

принципиально отличает ее от систем управления техническими системами;

– приведен возможный вариант структуры системы активного управления;

– определены этапы конструирования и исследования системы активного управления;

– даны обучающиеся модели активных процессов. Отмечено, что несмотря на внешнюю схожесть с непараметрическими алгоритмами адаптации, механизмы их работы существенно различаются.

Конкретизация систем управления, обучающихся моделей и алгоритмов принятия решений может быть осуществлена при исследовании реального организационного процесса. На этом пути могут появиться лавинообразные процессы и, естественно, необходимость их моделирования [5] и воздействия на них с теми или иными целями.

Библиографические ссылки

1. Медведев А. В. Теория непараметрических систем. Активные процессы // Вестник СибГАУ. 2011. Вып. 4. С. 52–58.
2. Бурков В. Н. Основы математической теории активных систем. М. : Наука, 1977.
3. Фельдбаум А. А. Принципы обучения людей и автоматов // Кибернетика, мышление, жизнь. М. : Мысль, 1964.
4. Нейман Дж. Фон. Теория самовоспроизводящихся автоматов. М. : Мир, 1971.
5. Medvedev A. V. Nonparametric stochastic approximation in adaptive systems theory // Applied Methods of Statistical Analysis. Simulation and Statistical Inference : proc. of the Intern. Workshop. Novosibirsk : Publishing House of NSTU, 2011. P. 195–214.

A. V. Medvedev

THEORY OF NON-PARAMETRIC SYSTEMS. ACTIVE PROCESSES – II

The paper considers an identification problem and active process control in conditions of partial information, along with management decisions and estimation result of active system influence. The active systems investigation method is offered. It contributes computer modeling. Non-parametric algorithms of decision making are given.

Keywords: active system, identification, a priori information, measurement, N-processes, non-parametric models, non-parametric algorithms, control by discrete and continuous processes, K-models.

© Медведев А. В., 2012

УДК 658.5.011.56:004

М. М. Михнев, К. Н. Поляев

РАЗРАБОТКА ФАЙЛОВЫХ СЕРВЕРОВ ДЛЯ СИСТЕМ АВТОМАТИЗАЦИИ ТЕХНОЛОГИЧЕСКОЙ ПОДГОТОВКИ ПРОИЗВОДСТВА

Рассматриваются проблемы разработки файловых серверов для систем автоматизации технологической подготовки производства, на примере технических решений, реализованных в файловом сервере информационно-поисковой системы средств технологического обеспечения.

Ключевые слова: технологическая подготовка, информационная система, файловый сервер.

Создание информационно-поисковых систем (ИПС) технологической подготовки производства масштаба предприятия, предусматривающих организацию электронного архива технической документации, на этапе проектирования требует принятия решений по организации данных в рамках выбранной для реализации системы управления базой данных (СУБД) и принятия решения о реализации в системе средств для хранения документов в электронном виде и работы с ними.

В мировой практике существуют различные подходы к решению данной задачи, такие как организация доступа к файлам электронных документов средствами применяемой в системе СУБД или использование для работы с файлами выделенного сервера.

Второе решение выглядит более рациональным, так как позволяет обеспечить высокую производительность системы в целом и аппаратно разместить на разных серверах сервер СУБД и файловый сервер, а также, в случае разработки собственного файлового сервера, реализовать требуемую функциональность и учесть специфику выполняемых клиентами операций с файлами.

При разработке очередной версии информационно-поисковой системы средств технологического обеспечения (ИПС СТО), применяемой в технологических подразделениях ОАО «ИСС» имени академика М. Ф. Решетнева» [1–7], для хранения учетных записей электронных документов использовались средства СУБД системы, а для хранения и управления файлами

документов – файловый сервер собственной разработки (рис. 1) [1].

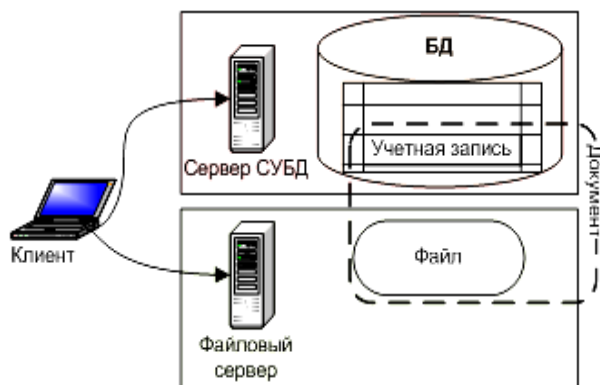


Рис. 1. Организация работы с документами в ИПС СТО

К файловому серверу системы были предъявлены следующие технические требования:

- файловый сервер должен реализовать средства, позволяющие регулировать его загрузку и производительность в зависимости от количества клиентов и характера использования ими ресурсов сервера;
- файловый сервер должен поддерживать иерархическую организацию файлов в виде единого непрерывного пространства, при этом физическое размещение файлов на стороне сервера должно быть прозрачным для клиентов;
- файловый сервер должен обеспечивать работу с большим объемом данных, физически размещенных на различных дисковых накопителях;
- файловый сервер должен обеспечивать максимальную безопасность при любых операциях с данными и в любом случае гарантировать их сохранность.

Средой разработки файлового сервера ИПС СТО стала Delphi 2007 как основная среда разработки всех существующих приложений системы. При создании сервера не привлекались сторонние компоненты, работа с сетью обеспечивалась использованием функций Win API (что позволяет реализовать подобную модель практически в любой среде разработки с минимальными затратами времени и ресурсов). Разработка сервера ИПС СТО проводилась в несколько этапов, результатом которых являлось создание универсальных классов Delphi, реализующих требуемую функциональность для решения поставленной задачи.

Разработка файлового сервера с требуемыми техническими характеристиками включала следующие этапы: 1) разработка универсального класса сокета; 2) разработка универсального класса сервера для протокола TCP/IP; 3) разработка универсального класса файлового сервера; 4) реализация в классе универсального сервера механизмов управления транзакциями.

На первом этапе был разработан универсальный класс сокета, используемый в качестве слушающего сокета сервера; сокета, выделяемого сервером для работы с клиентом; клиентского сокета. Особенно

стью реализации данного класса являлось наличие нескольких «перегруженных» (overloaded) конструкторов, что определялось многоцелевым использованием класса. Сокет работал в неблокирующем режиме, класс реализовал высокоуровневые методы передачи блока данных, передачи строки, передачи любого производного от TStream потока.

На втором этапе был разработан универсальный класс сервера, реализующий основные методы работы с клиентами и управление потоками, обслуживающими клиентские соединения, а также универсальный класс обслуживания одного клиентского соединения. На этом же уровне реализовалась система управления загрузкой сервера. Обработка каждого клиентского запроса выполнялась в отдельном потоке, параллельно с остальными, так же в отдельном потоке обслуживался слушающий сокет и производилось управление клиентскими соединениями.

При разработке системы управления загрузкой сервера для описания внутренних объектов сервера использовались следующие термины.

Слот – ячейка, свободная или занятая, ассоциирующаяся с одним потоком и совокупностью объектов, необходимых для его функционирования, предназначенных для обработки запроса одного клиента.

Пул – совокупность слотов.

Идентификация – комплекс мероприятий, позволяющий однозначно определить валидность клиента, установившего соединение.

Очередь – массив, организованный по принципу FIFO (первый пришел, первый ушел), каждый элемент которого ассоциируется с одним входящим клиентским соединением, прошедшим идентификацию.

Для управления загрузкой сервера требовалось реализовать объекты пула и очереди (рис. 2) и ввести регулируемые ограничения на их максимальный размер.

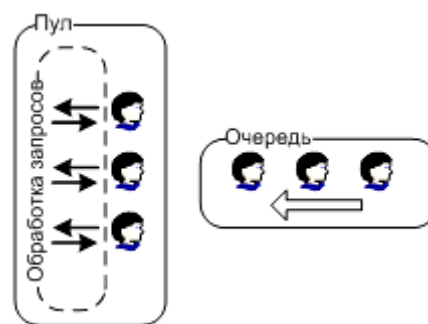


Рис. 2. Организация сервера

Управление подключениями клиентов должно выполняться в отдельном потоке и реализовываться в виде бесконечного цикла, прерываемого инициацией специального события (при завершении работы сервера). Алгоритм обработки клиентских соединений был следующим: а) после события входящего соединения выполняется идентификация клиента, если клиент идентифицирован успешно, выполняется

проверка наличия свободного слота в пуле для немедленной обработки запроса клиента, если клиент не идентифицирован, соединение принудительно разрывается сервером; б) если в пуле имеется свободный слот, инициируется новый поток, которым осуществляется обработка запроса клиента, если в пуле свободных слотов нет, выполняется постановка клиента в очередь; в) при постановке клиента в очередь проверяется наличие в ней свободных мест, если свободное место в очереди есть, клиент добавляется в конец очереди, если места нет, соединение принудительно завершается сервером (рис. 3).

Обработка каждого запроса производится в отдельном потоке, запуск потоков инициируется либо потоком, обслуживающим подключения (при наличии свободных слотов в пуле), либо другим клиентским потоком после завершения выполнения запроса (при извлечении ожидающего обработки соединения из очереди). Алгоритм обработки запросов клиентов следующий: а) клиентский поток запрашивает содержание запроса и немедленно его исполняет, затем инициирует завершение соединения с клиентом; б) перед завершением своей работы клиентский поток проверяет наличие соединений, ожидающих обработки в очереди, при наличии там соединений он извлекает из очереди первое соединение по порядку и инициирует его обработку, после чего завершается (рис. 4).

На третьем этапе был разработан универсальный класс файлового сервера, обеспечивающий, во-пер-

вых, доступ к файлам, физически размещенным в различных местах (на разных дисковых накопителях, разных директориях и т. д.), как к единому файловому архиву; во-вторых, безопасные методы работы с файлами, исключающие потерю данных по причинам обрыва сетевого соединения или же отказа аппаратных средств клиента. Также был минимизирован риск потери данных при аппаратных отказах сервера.

С этой целью для организации доступа к файлам применялись два уровня абстракции: уровень клиента и уровень сервера. Для обращения к определенному файлу клиент указывает полный путь к нему в файловом архиве (в терминах файлового архива), сервер определяет, в каком из имеющихся корневых контейнеров (локаций) размещается данный файл (в терминах локаций), и только для физического доступа к файлу, файловый сервер использует физический путь (в терминах файловой системы) (рис. 5).

Локациями были названы абстрактные контейнеры, физически отображающиеся на диск или директорию на диске сервера. Каждая локация имеет уникальный абсолютный путь в файловой системе сервера, вложенность локаций не допускается. Основное назначение локаций – организация файловых архивов, представляемых сервером в виде непрерывного логического пространства, физически распределенных по разным дисковым накопителям файлового сервера, общим объемом превышающих объем каждого из имеющихся дисковых накопителей сервера.

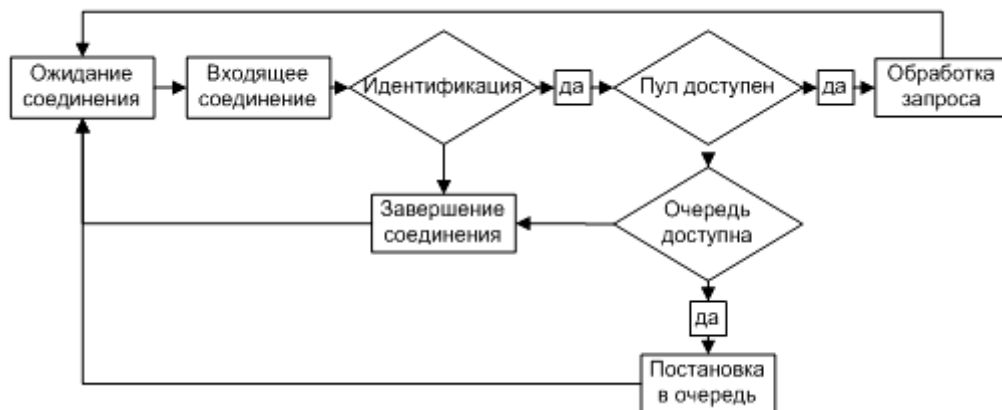


Рис. 3. Обработка входящих соединений сервером ИПС СТО



Рис. 4. Обработка запросов сервером ИПС СТО

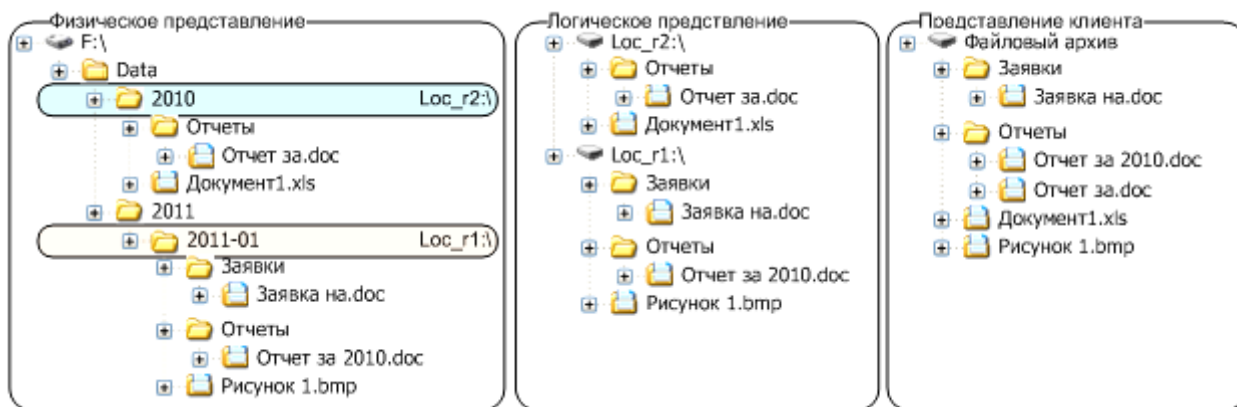


Рис. 5. Организация доступа к файлам

Такой подход позволяет расширить файловый архив на новый дисковый накопитель после того, как объем имеющихся в системе накопителей будет исчерпан (рис. 6).

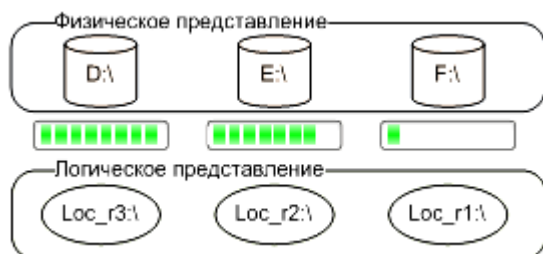


Рис. 6. Организация файлового архива при помощи локаций

Кроме того, локации могут быть использованы для структурирования файлового архива, например для хранения файлов, созданных за какой-то определенный период времени, что полезно при большом количестве файлов и упрощает работу администраторов при обслуживании сервера (рис. 7).



Рис. 7. Использование локаций для структурирования файлового архива

Локация может быть доступной для чтения или записи. Файловый сервер может использовать неограниченное количество локаций для чтения и только одну локацию для записи. Локация, используемая для записи, должна обязательно иметь соответствующую ей локацию для чтения (рис. 8).

При поиске требуемого файла файловый сервер просматривает известные локации для чтения в последовательности, определенной специальным списком (рис. 9).

В начале списка рекомендуется размещать локацию для чтения, соответствующую локации для записи, что позволит повысить скорость доступа к последним файлам, которые обычно востребованы в первую очередь.

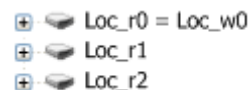


Рис. 8. Локации для чтения и записи



Рис. 9. Поиск файла файловым сервером

На четвертом этапе были разработаны безопасные методы работы с файлами, исключающие возможность потери данных в случае обрывов сетевого соединения и аппаратных отказов как на стороне клиента, так и на стороне сервера, для чего был использован механизм транзакций, широко применяемый в современных базах данных и операционных системах. Транзакции использовались только для запросов, связанных с приемом сервером данных и их записью на дисковые устройства сервера, а также с удалением данных на стороне сервера.

В ходе выполнения запроса сервер обеспечивал выполнение требуемых запросом операций на временном наборе данных, затем выполнял операцию актуализации полученных результатов на реальные данные и отправлял клиенту подтверждение завершения транзакции. В то время как продолжительность выполнения требуемых запросом пользователя операций на временном наборе данных может быть не ограничена по времени, выполнение актуализации результатов операции должно быть произведено

максимально быстро. При возникновении нештатных ситуаций (исключений) на любом этапе описанного выше процесса должен быть произведен откат на состояние до начала исполнения запроса (начала транзакции).

Сервером не использовались операции физического удаления файлов, вместо этого выполнялась их утилизация в специально отведенные для этого директории, с присвоением удаляемым файлам уникального имени. Задачи физического удаления файлов должны выполняться при проведении регламентных работ на файловом сервере.

С целью реализации изложенных выше концепций требовалось использовать в каждой локации две служебные директории: tmp – для хранения временных файлов, и recycler – для хранения удаленных файлов (рис. 10).

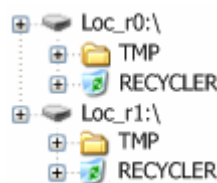


Рис. 10. Служебные директории файлового сервера

Следующие операции с файлами требовали модификации данных на стороне сервера, а следовательно, и выполнения их в виде отдельных транзакций: 1) загрузка нового файла; 2) загрузка файла взамен существующего; 3) переименование файла; 4) переименование файла с заменой существующего файла; 5) удаление файла.

Загрузка нового файла осуществляется во временную директорию локации для записи под временным

именем. После загрузки файл переименовывается и перемещается на требуемое место в той же локации. Клиенту отправляется уведомление о завершении транзакции (рис. 11).

Загрузка нового файла взамен существующего производится следующим образом: новый файл загружается во временную директорию локации для записи под временным именем, существующий файл перемещается в директорию для утилизации внутри содержащей его локации для чтения, при этом ему присваивается имя для утилизации. Загруженный файл переименовывается и перемещается на требуемое место в локации для записи. Клиенту отправляется уведомление о завершении транзакции (рис. 12).

Переименование файла выполняется в следующем порядке: файл переименовывается и перемещается в содержащей его локации для чтения, после чего клиенту отправляется уведомление о завершении транзакции (рис. 13).

Переименование файла с заменой существующего файла производится следующим образом: заменяемый файл перемещается в директорию для утилизации внутри содержащей его локации, при этом ему присваивается имя для утилизации. Переименовываемый файл перемещается внутри содержащей его локации. Клиенту отправляется уведомление о завершении транзакции (рис. 14).

Удаление файла выполняется следующим образом: файл перемещается в директорию для утилизации содержащей его локации, при этом ему присваивается имя для утилизации. Клиенту отправляется уведомление о завершении транзакции (рис. 15).

Следует отметить, что в материале, изложенном выше, под именем файла понимается имя файла, включая расширение и полный путь к нему в терминах локаций.

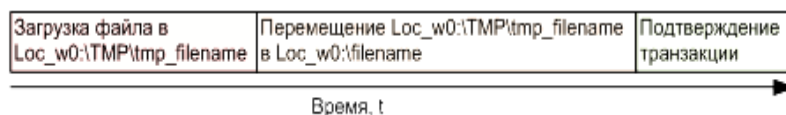


Рис. 11. Загрузка нового файла

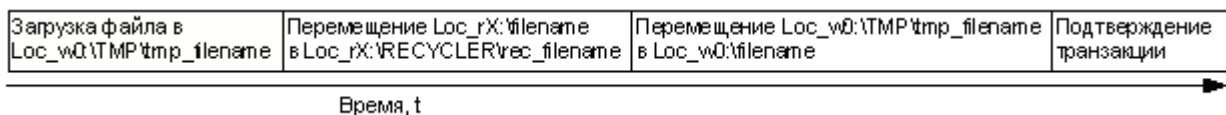


Рис. 12. Загрузка нового файла взамен существующего

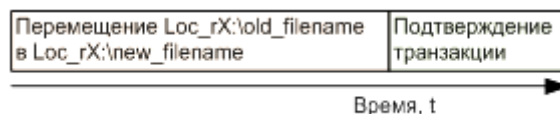


Рис. 13. Переименование файла

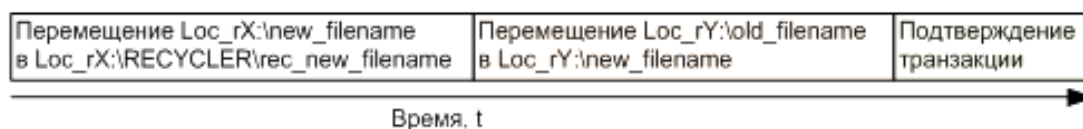


Рис. 14. Переименование файла с заменой существующего файла

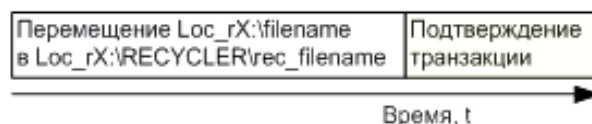


Рис. 15. Удаление файла

Принятая схема работы предусматривает загрузку новых файлов только в локацию для записи, в локациях для чтения производятся только операции переименования и перемещения файлов, что гарантирует неизменность размера хранимых в них данных. Следует также учесть, что выше приведены лишь общие схемы обработки операций, в то время как практическая реализация описанных процессов значительно сложнее.

Изложенные принципы реализованы в файловом сервере ИПС СТО, что позволило использовать систему ИПС СТО для организации электронного архива технической документации, а также реализовать электронный документооборот технологической документации.

Создание электронного архива технической документации позволило ликвидировать затраты на микрофильмирование документации, так как документация в электронном виде не подлежит микрофильмированию; обеспечить полную сохранность и качество документации в течение всего срока хранения; сократить сроки разработки технологической документации за счет непосредственного доступа к ней с любого рабочего места, оснащенного ПК, и ее унификации за счет использования ранее разработанной документации в качестве аналога; интегрировать электронный архив технологической документации в систему электронного документооборота в рамках внедрения на предприятии CALS-технологий; переквалифицировать две штатные единицы архивариусов в операторов ЭВМ (без приема дополнительных единиц).

Кроме того, перевод в электронный вид задела технологической документации обеспечил высвобождение 50 м² производственных площадей.

Разработанные в процессе создания сервера ИПС СТО универсальные классы могут быть использованы для решения задач обмена данными по протоколу TCP/IP, которые актуальны для обмена данными не только между удаленными системами, но и между приложениями, аппаратно исполняемыми на одной и той же системе, как, например, в приложениях Oracle для

Win 32/64; для решения задач создания серверов и файловых серверов, работающих по протоколу TCP/IP, с требуемой функциональностью.

Библиографические ссылки

1. Поляев К. Н., Михнев М. М., Злотенко В. В. Файл-сервер : программа для ЭВМ. Зарегистр. в Федер. службе по интелект. собственности, патентам и товарным знакам. Свидетельство об офиц. регистрации № 2005611979. М., 2005.
2. Поляев К. Н., Михнев М. М., Злотенко В. В. Библиотека : программа для ЭВМ. Зарегистр. в Федер. службе по интелект. собственности, патентам и товарным знакам. Свидетельство об офиц. регистрации № 5005611977. М., 2005.
3. Поляев К. Н., Злотенко В. В., Туркени Р. П. Регистратор извещений : программа для ЭВМ. Зарегистр. в Федер. службе по интелект. собственности, патентам и товарным знакам. Свидетельство об офиц. регистрации № 5005611978. М., 2005.
4. Поляев К. Н., Шевердов В. Ф., Злотенко В. В. Менеджер СТО : программа для ЭВМ. Зарегистр. в Федер. службе по интелект. собственности, патентам и товарным знакам. Свидетельство об офиц. регистрации № 2005612596. М., 2005.
5. Михнев М. М., Поляев К. Н., Злотенко В. В. Загрузчик : программа для ЭВМ. Зарегистр. в Федер. службе по интелект. собственности, патентам и товарным знакам. Свидетельство об офиц. регистрации № 2006611228. М., 2006.
6. Злотенко В. В., Поляев К. Н., Чупилко В. Д. Регистратор техпроцессов : программа для ЭВМ. Зарегистр. в Федер. службе по интелект. собственности, патентам и товарным знакам. Свидетельство об офиц. регистрации № 2006611285. М., 2006.
7. Бартенев В. А., Смирных В. Н., Злотенко В. В., Поляев К. Н. Zx Loader v.10 : программа для ЭВМ. Зарегистр. в Федер. службе по интелект. собственности, патентам и товарным знакам. Свидетельство об офиц. регистрации № 2007610319. М., 2006.

DEVELOPMENT OF FILE SERVERS FOR AUTOMATION SYSTEMS OF TECHNOLOGICAL PREPARATION OF PRODUCTION

The authors consider problems of development of file servers for automation systems of technological preproduction, for example, the technical solutions implemented in the file server of information retrieval system of technological support.

Keywords: technological preparation, information system, a file server.

© Михнев М. М., Поляев К. Н., 2012

УДК 681.142.2

П. Б. Могнонов, Д. Н. Чимитов, Н. В. Ярышкина

РЕАЛИЗАЦИЯ ИНТЕРПРЕТАТОРА С ЗАДЕРЖАННЫМ ВЫЧИСЛЕНИЕМ

Рассматривается один из способов оптимизации работы алгоритмов посредством интерпретатора с задержанным вычислением. Делается попытка реализовать алгоритм работы такого интерпретатора на базе расширенного функционального языка ЛИСП. Показывается значение задержанных вычислений для адаптации процедуры.

Ключевые слова: задержанные вычисления, интерпретатор, лямбда-исчисление, ЛИСП, полиморфизм.

Под интерпретатором с задержанным вычислением понимается такой интерпретатор, который вычисляет значения аргументов по необходимости, т. е. в тот момент работы программы, когда действительно необходимо значение вычисляемого выражения. Таким образом, в ходе своей работы при вызове какой-либо процедуры интерпретатор с задержанным вычислением определяет, какие аргументы требуется вычислить, а какие задержать. Задержанные аргументы при этом не вычисляются, а преобразуются в объекты, называемые рецептами [1].

В данной статье сделана попытка показать, как можно использовать интерпретатор с задержанным вычислением U для автоматического решения следующей задачи: пусть в банке процедур B имеется относительно обобщённая (или полиморфная) процедура $P(x_1, \dots, x_n)$. Тогда интерпретатор U , вызывая $P(x_1, \dots, x_n)$ и имея контекстные условия C на текущий момент t процесса вычисления, может задержать немедленное исполнение процедуры $P(x_1, \dots, x_n)$ на время, необходимое для конкретизации процедуры $P(x_1, \dots, x_n)$, или, другими словами, для адаптации процедуры $P(x_1, \dots, x_n)$ к текущему контексту C вычислительного процесса. Во время адаптации (т. е. во время задержки и интерпретации) процедуры P интерпретатор U определяет фактические значения некоторых параметров $\{x_{i1}, x_{i2}, \dots, x_{im}\}$, где $(m \leq n)$ исходя из контекста C . Процедура P с вычисленными фактическими параметрами определяет производную процедуру P_1 . Можно сказать, что процедура P является родителем процедуры P_1 , а P_1 – потомком P . Отметим, что в банке процедур B все процедуры частич-

но упорядочены отношением полиморфизма и образуют полную решетку (или декартово-замкнутую категорию Скотта [2; 3]), что отражает структуру знаний, записанную в банке процедур B .

Разработка структуры знаний банка процедур B в данной работе не затрагивается и предполагается, что эта задача инженерии знаний конкретной предметной области. Таким образом, имеем следующую схему работы интерпретатора U : **исходные данные**: [1] $P(x_1, \dots, x_n)$; 2) контекст динамического процесса вычисления C ; 3) задержка, необходимая интерпретатору U : $C \rightarrow \{x_{i1}, \dots, x_{im}\}$ для определения фактических параметров процедуры P] **результат**: конкретизированная (или адаптированная) процедура P_1 .

Рассмотрим варианты реализации алгоритма работы интерпретатора с задержанным вычислением, разработанного на базе расширенного функционального языка ЛИСП.

Реализация интерпретатора с задержанным вычислением для количественной оптимизации процесса вычислений. В качестве примера рассмотрим программу, выполняющую функцию формирования суммы и произведения бесконечного списка чисел:

```
сумпроизвспис(n) ≡ сумпроизв(список(1, n))
сумпроизв(x) ≡ {сп (x, 0, 1)
    где сп = λ(x, s, p)
    если равно(x, НИЛ) то двучлен(s, p)
    иначе сп (cdr(x), s+car(x), p*car(x))}
список(m, n) ≡ если m>n то НИЛ иначе
    cons(m, список(m+1, n))
двучлен(x, y) = cons(x, cons(y, НИЛ))
```