Информационная система для разработки технологий организации сложной совместной деятельности♣

© Сергей М. Абрамов, Сергей В. Знаменский,
Институт программных систем имени А. К. Айламазяна РАН
abram@botik.ru svz@latex.pereslavl.ru

© Надежда С. Живчикова, Андрей В. Котомин, Елена В. Титова
УГП имени А. К. Айламазяна
ming@pereslavl.ru klein@pereslavl.ru andia@andia.pereslavl.ru

Аннотация

Такие сложные виды совместной творческой деятельности, как формирование программы научной конференции, экспертиза научных проектов и отчетов, разработка сложного программного обеспечения и управление образовательной активностью, могут качественно выиграть от добротной информационной поддержки.

Описан подход к созданию программной среды для поиска и разработки лучших технологий информационной поддержки организации сложной совместной творческой деятельности.

1 Идея и ее истоки

Сложной совместной деятельностью мы будем называть многопользовательскую активность по поддержанию и развитию информационного контента со следующими особенностями:

1. Структура контента и организация многопользовательского доступа могут потребовать непредсказуемых изменений.
2. Неожиданно может потребоваться доступ к прежним состояниям любой части контента.

Речь здесь, разумеется, идет не о немыслимом интеллекте информационной системы, сверхгибко подстраивающейся под потребности пользователей, а о включенности в эту активность группы разработчиков системы. Острая потребность в такой системе ощущается в некоторых сферах управления социальными процессами, например, при разработке инновационных технологий организации обучения. При этом полезны такие тесно взаимосвязанные инструменты совместной деятельности, как индикация изменившихся данных и важных дел, требующих участия пользователя, интерфейсы совместного доступа к данным и совместной выработки решений, алгоритмы персонализации информации и автоматического учета активности, продуктивности и дисциплинированности каждого участника. Эти инструменты очевидно не могут создаваться вне системы или независимо друг от друга и способны дать значимый эффект лишь в комплексе.

Задача создания такой системы, по-видимому, ранее не рассматривалась по причине её сложности. В случае успешного решения результат мог бы оказаться полезным и в других сферах, поскольку современные стандарты управления качеством ISO 9001:2008, ГОСТ Р ИСО 15489 ориентируют на:

- надежное сохранение истории и авторства информации,
- удобный эффективно авторизованный доступ к информации,
- возможность перестройки бизнес-процессов,

то есть, по сути, на сложную совместную деятельность в нашем понимании.

Важнейшая из сфер деятельности, где такая информационная система также обещает мощный эффект,— это разработка сложного программного обеспечения.

Информационная система, обещающая повышение эффективности творческого сотрудничества с участием разработчиков, обязана стать идеальной интегрированной средой для ее (системы) дальнейшей разработки и сопровождения.

.

1.1 Сохранение истории и авторства

Первые попытки организации полной истории разработки были связаны с задачами управления версиями программ и решались многочисленными системами управления ревизиями исходных кодов.

Эти попытки быстро распространились на сопроводительную документацию и на другие документы. Возможность анализа изменений и возврата к прежней версии стала основой функциональности многочисленных wiki-сайтов, включая Википедию.

Труды 11[й] Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» - RCDL'2009, Петрозаводск, Россия, 2009.

Системы отслеживания изменений стали закладываться и в основу корпоративных систем для обеспечения сохранности информации и безопасности. Например, первая в России биллинговая система с открытыми исходными кодами Nadmin [4] автоматически сохраняет в файлах под управлением RCS вместе с измененными данными дату и время изменений и аутентифицирующие пользователя данные.

1.2 Организация доступа к документам

В информационной системе с добротным контролем качества управление версиями должно интегрироваться с эффективными механизмами организации доступа. В wiki-системах проработаны механизмы совместного создания и редактирования документов, но мало поддержаны средства организации экспертизы и утверждения согласованной информации. Часто подобные механизмы надстраиваются над wiki-системами, оказываясь недостаточно удобными и малоэффективными.

Пятилетний опыт интенсивной эксплуатации системы Twiki (FOSwiki) [10,13], все эти годы остававшейся в списке лидеров wiki-систем, выявил полную непригодность заложенной модели структурирования информации и разграничения доступа к ней для построения на ее основе качественной информационной системы.

Главной проблемой стало оповещение эксперта, руководителя (и вообще пользователя) о появлении ожидающих срочного рассмотрения или утверждения документов или поправок.

Здесь возникли неожиданные сложности. Если право поставить визу имеет любой из группы пользователей, то все они должны видеть необходимость этого действия до тех пор, пока кто-то из них не сделает дело. Список текущих дел пользователя должен динамически отражать изменения его статуса и состояния дел в системе.

Оповещение о деле не должно отвлекать от работ по сути совершенно не связанных с выполняемой. Его видимость должна учитывать близость к текущей деятельности. В идеале у пользователя должны быть возможности контекстно влиять на видимость оповещений.

Задача организации динамического оповещения пользователя об изменениях в данных, учитывающего возможные изменения текущей организации доступа, настроек и текущей активности пользователя, оказалась трудной.

Проблема в том, что обычное сохранение данных одним пользователем может оказаться настолько важным для некоторых других пользователей, что их полезно известить об этом, не дожидаясь, пока они закончат чтение сложной страницы или заполнение большого текстового поля, родственного к этому измененному данному. Другая ситуация — практически одновременное изменение общего данного двумя пользователями в различное состояние. Если не дать знать о возникшем конфликте, то может потеряться ценная информация. Сообщения о системных неполадках и жалобы пользователей полезно оперативно и гарантированно доводить до соответствующих администраторов и разработчиков системы, причем любой получивший может выключить тревожный сигнал.

Стандартный подход передачи сообщений через рассылку неприемлем по двум причинам:

1. Если опасность уже миновала, то внимание к ней привлекать излишне.
2. Если у пользователя изменилась роль, то он должен сразу же увидеть ситуацию по-новому.

Качественный анализ этих и других возможных ситуаций при каждом обращении клиента за возможными уведомлениями требует или доступа к истории сохранения, или несоразмерных серверных ресурсов для своевременного анализа состояния данных.

Стало ясно, что какая-то часть работы должна происходить при каждом сохранении информации и учитывать текущую активность всех пользователей в системе.

Нам не удалось найти в литературе и в сети Internet ни разумных подходов к решению этой проблемы, ни ссылок на программные средства, пригодные для ее приемлемого решения.

1.3 Перестройки структур и процессов

Обостряющаяся проблема потерь доступа к старой информации при существенном обновлении информационных систем хорошо известна. Если при замене информационной системы заказчик поддается соблазну переноса накопленных данных в новую систему, то он почти наверняка теряет какую-то их часть и какие-то из старых интерфейсов к этим данным. Если не поддается, то он теряет больше — возможность совместной обработки старых и новых данных.

Описанные издержки становятся серьезной проблемой в ситуации, когда продолжительность актуальности данных многократно превышает среднюю продолжительность стабильности структуры организации доступа к хранимой информации. Такая ситуация особенно характерна для систем поддержки научно-образовательной деятельности.

Известные описания «эволюционирующих» или «устойчиво развивающихся» информационных систем, адресованные этой проблеме (см., например, [9–11]) указывают частные решения проблемы, поскольку в них будущее развитие проистекает в жестко фиксированных рамках.

2 Поиск архитектуры системы

Рассматриваемая модель построения информационных систем крайне существенно отличается от доминирующей MVC [12] и подавляющего большинства других «рациональных» моделей информационных систем: в ней организация хранения и обработки данных принципиально не может определяться фиксированной бизнес-логикой, извлечен-

ной из предпроектного обследовании. Архитектура системы должна соответствовать логике эффективного обеспечения взаимодействия сотрудников, информирования их о важных изменениях, а не тем, на какие изменения в данных направлена их совместная деятельность.

Остается один ориентир: возможность создания эффективного и дружественного пользовательского интерфейса. Рассмотрим те полезные качества, которые отсутствуют в большинстве существующих систем.

2.1 Контекстная автономность

Основным содержанием обсуждаемой системы является слабо структурированная информация (данные и исполняемый код). При разумной организации такой информации каждый элемент идентифицируется строкой, начало которой несет семантическую нагрузку. Мы будем называть контекстом совокупность данных с фиксированным началом строки идентификатора.

Примерами контекстов являются содержимое директории в файловой системе или на статическом сайте и множество слов, начинающиеся на 'множ' в словаре.

Назовем информационную структуру контекстно-автономной, если организация доступа к данным любого контекста (и, разумеется, сами данные) могут быть изменены независимо от остальной части системы без приостановки сервисов, в которых контекст не участвует.

Контекстная автономность характеризует гибкость структуры системы. По сути она обещает, что любую разумно выделяемую часть системы можно перестроить с ее бизнес-логики на любую другую без приостановки сервиса.

Отметим, что произвольно перестраиваемая иерархическая модель данных рассмотрена в [2].

2.2 Сохранение истории и доступ к ней

Поскольку перестройки структуры легко приводят к потере доступа к данным, контекстная автономность должна дополняться специфическими механизмами безопасности. В системе, отслеживающей изменения в информации, естественно организуется доступ на чтение ранее доступной информации.

В системе хранятся все версии данных и исполняемого кода (и рабочие ссылки на них) и при обработке запросов с указанием времени запроса вызывается действительная на момент запроса версия. Это гарантирует полный доступ на чтение к прежним состояниям системы. Разумеется, доступ должен быть авторизован в настоящем времени и не может фальсифицировать историю.

Вот как это может выглядеть со стороны пользователя:

Изображение странички с заданным url на экране, которое каждые пол-секунды уточняется (перерисовываются изменившиеся фрагменты).

Дополнительная верхняя строчка содержит панель управления «машиной времени»: табло с временем отображаемой странички, справа от табло кнопки ускорения и замедления прокрутки в широком диапазоне, а слева две кнопки возврата: на 5 секунд и на минуту назад. На кнопках возврата отображается время, которое появится на табло при нажатии на кнопку.

Такой просмотр (ускоренный, замедленный или в режиме реального времени) истории информационно наполненной страницы с цветными диаграммами и выделениями проблемных участков даст беспрецедентно наглядное и точное представление о динамике происходивших в данных изменений.

Добротная реализация отслеживания изменений в основе информационной системы позволит сделать возможными отмену действий пользователя (подобную undo в офисных приложениях) и быструю ликвидацию системным администратором ближайших последствий несанкционированного доступа или перехода на недоработанную версию исполняемого кода.

2.3 От блокировок к поточной обработке

Пользователю удобно иметь быстрый доступ к корректировке сопутствующей информации. Но чем больше доступных для изменения данных содержит страница web-интерфейса, тем больше вероятность потери никем не замеченных изменений, сохраненных одним пользователем во время работы другого над другими данными этой страницы.

Стандартный подход к исключению потерь информации это использование транзакций и блокировок. Если, например, два пользователя одновременно пытаются перевести по рублю один со счета А на счет Б, а другой со счета Б на счет А, то стандартный механизм подразумевает сохранение в логах запросов от каждого пользователя и блокировку обоих счетов при выполнении каждого из переводов. Если два разных процесса-обработчика одновременно сначала блокируют счет с которого, а потом счет, на который переводятся деньги, то получается вечная блокировка deadlock, наличие которой снижает производительность системы. Чем сложнее обработка, тем труднее своевременно выявлять и распутывать такие ситуации в автоматическом режиме. При сложных обработках для восстановления системы порой неизбежен перезапуск сервиса с потерей последних изменений.

Вместо тормозящих блокировок и слепого накладывания одних изменений на другие, используемых в целях разрешения конфликта, предлагается использовать немедленную передачу каждого законченного изменения маленького фрагмента информации на сервер с оперативной индикацией изменений (или, по меньшей мере, их наличия) у каждого из пользователей, видящих форму для изменения этих данных. Теперь мы видим, какая часть информации относительно стабильна, а какая только что изменилась другими, где другой пользователь

правит данные, а где его правки наложились на наши.

В случае, когда (как в примере с переводом денег) для конфликта нет основы, в совместных блокировках различных данных (а именно они порождают проблему) никакой нужды нет. Пользователи могут одновременно взять по рублю с одних счетов и положить на другие. При этом никто не пострадает, если гарантируется, что взятый рубль будет положен и что в момент, когда рубли взяты, но еще не положены, не произойдет вычисление суммы средств на счетах или другой операции, требующей согласованности этих данных.

Для неблокирующего разрешения обоснованных конфликтов полезны доступность история изменений каждого фрагмента информации, контекстный чат либо форум для обсуждения спорных ситуаций и разбиение большого документа на независимо изменяемые фрагменты.

Мелкая на первый взгляд особенность взаимодействия в корне меняет базовые требования к организации хранения, обработки и визуализации данных. Данные счетов в базе как и их отражение на экране браузера больше не должны отражать каждый момент консистентное состояние данных.

На самом деле стандартные строгие требования к безупречности логических взаимосвязей данных в любой момент времени ACID (см. [7]) не столь непреложны для динамически меняющейся информации. Пользователю вполне достаточно, чтобы в каждый момент времени каждая частица экрана отражала состояние, в котором соответствующие данные находились в очень близкий момент времени. Разница незаметна, если близость составляет десятую долю секунды и не раздражает даже составляя секунды, если она естественно отражает перегруженность сервера или сложность вычислений.

Аналогично и функция, обрабатывающая данные информационной системы (например, вычисляющая сумму средств на счету), просто не должна выполняться, пока все входные значения не окажутся в стабильном состоянии. Если система может гарантировать быструю актуализацию входных данных, то о последнем можно судить даже по давности их изменений. Но она в любом случае может фиксировать информацию о транзакциях (как в обычных логах) и вместо блокировки данных откладывать выполнение функции. При перегрузках сервиса это грозит тем, что сводные данные на экране пользователя будут обновляться реже, чем первичные, но такая потеря качества обслуживания намного лучше, чем перебои в сервисе.

Избежать хаотических дёрганий страницы web-интерфейса из-за незамедлительного показа изменений во время работы нетрудно. Для этого могут использоваться фрагменты фиксированного размера и иконки, сигнализирующие о наличии не просмотренных изменений.

2.4 Динамическая навигация

Будучи одновременно включен в сотни совместных дел, пользователь должен видеть, какие из них предполагают или ожидают его немедленного участия. Если, например, в одном из них изменено сохраненное им значение, то для него может быть важно не откладывая разобраться с конфликтной ситуацией.

Подобных ситуаций может быть много и они могут в разной степени волновать пользователя и настаивать на его участии.

В любом состоянии прокрутки на экране должна быть видна иконка, отражающая важность не просмотренных пользователем изменений в его делах с учетом его персональных настроек. Эта иконка должна открывать контекстное меню, оптимизированное под текущую ситуацию, в котором были бы отражены наиболее важные для пользователя дела, наиболее близкие к текущему контексту.

2.5 Выбор средств реализации

Попытки частичного сочетания описанной функциональности в системе поддержки проведения научных конференций отчетливо выявила проблему производительности: выявление обновлений системы и доведение их до пользователей вылилось в обработку больших объемов рассредоточенных данных. Обоснованные сомнения в возможности получения нужной производительности при использовании SQL-сервера из-за неизбежных блокировок привели к в выводу о целесообразности опоры на Berkeley DB, обещающей в десятки раз более высокую производительность [8]. Это однозначно определило и выбор языка реализации: Perl оказался по сути единственным скриптовым языком для высокопродуктивных (под Apache ModPerl2) веб-приложений с полноценной поддержкой Berkeley DB.

Однозначность выбора средств реализации привела к архитектуре, ориентированной на выбранные средства.

3 Архитектура и особенности реализации

Поскольку принципы и средства такой информационной поддержки должны уточняться и прорабатываться в ходе эксплуатации системы, то она базируется на идеях эволюционного прототипирования [9] и должна стать удобной интегрированной средой для саморазработки.

Клиентская часть системы представляет собой обычный современный браузер с включенной поддержкой javascript (при работе без javascript рекомендуются периодические перезагрузки страницы чтобы не упустить важную информацию).

Клиентские скрипты

- регулярно (несколько раз в секунду при измененных данных, раз в несколько секунд при длительном бездействии пользователя) пересылают на сервер асинхронный запрос,

содержащий идентификаторы фрагментов страницы и изменения в данных, не фиксированные в базе серверных данных;

- получив в ответе обновления, показывают их на странице без ее полной перезагрузки;
- получив хеш последних из зафиксированных на сервере в течение последней минуты значений записанных пользователем данных, сверяют их с посылаемыми серверу и удаляют принятые сервером из списка отправляемых.

3.1 Компоненты серверной части

Загрузчик анализирует запрос и запускает код основной процедуры обработки конкретного запроса.

Задачей загрузчика является разбор параметров запроса к серверу и обращение к базе авторизации, которая должна вернуть имя той версии программного модуля из каталога, которую загрузчик должен вызвать.

Простота и неизменность кода загрузчика обеспечивают не только ретроспективную обработку с кодом и данными, актуальными на указанный момент времени, но и рабочее тестирование в той же системе следующей версии кода. Это облегчает и делает безопасными частые обновления системы и позволяет параллельно прорабатывать несколько веток кода.

Репозиторий приложенных файлов хранит всю бинарную информацию. Это статическая директория, файлы которой доступны в сети Internet по именам, а листинг недоступен. Имена файлов содержат дайджест содержимого, что практически исключает возможность их угадывания неавторизованным пользователем.

Хранилище текстовой и структурообразующей информации хранит текущее состояние и всю историю системы. Оно эффективно обрабатывает ретроспективные запросы с учетом пользовательских undo и учитывает записанные при тестировании данные только в тестовых запросах.

3.2 Базовая структура данных

Все данные, включая шаблоны и имена файлов исполняемого кода, логически хранится в базе типа BTree. Это означает, что записи в базе состоят из ключа и значения некоторого данного. Ключи записей логически образуют лексикографически упорядоченный список с двоичным деревом доступа к нему, позволяющим найти по строке первую запись, лексикографически не превосходящую строку запроса.

Ключ записи является конкатенацией строк, среди которых обычно есть

- префикс, идентифицирующий данное,
- строка обратного отсчета времени от конца XXI века.

Это позволяет очень быстро, за считанные (логарифмически растущие с числом записей в базе) микросекунды получить актуальное на любой заданный момент времени значение любого данного.

Значение данного является строкой, в которой может быть сериализована любая структура или код в Perl, ключ другой записи или имя файла с данными.

Значением может быть и директива пользовательского undo — отмены последних изменений. Директива делает неактуальными все сделанные пользователем изменения за период от указанного в ней момента времени до момента выдачи директивы и все их последствия. Среди таких изменений может быть директива undo, то есть неудачные изменения можно вернуть «по горячим следам».

Ключ записи может содержать тестовую пометку. При тестировании в системе учитываются все данные, и все изменения помечаются как тестовые и не отражаются на работе обычного пользователя, поскольку при обработке обычных запросов тестовые данные игнорируются. Это напоминает слои изображения в графическом редакторе тем, что слой тестовых данных как бы находится перед обычными слоями информации, но невидим для обычных пользователей.

Идентифицирующий данные префикс обычно является составным и формируется так, чтобы максимально использовать быстрый механизм двоичного дерева для проведения наследования данных при авторизации доступа к страничкам и выборе нужной версии шаблона.

3.3 Хранение и резервирование данных

Хранение данных осуществляется в базах типа BTree.

Архивная база A содержит все данные, сохраненные в системе до определенного момента. Большую часть времени она открыта только на чтение и представляет собой один большой файл, который после очередного пополнения данными архивируется стандартными утилитами работы с файлами.

Проверить сохранность старых данных может самостоятельный фоновый процесс, сверяющий сохраненные в последней и в старой резервной копии данные. Если не произошло порчи данных, то после создания очередной резервной копии можно спокойно удалить все предыдущие: все ее данные в неизменном виде содержатся в новой резервной копии. Таким образом, если учитывать место, требуемое для хранения резервный копий и логов в других информационных системах, сохранение полной истории изменений может оказаться даже более экономным в плане требований к размерам дискового пространства.

В любом случае хранение неактуальных версий файлов и другой информации безусловно связано с затратой ресурсов и естественно порождает проблему освобождения от лишней информации. Эта проблема автоматически решается выделением «белых пятен истории» (промежутков времени, история в которых недоступна) и удалением информации,

актуальной лишь в выделенных временных промежутках.

Для этого весь период времени с начала существования системы до суток назад равномерно по мере dt/t^2 разбивается например на 10000 промежутков и выделяется в качестве белых пятен десяток содержащих наибольшее количество записей. После этого уточняются границы белых пятен и фиксируются в особой записи базы. Все это может делать фоновый процесс, не взаимодействующий с сервером. Учитывая, что подавляющее большинство изменений вносится в периоды пиковой загрузки, пятна общей меры не превышающей один процент могут спрятать более 90% неактуальной информации.

Если в фоновом режиме составить список ключей базы, подлежащих удалению, то можно нацелить серверные процессы на чистку базы в периоды минимальной загруженности.

Если сверка в фоновом режиме содержимого актуальной и ранее сохраненной резервной копии показывает идентичность значений ключей, не покрытых белыми пятнами истории, то хранение старых резервных копий теряет смысл. Это облегчает задачу хранения резервных копий.

Буферная база B содержит данные, полученные в течение последней доли секунды или нескольких секунд и модифицируется всеми серверными процессами, обслуживающими запросы пользователей.

Перед чтением данных, переносимых в базу текущей информации C, буферная база блокируется и проверяется на управляемость. При сбое эта база не восстанавливается, а просто пересоздается, что не приводит к потерям ценной информации, поскольку скрипты браузера повторяют отправку на сервер измененных данных до получения подтверждения.

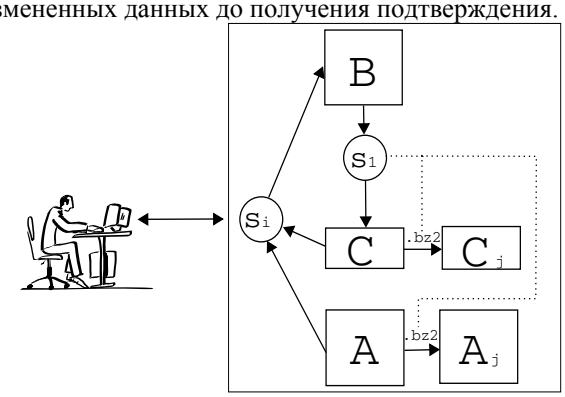

Рис. 1. Основные потоки информации

База текущей информации C содержит долю процента сохраненной в системе информации, что обеспечивает ее быстрое восстановление из резервной копии. Вероятность, что такое восстановление потребуется, сведена к минимуму тем, что модифицирует ее только один процесс и тот ничего не удаляет, а только чередует перенос подготовленных данных из буферной базы с резервным копированием базы. Резервные копии базы текущей информации реализуют редкое для баз данных инкрементное резервное копирование.

В случае гибели процесса в момент записи восстановление такой базы происходит в доли секунды и обычно даже не требует резервной копии. При необходимости просто организуется и репликация.

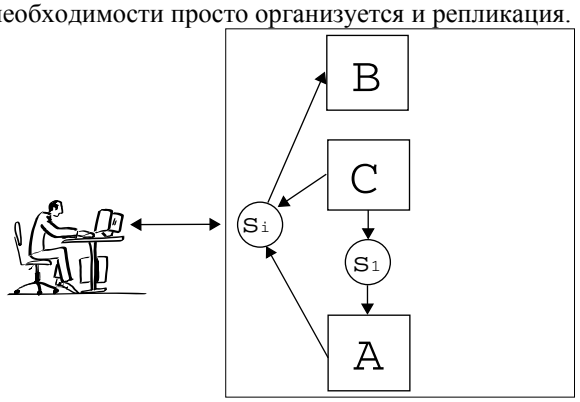

Рис. 2. Пополнение архивной базы.

3.4 Рабочий цикл серверного процесса

При получении запроса серверный процесс аутентифицирует пользователя и по командной строке запроса определяет, относится ли запрос к рабочей системе или к тестированию нового кода в ней. Если идет тестирование, то проверяется наличие в настройках пользователя указание на особую версию системы и подгружается либо активизируется соответствующая версия.

Далее проверяется вариант пользовательского доступа к запрошенной странице, считывается показатель загруженность сервера и показатель загруженности сервера запросами данного пользователя и выбирается вариант обслуживания. Из данных, сохраненных в базах C и A, формируется ответ (асинхронный запрос может остаться и без ответа).

Чтобы ответ на любой допустимый запрос отнял меньше ресурсов, при ответе никаких вычислений не происходит, только в сохраненные в базах шаблоны подставляются данные из этих же баз. Данные запроса сохраняются в буферной базе и ответ отправляется.

Если сервер достаточно свободен и пользователь не узурпировал слишком много серверных ресурсов, то кроме выдачи минимально возможного ответа процесс проводит очередную часть работы по уточнению пользовательского профайла и поиску интересной для пользователя информации и фиксирует в базе результаты поиска. Поскольку эта работа требует в десятки раз больше ресурсов, чем обычное обслуживание, то ее исключение амортизирует пики нагрузки, позволяя избежать привычно резкого падения производительности сервисов и катастрофического снижения надежности баз данных при пиковых перегрузках.

4 Заключение

Описанная система http://edu.botik.ru разрабатывается в научно-учебном комплексе ИПС имени А. К. Айламазяна РАН и УГП имени А. К. Айламазяна в г. Переславле-Залесском. С ее использованием с 2007 года систематически прово-

дятся конференции с перекрёстным рецензированием [1,5,6] и с 2008 года ведется трекинг студенческих проектов. Система ежегодно перерабатывается с сохранением функциональности. В 2009 году система перестраивается на описанных принципах с целью достижения нового качества организации учебного процесса.

Литература

[1] Амелькин С. А., Знаменский С. В. Информационная поддержка организации сложной совместной деятельности. // Труды международной конференции “Программные системы: теория и приложения”, ИПС имени А. К. Айламазяна РАН, г. Переславль-Залесский, май 2009. Переславль-Залесский: Изд-во “Университет города Переславля”, Т.1, 2009. — с. 123–132

[2] Живчикова Н. С., Титова Е. В. Логическая модель изменчивых организационных структур. // Тезисы международной конференции "Системы проектирования, технологической подготовки производства и управления этапами жизненного цикла промышленного продукта (CAD/CAM/PDM - 2008)". М: Институт проблем управления РАН, 2008 - с.77-81.

[3] Знаменский С. В. Хорошо масштабируемое автономное администрирование доступа. // Международная конференция "Программные системы: теория и приложения", Переславль-Залесский, октябрь 2006, Наука,-Физматлит, М. Т. 1, 2006. - с. 155-169.

[4] Карлаш А. В., Абрамов С. М., Жбанов П. Г., Нестеров А. С., Ермилова Е. В., Шевчук Ю. В. Надмин - административно-расчетная система для региональных сетей. // Труды Всероссийской научной конференции " Научный сервис в сети Интернет ", 20-25 сентября 2004 г. Новороссийск, Изд-во МГУ, 2004. - М., с. 195 - 200. http://skif.pereslavl.ru/psi-info/rcmsregional.nets/regional.nets-publication/2004-rus/08-04.pdf.

[5] Коряка Ф. А.. Автоматизированная система управления вузом - UPIS. // XI научно-практическая конференция "Университета г. Переславля". Переславль-Залесский, апрель 2007, изд.-во "Университет города Переславля", Т.1, 2007. - с. 59-63. - эл. ресурс: http://wiki.botik.ru/up/pub/IS4UGP/StudConf/1-2/03-koryaka-p-59.pdf.

[6] Степанов Д. Н. Алгоритм назначения рецензентов как часть проведения научных конференций при поддержке информационной системы UPIS. // XII научно-практическая конференция Университета г. Переславля. Переславль-Залесский, апрель 2008, изд.-во "Университет города Переславля", Переславль-Залесский. Т. 1, 2008. - с. 155-169.

[7] ACID - // Материал из Википедии - свободной энциклопедии, электронный ресурс, http://ru.wikipedia.org/wiki/ACID.

[8] Berkeley DB // Материал из Википедии - свободной энциклопедии, эл. ресурс: http://ru.wikipedia.org/wiki/Berkeley_DB.

[9] Andersson Pontus; Birath David; .Serenity Patrik Willard: A Case Study in Developing Sustainable Information Systems. University essay from IT-universitetet I Gцteborg/Tilldmpad information-steknologi. Information Systems Development: Advances in Theory, Practice and Education. Edited by O. Vasilecas et al., Springer, 2005, эл. ресурс: http://www.essays.se/essay/17da5786ee/.

[10] Foswiki - The free and open source enterprise wiki. 2008-2009, эл. ресурс: http://www.foswiki.org/.

[11] Mart Roost, Karin Rava, Tarmo Veskioja, Supporting self-development in service oriented information systems, // Proceedings of the 7th Conference on 7th WSEAS International onference on Applied Informatics and Communications, p.52 - 57, August 24 - 26, 2007, Vouliagmeni, Athens, Greece.

[12] Model-View-Controller (MVC) // Материал из Википедии - свободной энциклопедии, эл. ресурс: http://ru.wikipedia.org/wiki/MVC.

[13] TWiki - the Open Source Enterprise Wiki and Web 2.0 Application Platform by Peter Toeny and contributing authors, 1998-2009, эл. ресурс: http://www.twiki.org.

Information System for Complex Collaboration Technologies Development

S.M. Abramov, S.V. Znamenskij, N.S. Zhivchikova, A.N. Kotomin, E.V. Titova

Such creative collaboraton forms as scientific conference program development, project expertise, learning management and complex software development may benefit greatly from a good informational support.

The experimental approach to integrated development environment design for a best complex collaboration technologies research is briefly described.

♣ Работа поддержана грантом РФФИ № 09-07-00407.