при этом вмешательство требуется для ввода недостающих исходных данных или реагирования на отклонения, не запланированные в рамках процесса.

Опыт, накопленный в ходе автоматизации отдельных составляющих бизнес-процессов, позволяет делать выводы о возможности полной автоматизации. При этом необходимо отделять автоматизацию выполнения бизнес-процессов от принятия управленческих решений: в любом случае предприятие не может функционировать само по себе, без текущей управленческой деятельности. Вместе с тем, появляются новые возможности для прогнозирования и стратегического планирования: для автоматизированных бизнес-процессов проще оценивать ресурсы и задавать граничные условия.

Автоматизированное управление бизнес-процессами — результат длительной деятельности по автоматизации предприятия, настройке и доработке программного обеспечения и обучению персонала. Переход на этот уровень возможен только для высокотехнологичных предприятий, нацеленных на постоянное улучшение бизнес-процессов. Для многих предприятий этот уровень недостижим по причине постоянного изменения выпускаемой продукции: нерентабельно оптимизировать процессы, которые вскоре устареют. Некоторые процессы невозможно полностью авто-

матизировать из-за высокой доли неопределенности или других причин. Иногда стоимость полной автоматизации существенно выше стоимости автоматизации отдельных задач и неприемлема для предприятия. Таким образом, данный уровень автоматизации является для среднего и малого бизнеса скорее отдаленной целью, чем насущной необходимостью. Вместе с тем, постепенное движение в данном направлении позволяет вести постоянную оптимизацию управляющих процессов и периодически оценивать перспективы дальнейшего развития.

Список литературы

1. **Абдикеев Н. М.** Реинжиниринг бизнес-процессов. М.: Эксмо, 2007.
2. **Воскресенская Е. А., Степанов А. В.** Опыт внедрения PLM-системы на промышленном предприятии. Ч. 1. "CAD/CAM/CAE Observer". 2005. № 4(22).
3. **Елиферов В. Г., Репин В. В.** Бизнес-процессы: Регламентация и управление. М.: ИНФРА-М, 2006.
4. **Зубинский А.** CAD-тенденции // Компьютерное обозрение. 2004. № 15(433).
5. **Зубков В.** Концепция улучшения качества в технике // Стандарты и качество. 2007. № 3.
6. **Левин А., Судов Е.** CALS — сопровождение жизненного цикла // Открытые системы. 2001. № 3.
7. **Макаров С.** Точка зрения на хаос. К выбору модели комплексной автоматизации // Компьютерра. 2002. № 22.
8. **Павлов В. В.** Структурное моделирование в CALS-технологиях. М.: Наука, 2006.
9. **Проблемы** автоматизации управления предприятием // BYTE. 2006. № 9.
10. **Репин В. В.** Бизнес-процессы компании: построение, анализ, регламентация. М.: РИА "Стандарты и качество", 2007.
11. **Ройс У.** Управление проектами по созданию программного обеспечения. Унифицированный подход. М.: Лори, 2002
12. **Ширяев Н.** Критерии сравнения систем TDM/PDM // САПР и графика. 2002. № 1, 2.
13. **Ширяев Н.** Некоторые аспекты внедрения PLM-решения при комплексной автоматизации предприятия // САПР и графика. 2005. № 1.
14. **Ширяев Н.** CALS, PDM, PLM, далее — везде... // САПР и графика. 2002. № 12.
15. **Elliot L.** PLM-системы: подходит ли один масштаб для всех? Ч. I. Взгляд аналитиков рынка PLM-систем // CAD/CAM/CAE Observer. 2006. № 3(27).
16. **Elliot L.** PLM-системы: подходит ли один масштаб для всех? Ч. II. Взгляд поставщиков PLM-продуктов // CAD/CAM/CAE Observer. 2006. № 4(28).
17. **Elliot L.** PLM-системы: подходит ли один масштаб для всех? Ч. III. Взгляд пользователей PLM-продуктов из сферы SMB // CAD/CAM/CAE Observer. 2006. № 5(29).
18. **Miller Ed, MacKrell J.** Интеграция PLM- и ERP-систем. Ч. I. // CAD/CAM/CAE Observer. 2006. № 1(25).
19. **Orr J.** PDM — ожидания и реальность // CAD/CAM/CAE Observer. 2006. № 6(30).
20. **Р 50.1.031—2001.** Рекомендации по стандартизации. Информационные технологии поддержки жизненного цикла продукции. Терминологический словарь. Ч. 1. Стадии жизненного цикла продукции. Госстандарт России, 2001.

СТРАНИЦЫ ИСТОРИИ

УДК 004

В. В. Шилов, канд. техн. наук, проф.,
"МАТИ"—РГТУ имени К. Э. Циолковского,
Москва
shilov@mati.ru

"Некий Томас Хилл..."

История вычислительной техники — не только история машин; это не в меньшей степени история людей, создававших машины. При этом сделанное ими надо оценивать в контексте своего времени. Вряд ли стоит снисходительно относиться к достижениям, сколь бы наивными или незначительными ни выглядели они сегодня. И уж тем более не стоит выказывать эту снисходительность, если нам мало что о них известно.

В одной из статей об истории вычислительной техники мне встретилась фраза: " ...некий Томас Хилл". Слово *некий*, означающее "тот, о ком ничего не известно", в современном русском языке имеет совершенно четкий пренебрежительный оттенок. Сразу же подумалось — а можно ли кого бы то ни было называть "неким"? Если кто-то не слышал, к примеру, об Аристотеле, это еще не основание для высказывания "...как говорил некий Аристотель...".

Итак, попробуем разобраться, кем же был упомянутый выше "некий" Томас Хилл.

* * *

Сегодня, взятая в ретроспективе, история вычислительной техники зачастую предстает несколько упрощенной, "линейной". Например, о знаменитом арифмометре Тома де Кольмара (изобретенном около 1820 г.), давно находящемся в центре внимания историков, принято отзываться исключительно как об эпохальной вехе. Его историю представляют едва ли не как триумфальное шествие по миру. Однако при ближайшем рассмотрении такая картина не кажется бесспорной. На самом деле первые пятьдесят (!) лет своего существования арифмометр де Кольмара передовым творением техники отнюдь не считался...

И, в частности, одной из причин, которая не позволяла арифмометру получить более широкое распространение, было неудобство работы на нем. Особенно остро стояла проблема ввода чисел. Начиная с суммирующей машины Паскаля, числа устанавливали, непосредственно вращая счетные колеса, с помощью штифта или, как в арифмометре де Кольмара, двигая специальные установочные кнопки.

В то же время идея ввода информации с помощью нажатия клавиш буквально носилась в воздухе. Так, она уже была востребована создателями первых пишущих машин[1]. Правда, о самых ранних пишущих машинах неизвестно практически ничего. Например, мы не знаем устройство пишущей машины, запатентованной 7 января 1714 г. английским инженером Генри Миллом (*Henry Mill*, 1683—1771). То же самое можно сказать о пишущих машинах Фридриха фон Кнаусса (*Friedrich von Knauss*, 1724—1789), по некоторым данным построенными около 1770 г., и жителя городка Фивиццано итальянца Агостино Фантони (*Agostino Fantoni*), построенной около 1802 г. Тем не менее, вполне допустимо предположить, что клавиатура была частью их конструкции. Во всяком случае, последующие изобретатели клавиатуру использовали. Первым из них стал американец Уильям Остин Берт (*William Austin Burt*, 1792—1858), получивший 23 июля 1829 г. патент на печатающую машину (*Typographer*). Несколько позднее были изобретены машины Чарльза Тёрбера (*Charles Thurber*, 1803—1886), патент № 3228 от 26 августа 1843 г. (рис. 1), Альфреда Эли Бича (*Alfred Ely Beach*, 1826—1896), патент № 15164 от 24 июня 1856 г. (рис. 2) и др. Аналогичные проекты появлялись и в Европе. Таким образом, к середине XIX в. конструкторами пишущих машин идея использования клавиш для передачи символьных и цифровых данных уже была опробована на практике.

И в это же время возможностями клавишного ввода заинтересовались создатели арифметических машин. Так, 5 февраля 1850 г. Дюбуа Пармели (*Du Bois D. Parmelee*) из Нью-Пальтца, шт. Нью-Йорк, получил патент № 7074 на одноразрядную суммирующую машину с *клавишным вводом* (рис. 3). Это была первая в истории вычислительной техники попытка использовать ввод чисел с клавиатуры.

Сумматор Пармели имеет девять клавиш *c*, соответствующих цифрам от 1 до 9. Нажатие клавиши опускает рычаг *E*, закрепленный на шарнире *f*, и через собачки *m* и *k* вызывает перемещение зубчатой рейки *B* вверх на

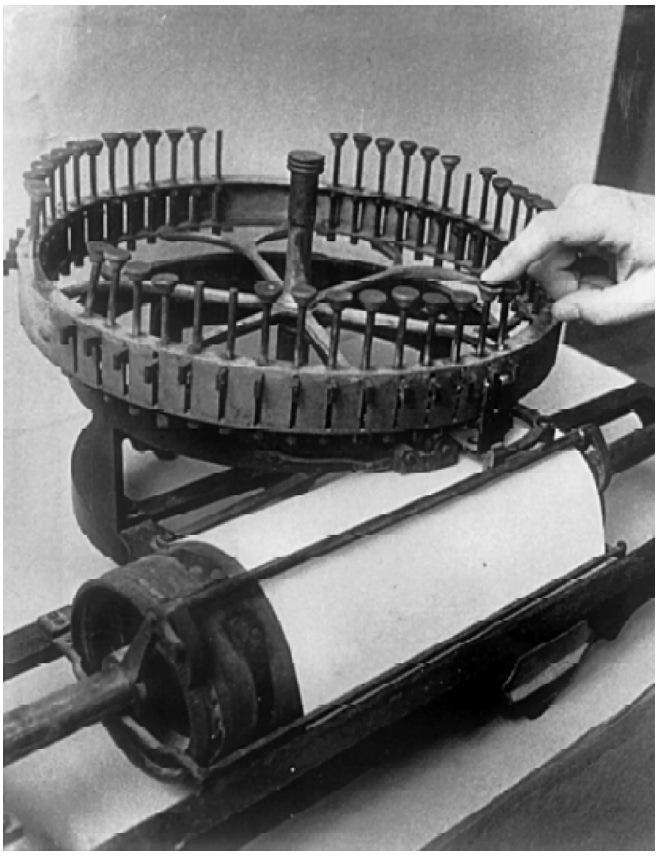


Рис. 1. Пишущая машина Чарльза Тёрбера (1843 г.)

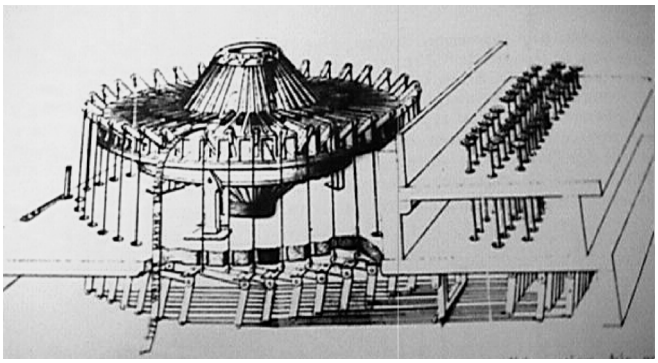


Рис. 2. Пишущая машина А. Эли Бича (1856 г.)

определенное число позиций. Так, при нажатии клавиши 4 рейка *B* выдвинется на четыре позиции, и поскольку возле каждого зубца написан его номер, будет зафиксировано число 4. Если теперь нажать клавишу 7, то рейка выдвинется еще на семь позиций, показав сумму 11. Когда рейка выдвинется вверх на всю длину, полученную сумму следует зафиксировать на бумаге для последующего использования, и освободить рейку, потянув за кольца *p* и *p′*, посредством проволок *o* и *o′*, связанных с собачками *k* и *m*. Поскольку рейка *B* при этом упадет до упора вниз, машину при работе необходимо ставить на самый край стола! Конкретную длину рейки Пармели не указывает, говоря, что она может быть произвольной.

Скорее всего, машина Пармели так и не была построена. Но практически одновременно с ним

[1] Пишущие машины практически не попадают в поле зрения историков вычислительной техники. А между тем их влияние на развитие механических счетных машин было весьма значительным.

свою клавишную суммирующую машину предложил Виктор Шильт (*Victor Schilt*) из Швейцарии[2]. Во время Всемирной выставки 1851 г. в Лондоне она экспонировалась в Хрустальном дворце и даже получила Похвальный отзыв жюри. Машина Шильта стала, таким образом, первым действительно изготовленным вычислительным устройством с клавишным вводом данных (к сожалению, ее конструкция в литературе до сих пор не описана). В настоящее время она находится в коллекции Национального музея американской истории в Вашингтоне.

Однако одноразрядные машины Пармели и Шильта, хотя и продемонстрировали принципиальную возможность клавишного ввода, серьезного практического значения иметь все-таки не могли. Для использования на практике были необходимы многоразрядные машины, так что требовалось решить проблему переноса из разряда в разряд[3]. И первым, кто попытался ее решить, стал американец Томас Хилл (рис. 4).

Томас Хилл родился в Нью-Брансвике (шт. Нью-Джерси) 7 января 1818 г. Его отец, Томас Хилл-старший (1771—1828) эмигрировал из Англии в 1791 году. Причиной этого послужили религиозные притеснения, которым подвергались в бывшей метрополии приверженцы некоторых направлений протестантизма. В Новом свете он попробовал заниматься фермерством, а затем завел кожевенное дело. Судя по тому, что Хилл-старший исполнял обязанности судьи главного суда

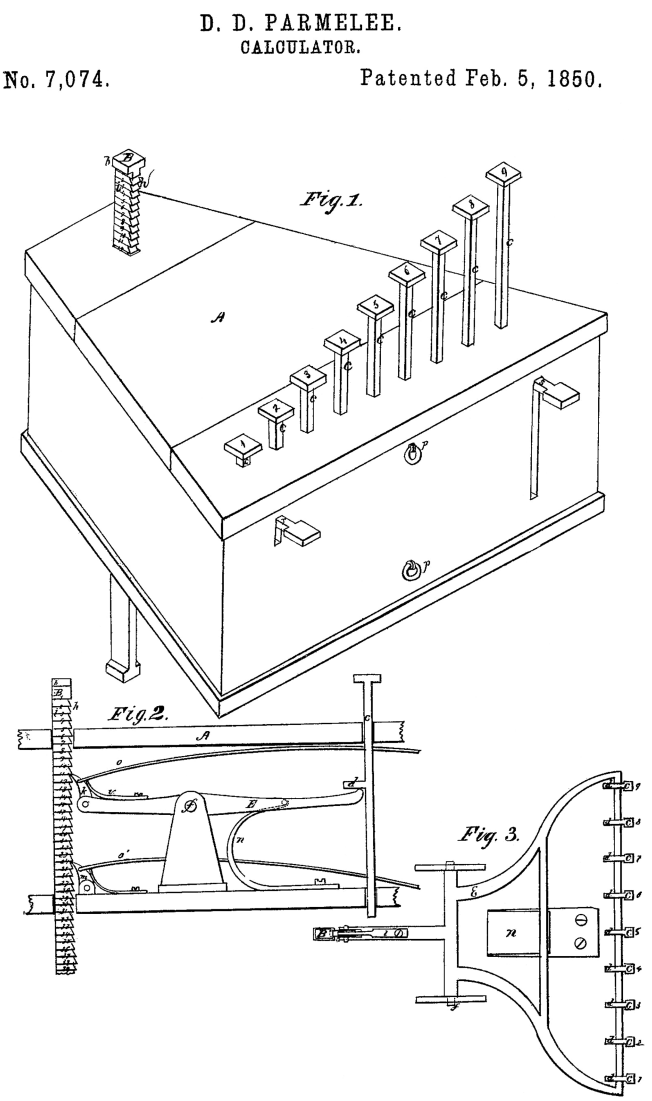


Рис. 3. Клавишный сумматор Дюбуа Пармели (чертеж из патента, 1850 г.)

первой инстанции по гражданским делам, он был в местном обществе человеком уважаемым.

Отец сыграл большую роль в формировании личности сына. Он любил природу и поощрял интерес ребенка к естественным наукам, увлекательно рассказывая ему о растениях и их свойствах. Возможно, именно беседы с отцом пробудили и развили в мальчике любознательность. В детстве Хилл практически не посещал школу, читать и писать его научили мать и сестры[4]. Овладев грамотой, Томас стал усердным читателем. Он рано начал интересоваться ботаникой, философией и математикой, а к двенадцати годам уже прочел труды Бенджамина Франклина и Эразма Дарвина.

Мать Хилла, Генриетта (урожденная Баркер), умерла в 1824 г. в возрасте 50 лет. Спустя несколь-

[2] В отечественной литературе его (скорее всего, ошибочно) называют англичанином (см., напр., [2, с. 62]). Во всяком случае, на Всемирной выставке в Лондоне в 1851 г. суммирующая машина Шильта (экспонированная под № 59) представляла Швейцарию. Отметим, что никакие биографические сведения о Шильте неизвестны. Зато биографию Пармели в общих чертах мне удалось восстановить. Он родился в 1830 г., в 1858 г. окончил университет Нью-Йорка как врач и химик. Пармели был весьма заметным в свое время химиком и изобретателем: я сумел разыскать еще семь его патентов, выданных в период с 1858 г. по 1872 г. (ни один из них не имеет отношения к вычислительной технике). В патентах №№ 21697 и 26551 описываются способ и аппаратура для производства изделий из каучука. Они длительное время применялись в американской промышленности, а сам Пармели до конца жизни оставался консультантом ряда химических компаний. В патентах №№ 108510, 114191 и 128166 предлагается способ электролитического покрытия внутренней поверхности металлических труб серебром, никелем и др., а в патенте № 91962 — способ утилизации жестяных отходов посредством их спрессовывания в цельные болванки. Но, пожалуй, самое важное из изобретений Пармели — новый способ крепления механических протезов конечностей (патент № 37637, 1863 г.) с помощью прижимаемых атмосферным давлением резиновых муфт (вместо прежних застежек и завязок), впоследствии ставший общепринятым. Дюбуа Пармели умер в 1897 г.

[3] Пармели, понимая ограниченность возможностей своего сумматора, указал в патенте, что в качестве индикатора получаемой суммы вместо рейки можно использовать диск или "несколько дисков, один для десятков, другой для сотен и т. д." Однако соответствующего механизма он так и не предложил.

[4] Хилл имел двух братьев и четырех сестер.

Рис. 4. Томас Хилл

ко лет скончался и отец, и десяти лет от роду Хилл остался сиротой. С 1830 по 1833 гг. он работал учеником у местного печатника, а следующий год провел вместе со старшим братом в Академии Нижнего Дублина (*Lower Dublin*), местечка в округе Филадельфия (шт. Пенсильвания), с 1854 г. вошедшего в черту города. В это время он начал задумываться о специальности инженера-строителя, но необходимость зарабатывать на жизнь заставила его вернуться в родной город, где Хилл прожил еще три года, работая помощником аптекаря.

Однако юноша жадно тянулся к знаниям, и, оставив в 1838 г. работу в аптеке, Хилл начал готовиться к поступлению в колледж. В 1839 г. он стал студентом Гарвардского колледжа и в 1843 г. окончил его со степенью бакалавра гуманитарных наук. Во время учебы Хилл зарекомендовал себя как один из самых выдающихся студентов (в своем выпуске он имел второй результат). Его учителями были крупнейший американский математик и астроном Бенджамин Пирс (*Benjamin Peirce*, 1809—1880) и известный физик и астроном Джозеф Лаверинг (*Joseph Lovering*, 1813—1892), будущий президент Американской ассоциации содействия развитию науки. Так что неслучайно именно математика и астрономия особенно привлекали Хилла: еще будучи студентом, он получил интересные результаты в исследовании кривых высших порядков.

В 1843 г. в Гарварде была организована обсерватория, которую возглавил известный астроном Уильям Бонд (*William Cranch Bond*, 1789—1859). Вероятно, опыт астрономических наблюдений и практические советы Бонда помогли Хиллу создать несколько новых астрономических инструментов. Наибольшую известность из них получил *оккультатор*[5], с помощью которого вычислялись моменты начала и окончания солнечных затмений для различных широт. Оккультатор Хилла длительное время использовался на практике, — так, с его помощью были получены таблицы для затмений марта 1854 г., мая 1857 г.[6] и др. Расчеты с помощью оккультатора не требовали большого времени и обеспечивали вполне приемлемую точность. Например, составление таблицы для мая 1857 г. заняло всего 6 часов, а ошибка в вычислениях колебалась в пределах от 12 секунд (Вашингтон) до 1 минуты (Сан-Франциско). 13 октября 1842 г. Комитет по науке и искусствам Института Франклина в Филадельфии рекомендовал наградить создателя инструмента медалью Скотта — высшей наградой Института[7]. Через несколько недель другая комиссии того же института рекомендовала наградить оккультатор Почетным отзывом (*A Certificate of Honorary Mention*) 12-й Выставки товаров американских производителей (в то же время, по мнению комиссии, некоторые другие изобретения были достойны серебряной медали).

Вопрос о том, был ли Хилл награжден медалью Скотта, далеко не прост. В работе [5] Хилл назван в числе лауреатов за 1842 г. (с. 422). Это же утверждается во многих других публикациях, в том числе в монографии "Столетие американской математики"[8]. В то же время на официальном сайте Института Франклина сказано, что в 1835—1848 гг. медаль Скотта вообще не присуждалась. Скорее всего, причина разночтений заключается в том, что статус награды в то время еще окончательно не устоялся.

---

[5] Оккультация (от лат. *occultatio* — покрытие) — прохождение небесного тела между наблюдателем и более далеким небесным телом, которое при этом становится совсем или частично невидимым.

[6] См. письмо Хилла редактору: The Astronomical Journal. Vol. IV. № 18.9 January 1856. P. 138—139.

[7] Комиссия отметила, что "инструмент должен быть в высшей степени полезен тем, кто желает наблюдать затмения, но не знаком с использованием логарифмов и тригонометрических формул". Медаль Скотта, которая сегодня является одной из самых престижных научных наград США, вручается с 1822 г. и была учреждена на средства, оставленные по завещанию эдинбургского аптекаря Джона Скотта (*John Scott*,? – 1815). До 1918 г. размер денежной части премии составлял 20 долларов. Историю награды см. в работе [5].

[8] A Century of M athematics in America (Ed. By P. Duren). Part I. AMS, 1988. P. 6.

Так что отнюдь не случайно по окончании курса Хилл получил предложение занять пост директора Национальной обсерватории. Предлагали Хиллу также остаться преподавателем в колледже. Однако точные науки все-таки не стояли у Хилла на первом месте. Получившего блестящее классическое образование Хилла (среди прочих, он знал древнееврейский и другие восточные языки) более влекла сфера гуманитарная.

В 1843 г., еще будучи студентом, Хилл напечатал за свой счет в Кембридже (шт. Массачусетс) небольшую книжечку "*Christmas, and Poems on Slavery*". Она была посвящена выдающейся активистке движения против рабства Элизе Ли Фоллен (*Eliza Lee Follen*, 1787—1860) и, помимо заглавного ("Рождество"), включала четыре стихотворения аболиционистской направленности. Книга предназначалась для продажи в благотворительных целях на рождественской ярмарке в Бостоне, — такая форма сбора средств на нужды движения за отмену рабства была тогда достаточно распространенной. Однако сборник Хилла серьезно выделялся из литературы такого рода своими незаурядными художественными достоинствами. Особенную популярность приобрели два его стихотворения, "Материнская молитва" и "Беглец". Они так и остались высшим достижением Хилла в поэзии, хотя он на протяжении всей жизни продолжал писать и публиковать стихи как лирического, так и духовного содержания (их наиболее полное собрание "*In the Woods and elsewhere*" увидело свет в Бостоне в 1888 г.).

В 1845 г. Хилл окончил также Гарвардскую школу богословия (которая научных степеней в то время не присваивала) и принял приход унитарианской церкви в Волтхэме, шт. Массачусетс[9]. 14 лет протекли в занятиях наукой, пастырской службе и размышлениях. В этот период он опубликовал такие книги, как "Геометрия и вера. Дополнение к девятому Бриджуотерскому трактату" (1849 г., переиздана в 1874 и 1882 гг.), "Первые уроки геометрии" (1854 г.), "Иисус, толкователь природы, и другие проповеди" (1859 г.). Так, в первой из них (созданной под влиянием идей Чарльза Бэббиджа, изложенных в его "Девятом Бриджуотерском трактате"; в предисловии Хилл пишет, что выбором названия он хотел подчеркнуть свое восхищение сочинением английского ученого) Хилл доказывал, что наука и религия не противоречат друг другу, — напротив, наука подтверждает Божьи законы. По его мнению, науке противостоит лишь искаженная форма религии, которую недобросовестные служители церкви превратили в политическое орудие[10]. Свои взгляды Хилл излагал в многочисленных статьях в религиозных, математических и астрономических изданиях, он принимал участие в различных энциклопедиях, а также занимался разработкой новых методов обучения. Именно в этот период утвердилась репутация Хилла как серьезного ученого, литератора и педагога.

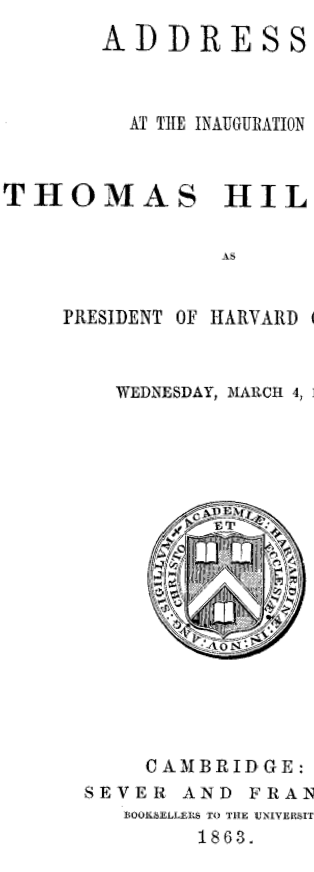
Такая жизнь полностью отвечала его склонностям, но в 1859 году в ней произошел резкий поворот. Хилл всегда сознавал исключительную роль образования для общества. "История Америки началась с основания Антиохийского колледжа", — подчеркивая эту роль, написал он как-то одному из друзей. По стечению обстоятельств именно Антиохийский колледж в Йеллоу-Спрингс (шт. Огайо) ему довелось возглавить спустя несколько лет. Это популярное учебное заведение переживало непростые времена — постоянная нехватка средств сопровождалась ожесточенной борьбой между фракциями унитарианцев и "христиан"[11] в попечительском совете. Для выхода из кризиса требовался человек, способный встать над схваткой, и кандидатура Хилла была сочтена наиболее подходящей для этого. Воззрения Хилла отличались значительным либерализмом, и он действительно попытался избавить колледж от сектантской ограниченности (так, при нем в колледже обучались студенты, принадлежавшие к семи конфессиям). Хилл поставил перед собой цель превратить колледж в одно из ведущих учебных заведений страны. Однако решить нараставшие финансовые проблемы ему так и не удалось, религиозные разногласия также не прекращались. Все это, в конце концов, вынудило Хилла оставить колледж.

Предвидя скорую отставку, весной 1862 г. Хилл принял предложение занять пост президента Гарвардского колледжа. 1 декабря он приступил к исполнению новых обязанностей (церемония инаугурации, состоявшаяся 4 марта следующего года, во всех подробностях описана в изданной университетом брошюре, рис. 5). Хилл надеялся реформировать сложившуюся в университете систему обучения, и отчасти это ему удалось. Так, он

[9] Унитарианство — либеральное движение в протестантизме, признающее единоличие Бога в лице Бога-отца и не принимающее доктрину триединства (троицы). Как конфессия, унитарианство оформилось в Англии в середине XVII в. и жестоко преследовалось (так, закон о смертной казни за отрицание догмата троицы был отменен только в 1813 г.). В Америке официально действует с 1796 г. Никогда не будучи многочисленной, унитарианская церковь имела огромное влияние, а ее членами были крупнейшие американские интеллектуалы — ученые, писатели, педагоги и политики, включая пять президентов США: Джона Адамса, Томаса Джефферсона, Джона Куинси Адамса, Милларда Филмора и Уильяма Тафта.

[10] Доктрина универсализма предполагает возможность модификации учения в соответствии с полученными в ходе развития науки данными.

[11] "Христиане" (*Christian Connexion*) — одна из ветвей протестантизма.


ADDRESSES

AT THE INAUGURATION OF

THOMAS HILL, D. D.,

AS

PRESIDENT OF HARVARD COLLEGE,

WEDNESDAY, MARCH 4, 1863.

CAMBRIDGE:
SEVER AND FRANCIS,
BOOKSELLERS TO THE UNIVERSITY.
1863.

Рис. 5. Обложка брошюры, выпущенной в Гарварде по случаю инаугурации Т. Хилла (1863 г.)

сумел существенно поднять уровень требований к поступающим. Именно во время его президентства была введена элективная система и студенты получили возможность выбирать курсы из нескольких предлагаемых, а для поддержки малоимущих учащихся были установлены стипендии. При факультетах колледжа и профессиональных школах были учреждены Академические советы. Появились новые направления обучения, — по инициативе Хилла в Гарварде были открыты кафедры геологии и горного дела. Деятельность Хилла получила большой общественный резонанс, и в период президентства в Антиохии и Гарварде он превратился в фигуру общенационального масштаба. Тогда же его научные заслуги были отмечены степенями доктора богословия (Гарвард, 1860 г.) и доктора права (Йель, 1863 г.)

Биографы Хилла и историки Гарвардского университета расценивают его президентство как достаточно успешное [3, 4]. Но в целом в эти годы Хилл не был счастлив. В колледже была весьма сильна оппозиция его политике, и реализация его программы встречала постоянное сопротивление. Трудностей в повседневном управлении добавлял мягкий характер Хилла, — административные способности у него почти полностью отсутствовали. Далеко не всем нравилась и присущая Хиллу простота поведения. Так, Хилл был заядлым садовником (посаженные им деревья до сих пор растут в парке Антиохийского колледжа), и его привычка самому перекапывать в саду землю, не прибегая к помощи садовника, многими расценивалась как неподобающая для президента...

Крайне трудным этот период был для Хилла и в личной жизни. В 1864 г. скончалась его жена Энн, с которой они прожили 19 лет (у них были два сына и четыре дочери[12]). Спустя два года он обвенчался с Люси Шепард, однако второй брак оказался непродолжительным. Вскоре у его супруги открылась неизлечимая болезнь, унесшая ее в 1869 г. в возрасте 32 лет.

Все эти обстоятельства, а также пошатнувшееся здоровье привели Хилла к решению подать в отставку, которая состоялась 30 сентября 1868 г. После двухлетнего отдыха Хилл вернулся к активной общественной жизни и в течение 1871 г. представлял Волтхэм в Законодательном собрании штата. Затем по предложению своего друга и коллеги по Гарвардскому университету, знаменитого естествоиспытателя и путешественника Жана Луи Родольфа Агассиса (*Jean Louis Rodolphe Agassiz*, 1807—1873), он принял участие в его научной экспедиции к побережью Южной Америки[13].

Найдя по возвращении из экспедиции, что его здоровье окончательно нормализовалось, Хилл принял предложение возглавить "Первый приход" в Портленде, шт. Мэйн (рис. 6) — один из крупнейших и наиболее влиятельных приходов унитарианской церкви. Это были самые спокойные, умиротворенные годы его жизни. Простые прихожане обожали своего пастора, а у портлендской аристократии Хилл пользовался непререкаемым моральным авторитетом. Он писал[14], выступал с проповедями и лекциями, проводил различные педагогические эксперименты и ставил научные опыты. В Портленде Хилл прослужил до самой смерти, последовавшей 21 ноября 1891 г. после длительной болезни.

---

[12] Их старший сын, Генри Баркер Хилл (Henry Barker Hill, 1849—1903) стал известным химиком, членом Национальной Академии наук США.

[13] Агассис, уроженец Швейцарии, был учеником Жоржа Кювье. Его важнейшие работы относятся к эмбриологии, сравнительной анатомии, палеонтологии и др. Агассис разработал метод тройного параллелизма, сыгравший важную роль в доказательстве теории эволюции. При этом сам он был убежденным сторонником креационизма и непримиримым противником теории эволюции Дарвина. Он считал, что виды неизменны и свои доводы против эволюции изложил в "Очерках классификации" (1859 г.). Хилл полностью разделял эти взгляды своего друга и коллеги и неоднократно высказывал их в печати и в публичных лекциях.

[14] Среди работ этого периода — учебник "Практическая арифметика" (1881 г.) и многочисленные статьи.

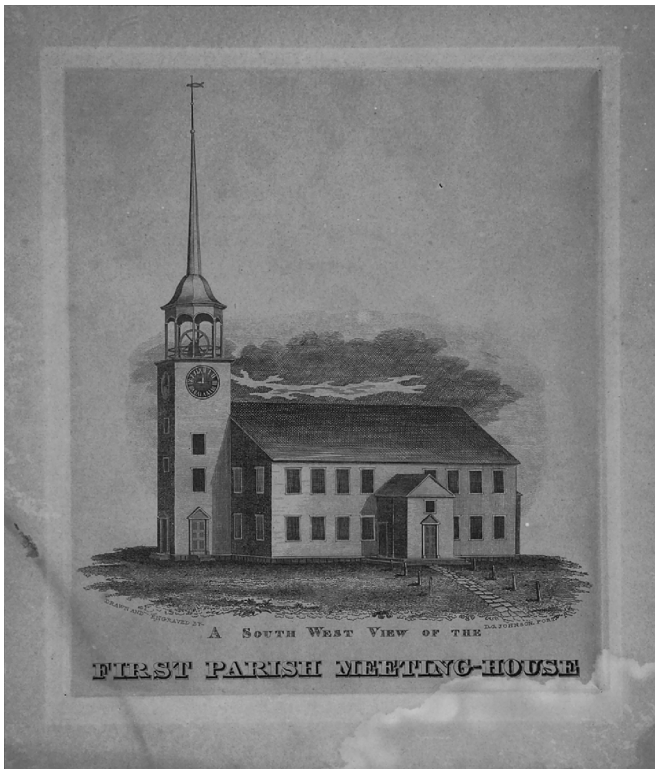


Рис. 6. Молитвенный дом Первого прихода в Портленде (рис. 1825 г.)

Итак, даже краткий очерк жизни Томаса Хилла убедительно свидетельствует о том, что это действительно был незаурядный человек, оставивший заметный след в интеллектуальной истории своей страны. Но нас особенно интересует вклад Хилла в развитие вычислительной техники. И здесь следует сказать, что интерес к проблеме вычислений сопровождал Хилла всю жизнь. Вряд ли приходится сомневаться в том, что Хилл, со студенческих лет активно занимавшийся астрономией, обратился к проблеме вычислений именно в это время. Собственно, уже оккультатор был прибором, призванным упростить проблему вычислений применительно к конкретной задаче.

То же самое относится и к другому прибору, изобретенному Хиллом в более поздний период, — речь идет об аналоговом морском навигационном приборе, которому автор дал название *наутригон* (*nautrigon*). Как известно, после начала эпохи великих географических открытий на протяжении нескольких столетий едва ли не основной проблемой практической навигации оставалось точное определение долготы в открытом море. Более того, временами решение этой задачи становилось одним из центральных вопросов не только мировой науки, но и мировой политики. Были предложены десятки, если не сотни, различных методов, включая самые экзотические[15], но только после появления морского хронометра Гаррисона (модель H4, испытанная в 1764 г.) она получила удовлетворительное решение. После этого на повестку дня встало создание метода совместного определения широты и долготы наблюдателя. Мы не будем останавливаться на истории этого вопроса (например, в1808 г. свой метод, основанный на последовательном решении трех сферических треугольников, получаемых из тригонометрических уравнений, предложил К. Ф. Гаусс). Однако полученные Гауссом и другими математиками аналитические решения были слишком сложны для применения на практике. Наибольшее распространение получил так называемый метод Самнера, созданный в 1837 г. и опубликованный спустя 6 лет[16]. Однако он также был сопряжен со значительной вычислительной работой. В то же время достаточно простое графическое решение этой задачи с использованием земного и звездного глобусов, известное с XVI в., не обеспечивало необходимой точности (для нахождения местоположения судна с точностью от 1 до 2 мин был необходим глобус диаметром до 10 м). Именно для нахождения приближенного решения этой задачи посредством метода Самнера и был предназначен наутригон (Хилл также называл его *самнером* в честь изобретателя метода), описание которого было издано в Портленде в 1876 г. в виде 24-страничной брошюры[17]. Можно предположить, что изобретение наутригона стало следствием участия Хилла в экспедиции Агассиса, во время которой он впервые непосредственно соприкоснулся с практикой кораблевождения.

И, наконец, нельзя не упомянуть еще одно изобретение Хилла, также связанное с вычислениями, на этот раз календарными. В архиве библиотеки Гарвардского университета хранится действующая модель "вечного" календаря, изготовленная Хиллом около 1885 г. (рис. 7).

Но конечно самое важное из его изобретений, относящихся к вычислениям, — это первая в ис-

---

[15] См., например, роман знаменитого семиотика и писателя-постмодерниста Умберто Эко "Остров накануне" (1994 г.).

[16] A New and Accurate Method of Finding a Ship's Position at Sea, by Projection on Mercator's Chart" (Boston, 1843). Экземпляр этой брошюры, в течение 20 лет выдержавшей не менее шести изданий, в обязательном порядке должен был находиться на каждом американском судне. Томас Самнер (*Thomas Hubbard Sumner*, 1807—1876) учился в Гарварде, по семейным обстоятельствам ушел в море и свое открытие сделал во время одного из плаваний. Судьба Самнера сложилась трагично, последние 26 лет жизни он провел в сумасшедшем доме.

[17] "Description of the Nautrigon, with Directions for Its Use; an Instrument for Solving Problems in Navigation, ..." На титульном листе указано, что наутригон был запатентован 22 февраля 1876 г., однако, согласно справке архива Бюро патентов США, в этот день такой патент зарегистрирован не был. Поскольку запросы к этому архиву возможны только по дате или по номеру (который нигде в литературе не приводится), ознакомление с текстом патента оказалось невозможным. К сожалению, недоступной осталась и упомянутая брошюра — это крайне редкое издание, согласно каталогам, сохранилось лишь в шести экземплярах.

Рис. 7. Механический календарь Т. Хилла (ок. 1885 г.)

тории вычислительной техники *многоразрядная* клавишная суммирующая машина, запатентованная в США 24 ноября 1857 г. (патент № 18692), которую автор назвал *арифмометром* (*Arithmometer*). Слово "многоразрядная" выделено здесь не случайно – дело в том, что в литературе [1, 2] машину Хилла принято называть двухразрядной. На самом же деле, двухразрядной была только изготовленная Хиллом модель (рис. 8).

Механизм клавишного сумматора Хилла показан на рис. 9, взятом из патента.

Основу конструкции составляют закрепленные на оси *B* счетные колеса *C* и *D* (в патенте оговорено, что число счетных колес может быть при необходимости увеличено). По ободу каждого колеса на равном расстоянии нанесены группы цифр от 0 до 9 (число таких групп не оговаривается и зависит от диаметра колес и расстояния между соседними цифрами). Эти цифры используются при сложении и умножении. Рядом с каждой цифрой написано ее дополнение до 9, — меньшего размера, чтобы было проще различать их. Дополнения используются при вычитании и делении.

С правой стороны каждого счетного колеса имеется храповое колесо *a*. Оно управляется собачкой *b*, закрепленной на оси рычага *E*. Пружина *d* прижимает собачку к храповому колесу. Другая пружина, *f*, отжимает рычаг *E*, закрепленный на оси *e*, вверх. Над каждым рычагом *E* помещен ряд из 10 кнопок (клавиш) *i*, пронумерованных от 1 до 0 (на рисунке показаны только шесть из них). Штифты этих кнопок имеют одинаковую длину, так что глубина, на которую они опускаются при нажатии, определяется позицией кнопки над рычагом. Тем самым они отклоняют свободный конец рычага *E* на разное расстояние. В результате, при нажатии клавиши *1* собачка сдвинет храповое колесо на один зубец, и, соответственно, счетное колесо повернется на одно деление. (Число зубцов равно числу цифр на ободе счетного колеса.) При нажатии клавиши *5*, расположенной ближе к закрепленному концу рычага, его отклонение будет больше, и собачка сдвинет храповое колесо на 5 зубцов и т. д. Цифры видны в окошке, имеющемся в корпусе машины.

В верхней части машины имеется закрепленная в корпусе муфта *F* с рычагами *H* и *G*, в которой свободно вращается ось *m*. Справа от счетного колеса *C* и соответствующего храпового колеса на той же оси помещено зубчатое колесо *I*. На нем, на равном расстоянии друг от друга, имеются зубцы *n* в количестве, равном числу групп по 10 цифр на колесе *C*. Когда счетное колесо *C* поворачивается на 10 делений, один из зубцов *n* зацепляет рычаг *H* и поворачивает муфту *F*. Соответственно, поворачивается и рычаг *G*, который толкает собачку *l*, сдвигающую счетное колесо *D* на одно деление. Пружина *p*, закрепленная на корпусе прибора, прижимает собачку *l* к храповому колесу, а также возвращает на место рычаги *H* и *G* после расцепления с зубцом *n*.

Для сложения все счетные колеса устанавливаются так, что в окошке появляются нули. В первом ряду нажимается клавиша, соответствующая первой цифре, затем второй и т. д. Таким образом, будут сложены все единицы, причем будут произведены все необходимые переносы. Затем во втором ряду последовательно нажимаются все клавиши, соответствующие десяткам и т. д. Результат (сумма) отображается в окошке. При вычитании используются маленькие цифры. Колеса устанавливаются так, чтобы в окошке появилось уменьшаемое. Цифры вычитаемого вводятся последовательно так же, как и в случае сложения, и результат читается в окошке.

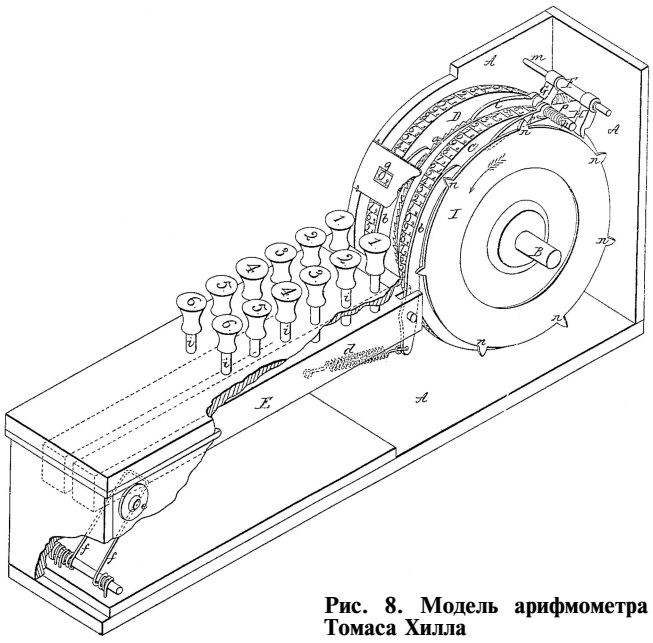


Рис. 8. Модель арифмометра Томаса Хилла

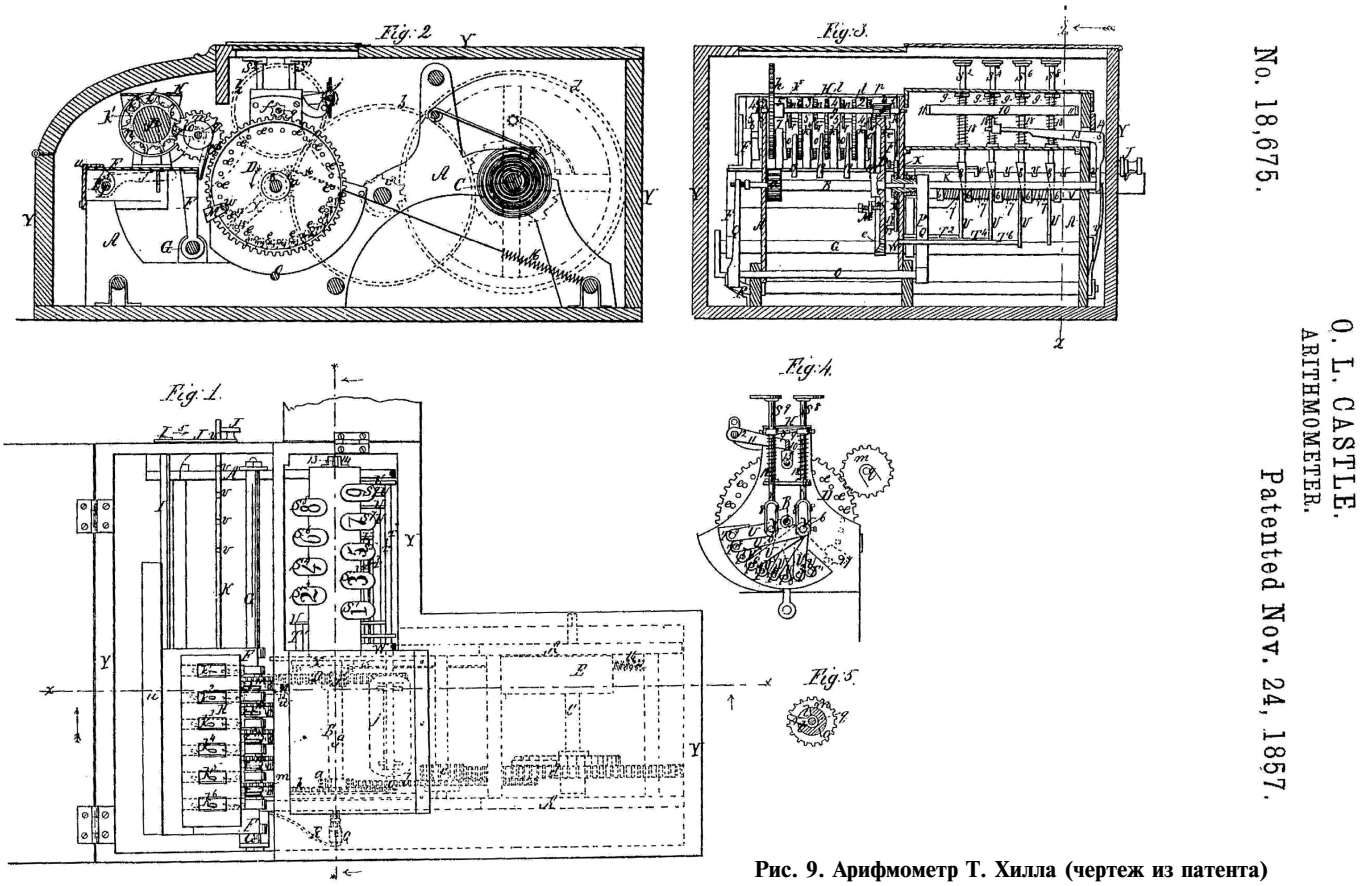


Рис. 9. Арифмометр Т. Хилла (чертеж из патента)

Умножение выполняется как последовательность сложений. Так, чтобы умножить 7 на 3, надо три раза нажать клавишу 7, после чего в окошке появится произведение — 21. Для чисел большей разрядности порядок действий такой же — число устанавливается в окошке, и затем прибавляется к себе требуемое число раз. Аналогично следует действовать при делении (используются маленькие числа). Например, если надо разделить 7 на 2, то следует установить на колесе *C* маленькую цифру 7 и нажимать клавишу 2 (т. е. вычитать 2 из 7) до тех пор, пока в окошке не появится цифра, меньшая 2 (остаток от деления). Целая же часть от деления равна числу нажатий на клавишу, в данном случае — 3.

Первыми в отечественной литературе устройство и принципы функционирования клавишного сумматора Хилла кратко описали Р. С. Гутер и Ю. Л. Полунов [2, с. 63][18] (заметим, что в англоязычной литературе такого описания нет до сих пор). Оценивая машину, они заметили, что она "...имела некоторый успех и была выставлена в Национальном музее в Вашингтоне, однако серьезные конструктивные недостатки, не говоря уже о малой разрядности, помешали ее дальнейшему распространению" [там же]. Эту оценку повторили другие отечественные авторы: "Как двухразрядная машина устройство Хилла было несколько более удобным в работе по сравнению с одноразрядными. Этим и объясняется некоторый успех этой конструкции, уязвимым местом которой была трудность коррекции при неверном или случайном нажатии клавиши" [1, с. 104].

К сожалению, точно определить степень этого успеха сегодня уже трудно. Однако косвенно о нем может свидетельствовать один до сих пор не замеченный факт. Дело в том, что, если говорить строго, то не только патент Хилла претендует на право считаться первым! По удивительному стечению обстоятельств в один день с Хиллом, 24 ноября 1857 г., получил патент на многоразрядную клавишную суммирующую машину, также названную арифмометром, изобретатель из округа Мэдисон (шт. Иллинойс) Орландо Касл (*Orlando Lane Castle*)[19]. Номер этого патента — 18675

[18] В этом описании следует лишь еще раз уточнить, что Хилл не оговаривает разрядность своей машины. Кроме того, он не только не фиксирует число зубьев храпового колеса (в [2] приводится цифра 63), но даже прямо указывает, что оно может меняться в зависимости от размеров колеса.

[19] Еще более удивительно, что в этот же день патент № 18711 на счетное устройство (Calculating Device) был также выдан еще и Джеймсу Смиту (James D. Smith) из Нью-Йорка. Правда, это был не клавишный арифмометр, а круглый карманный сумматор. Так или иначе, это единственный случай в XIX веке, когда в один день трем разным людям были выданы три патента в области вычислительной техники.

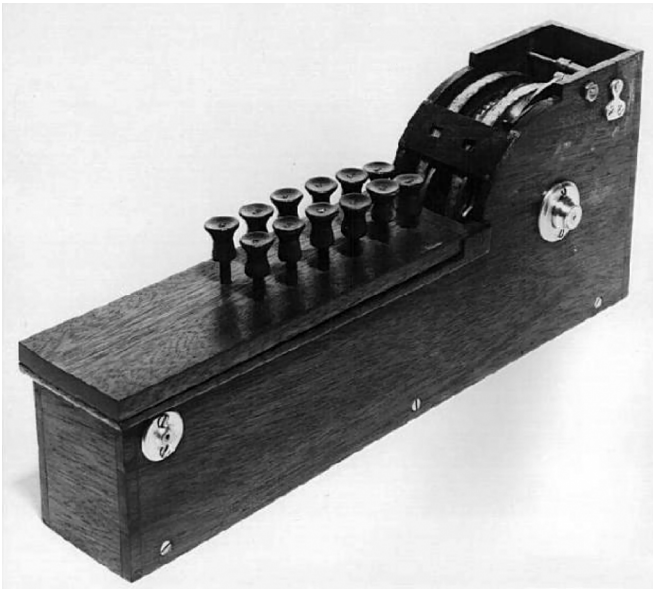


Рис. 10. Арифмометр Орландо Касла (чертеж из патента, 1857 г.)

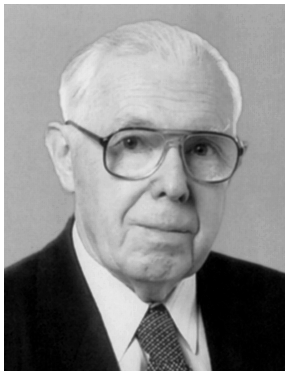
(рис. 10). К сожалению, в то время в патентах еще не было принято указывать дату подачи заявки, так что невозможно сказать, кому из двух изобретателей, Хиллу или Каслу, принадлежит приоритет (номера патентов, выданных в один день, всегда упорядочены по фамилии автора, так что меньший номер обоснованием приоритета служить не может).

Машина Хилла (точнее, ее модель, изготовленная автором для представления в Бюро патентов) действительно привлекла к себе внимание. Это следует хотя бы из того, что мы и сегодня помним о ней. А вот о патенте Касла (даже о патентах, — поскольку ровно год спустя, 2 ноября 1858 г., он получил также патент № 21941 на арифмометр несколько усовершенствованной конструкции) нет упоминаний ни в одной работе по истории вычислительной техники, даже в знаменитой книге Эрнста Мартина [6], которую не случайно называют "Библией арифмометров". И это может означать только одно, — он, в отличие от арифмометра Хилла, внимания к себе не привлек. Конечно, и арифмометр Хилла, судя по всему, так и остался в единственном экземпляре и на практике никогда не использовался. О причинах этого можно только строить предположения. В любом случае, именно интерес, который вызвал в обществе клавишный арифмометр Хилла, мог стимулировать других изобретателей к работе в том же направлении. Однако новые — и уже достаточно успешные — проекты клавишных счетных машин появились только в начале 1870-х годов.

Список литературы

1. **Апокин И. А., Майстров Л. Е.** История вычислительной техники (От простейших счетных приспособлений до сложных релейных систем). М.: Наука, 1990. 264 с.
2. **Гутер Р. С., Полунов Ю. Л.** От абака до компьютера. 2-е изд., испр. и доп. М.: Знание, 1981. 208 с.
3. **Kidwell P. A.** Thomas Hill: Minister, Intellectual and Inventor // Rittenhouse. 1998. Vol. 12. N 48. P. 111—119.
4. **Land W. G.** Thomas Hill, twentieth president of Harvard. Harvard University Press, 1933. 263 p.
5. **Fox R.** The John Scott Medal // Proceedings of the American Philosophical Society. 1968. Vol. 112. N 6. December. P. 416—430.
6. **Martin E.** The Calculating Machines: Their History and Development, ed. P. A. Kidwell and M. R. Williams. 1992. 412 p. (Английский пер. с немецкого издания 1925 г.)

Поздравляем юбиляра!



Главному научному сотруднику Института системного программирования РАН, Заслуженному деятелю науки и техники РСФСР, Лауреату премии Совета министров СССР, Лауреату премии Правительства РФ, доктору технических наук, профессору

Владимиру Васильевичу ЛИПАЕВУ,

известному ученому в области информатики и вычислительной техники исполняется 80 лет.

Талантливый ученый, преподаватель и блестящий организатор Владимир Васильевич Липаев является одним из ведущих специалистов в области автоматизации проектирования программного обеспечения, руководителем разработок ряда крупных инструментальных систем программной инженерии для автоматизации технологических процессов жизненного цикла сложных комплексов и программ специального назначения. Под его руководством выросла большая плеяда молодых научно-технических кадров. Высокие человеческие качества в сочетании с высоким профессионализмом, принципиальностью и ответственностью снискали ему большое уважение учеников и коллег.

Дорогой Владимир Васильевич!
Поздравляем Вас с юбилеем и желаем Вам крепкого здоровья,
большого счастья, новых творческих успехов в Вашей многогранной деятельности!

Редколлегия и редакция журнала.

CONTENTS

Адрес редакции:

107076, Москва, Стромынский пер., 4/1

Телефон редакции журнала **(495) 269-5510**

E-mail: it@novtex.ru

Дизайнер *Т.Н. Погорелова*. Художник *В.Н. Погорелов*.
Технический редактор *О. А. Ефремова*. Корректор *Е. Г Волкова*

Сдано в набор 08.04.2008. Подписано в печать 20.05.2008. Формат 60×88 1/8. Бумага офсетная. Печать офсетная.
Усл. печ. л. 10,78. Уч.-изд. л. 12,72. Заказ 497. 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