Управление модификацией программного кода корпоративных информационных систем (КИС)

Н.Г. Бернард, И.О. Красильников (МНПП НАМИП)

Сформулирована задача управления изменениями программного кода в процессе внедрения КИС на предприятии. Описан подход, позволяющий синхронизировать новые версии программного продукта, разработанный специалистами консалтинговой компании МНПП НАМИП.

Как организовать инициацию, учет и хранение изменений в программном коде информационных систем, чтобы можно было точно знать причину и источник этих изменений, а при необходимости отменять (откатывать) изменения, причем, не только последние?

После того, как мы наработали корпоративные стандарты и инструкции, этот вопрос для нас стал решенным. Практика показала, что такой способ управления модификацией кода, который мы используем в наших проектах внедрения, является надежным и полноценным – он был зафиксирован в последней редакции корпоративных стандартов, которые прошли сертификацию по ISO9001:2000. Однако, общаясь с коллегами – консультантами из других фирм, членами IT-ассоциаций и просто собеседниками, мы убедились в том, что тема не потеряла своей актуальности: бизнес-консультанты и разработчики КИС продолжают делиться своим опытом в этом вопросе. Это обстоятельство побудило нас рассказать в подробностях о нашем решении.

Проблема

Для чего нужно управлять изменениями кода? Во-первых, для того, чтобы синхронизировать модификации от разработчика системы (они выходят в свет как новые релизы, версии, подверсии и пр.) и модификации внедряющей компании, которые направлены на адаптацию системы к конкретному заказчику. Даже если разработчик сам внедряет свою систему, этими работами часто занимаются разные отделы.

Во-вторых, управление изменениями кода необходимо при совместной работе, когда каждый программист получает задание по изменению определенной части текста программы. В данном случае можно говорить о синхронизации внутренних модификаций.

В третьих, управление изменениями кода помогает разграничивать ответственность за изменения программ между внедряющей компанией и штатом программистов заказчика.

Итак, мы постараемся с техническими подробностями рассказать, как у нас организовано управление модификацией кода.

Синхронизация новых версий программного продукта и модификации внедряющей компании

Прежде чем описать механизм синхронизации, мы должны осветить некоторые моменты в структуре ПО для КИС. Для систем, которые поставляются с открытым кодом, такая структура является обычной. Чтобы не быть голословными мы опишем ее на примере системы, которую внедряем сами – iRenaissance.ERP. Ее разрабатывает американская компания Ross Systems – апологет автоматизации бизнес-процессов предприятия (www.rossinc.com, www.irenaissance.ru).

Блоки открытого кода поставляются в системе папок в виде отдельных файлов. Для исходных файлов предусмотрена папка с номером версии (например, "V51"). Когда возникает задача изменения/добавления функциональности, программист модифицирует исходный код и выкладывает измененный файл в папку "custom51", которую система загружает с более высоким приоритетом по сравнению с "V51". Естественно, название модифицированной копии исходного файла, лежащего в "custom51", должно оставаться неизменным, тем же что и у исходного.

Для файлов из новых версий, которые присылает разработчик – Ross Systems – предусмотрена специальная папка "ross51", приоритет которой ниже, чем у "custom51", но выше, чем у "V51". Соответственно, конечные пользователи имеют возможность дифференцировано обновлять версии системы, то есть добавлять только те файлы с обновлениями из новых версий, которые не нарушат сложившуюся функциональность системы.

При создании большого функционального блока/режима программист, как правило, создает новый отдельный файл, для которого нет исходного файла в "V51". Вновь созданные файлы также содержатся в папке "custom51" и прочитываются системой так же, как и модифицированные.

Эта структура хранения открытого кода позволяет производить дискретные замены, обновления версий и пр., когда требуется поставить не всю систему целиком, а лишь те файлы, которые подверглись изменениям. Синхронизация новой версии системы с модификациями внедряющей консалтинговой компании тоже становится дискретной – собственно слияние добавленной функциональности от разработчика и модификаций от консультанта затрагивает определенные конкретные файлы. К чести сказать, в поставках новых версий Ross Systems досконально перечисляет все новые части кода. Таким образом, поиск новых фрагментов (если он необходим) проходит с помощью такой текстовой карты изменений, и этот сервис со стороны поставщика весьма ценен для программистов. Технически слияние выглядит

как перенос фрагмента модифицированного кода в соответствующее место кода новой версии. Место модифицированного кода в новой версии, равно как и непротиворечивость работы программы в целом, мы проверяем сначала по описанию от Ross Systems, затем при прочтении самого кода, и после вставки кода в тестировании полученного файла. Тестирование является обязательным согласно нашим стандартам управления качеством. Весь процесс тестирования является контрольным моментом в управлении проектом внедрения, поэтому он подробно описан в соответствующем стандарте системы менеджмента качества (СМК). Результаты тестирования фиксируются по заданной форме.

В процессе исполнения проекта производится многоступенчатая проверка ПО и сопроводительной документации. Ниже приведена выдержка из корпоративного стандарта "управления проектом".

"...1) *Отладка программ*. Данную проверку проводят программисты, разрабатывающие конкретный программный код. Критерий успешной проверки – отсутствие ошибок компиляции и выполнения при работе всех режимов программы на произвольных введенных данных, а также соответствие всех выходных экранных и печатных форм проектной документации. Результаты проверки фиксируются в ежемесячных отчетах программистов и утверждаются руководителем проекта.

При положительных результатах тестирования программа и сопроводительная техническая документация передаются системному аналитику для дальнейшего тестирования. При отрицательных результатах тестирования программисты производят необходимую доработку программ.

2) *Тестирование работы прикладного ПО на соответствие требованиям потребителей*. Данную проверку на этапе разработки проводят системные аналитики, специализирующиеся в конкретных предметных областях и работающие в непосредственном контакте с представителями потребителей, для которых были разработаны тестируемые модули.

Тестирование проводится с помощью технической документации на ПО, предоставленной программистами. Для проведения тестирования потребительских функций системный аналитик подготавливает тестовые примеры для всех режимов программы – тестовые входные/выходные данные на основании проектной и технической документации.

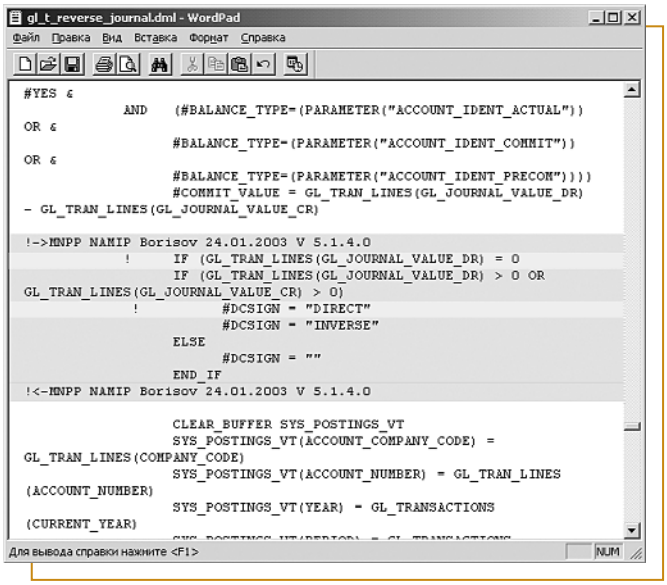
Критерий успешной проверки: соответствие выходных данных, полученных в результате работы программы, выходным данным тестовых примеров во всех режимах программы; соответствие описаний в технической документации реальной работе всех режимов ПО.

Результаты проверки фиксируются в ежемесячных отчетах системных аналитиков и утверждаются руководителем проекта. При положительных результатах тестирования прикладное ПО и сопроводительная техническая документация передаются потребителю для проведения обучения и тестовой и опытной эксплуатации. При отрицательных результатах тестирования системный аналитик возвращает ПО и техническую документацию на доработку (или дополнительную настройку) программистам."

Остается главный вопрос: выделение модифицированного фрагмента кода в программном файле. С этим связана не только возможность передачи модификаций в новую версию продукта, но и остальные вопросы управления изменениями кода.

Согласно корпоративным стандартам, каждый программист, создающий модифицированный блок, вносит в текст программы контрольную информацию в виде закомментированного сообщения. Причем так отмечается начало и конец блока. Например, в начале блока записывается значок начала блока (->), идентификатор компании и ФИО разработчика, дата изменения и номер внутренней версии файла (рисунок):

! -> MNPP NAMIP Borisov 20.05.03 v.5.1.3.17.



В конце блока пишется та же информация со значком завершения блока (<-). Номер внутренней версии файла меняется после каждого изменения с учетом ранга изменения во внутреннем номере версии есть 4 ранга:

1–2 – номер версии исходной системы (от разработчика);

3 – номер версии после серьезной до/переработки функциональности;

4 – номер версии после незначительных доработок или устранения ошибок.

Этот же номер внутренней версии стоит в самом начале файла и обновляется программистом при модификации кода.

Операции программиста по поддержанию версии файла являются обязательными и прописаны в документе *"Регламент управления изменениями"*, который является элементом общей СМК.

Надо заметить, что исходные строки кода, которые будут замещены новыми, мы не удаляем из текста, а инактивируем, переводя в комментарии. Это дает возможность быстро откатить изменения, если они при-

знаны "не соответствующими" после тестирования. В процессе работы с вышеописанной системой мечения новых блоков мы нашли сохранение прежнего кода очень удобным по еще одной (не очевидной) причине: назначение модификаций, выполненных одним программистом, становится более ясным для остальных.

Внутрикорпоративное сотрудничество

Нет нужды доказывать, что коллективная работа штата программистов требует идентификации блоков кода для нормального разделения области работ, определения структуры ответственности, возможности "популяризации" сделанных изменений, передачи наработок и текущего фронта работ другому программисту.

Для возможности одновременной коллективной работы наши программисты используют MS Visual SourceSafe, благодаря партнерству с Microsoft. Собственно, область работ программиста определяется в регулярном, обязательном для каждого из сотрудников документе – *ежемесячном плане*, который формируется по заявкам на доработки, а формат и структура ответственности – в других документах СМК.

Рассмотрим еще раз метку блока измененного кода: ! -> MNPP NAMIP. Borisov 20.05.03 v.5.1.3.17.

Символы начала и окончания блока служат для вычленения его из текста для переноса в текст программы следующей версии, номер версии – для отслеживания истории изменений и возможности последовательного отката (подробно о технологии отката – ниже).

Что касается идентификатора программиста, то само поддержание работоспособности модификаций было бы невозможно, если бы программист не был в состоянии сопровождать свои же части кода по прошествии любого отрезка времени: идентификатор программиста в метке блока нужен ему самому не меньше, чем его коллегам.

Кроме того, он нужен еще для формализации рабочих моментов – в своем *ежемесячном отчете* программист предоставляет результаты выполненных заданий в четких дискретных единицах, ссылаясь на "свои" блоки модифицированного кода. Результат выполненного задания фиксируется в документе – *своде заявок на доработку*, где программист указывает файл и версию изменения напротив задания на доработку. Так мы увязываем идентифицированный блок измененного кода с тем заданием, по которому он был сделан, и, в случае необходимости, можем откатить именно эту доработку, даже если соответствующие изменения кода распределены по текстам программ системы (что обычно и бывает).

Откат версий

Мы сохраняем в коде программы закомментированные строки кода предыдущей версии (вплоть до версии x.x.0.0.), и благодаря этому, как уже говорилось выше, получаем возможность отката версий на любую "глубину". Действительно, при определении "глубины" отката мы исследуем "закомментированные" строки кода с метками блоков и имеем возможность удалить все более старшие версии блоков, а эти строки "раскомментировать". Но в реальности мы производим последовательные откаты гораздо проще: находим в архиве версию файла, соответствующую дате нужного изменения, и заменяем ей тот, который находится в папке "custom51". Естественно, что для этого программисты и системные аналитики ведут четко "прописанную" архивацию – сохраняют предыдущие версии файлов в папках, названия которых включают даты обновления. Например, 20 мая 2003 г., перед тем как переместить в папку "custom51" обновленный файл, мы создаем папку "03-05-20" в директории архива и переносим туда одноименный файл с текущей версией кода. Работа с датами изменений возможна благодаря тому, что мы фиксируем ее в метке блока изменений кода.

Разграничение ответственности между заказчиком и исполнителем

Особое значение приобретает такая система мечения блоков кода, когда речь идет о передаче изменений заказчику. По описанной выше технологии создается структурированный по дате архив всех изменений. Даты – это как раз тот параметр, который наиболее часто фигурирует в процессе взаимодействия с операторами и службой АСУ заказчика. Надо заметить, что сотрудники отдела АСУ заказчика также обучены программированию на языке системы iRenaissance.ERP (язык 4 поколения Gembase 4GL) и имеют право модифицировать код параллельно с нами. Но модификации "от заказчика" могут попасть в наши версии только официальным путем, с обязательной регистрацией в наших реестрах. Естественно, что эти модификации также проходят стандартное тестирование и получают метку блоков с именем компании заказчика в качестве идентификатора.

Таким образом, у нас сохраняется полная история модификации кода, которая в случае разногласий с заказчиком, может дать четкий ответ на вопрос о происхождении "не соответствующего" блока измененного текста программ.

Когда заходит разговор о недостатках и преимуществах открытого кода, вопрос разделения ответственности между программистами автоматизируемого предприятия и внедряющей компании всплывает первым. Однако в нашей практике пока еще не было случая таких претензий со стороны заказчика, может быть потому, что помимо четкого управления изменениями, мы стараемся сделать атмосферу общения с заказчиком максимально уважительной. Для нас и программистов, и консультантов, каждый заказчик – VIP-клиент. Это уже не прописано в наших стандартах, но оформлено в миссии компании и вошло в принципы корпоративной этики.

Бернард Николай Геннадьевич *– системный аналитик,*
Красильников Илья Олегович *– руководитель проектов консалтинговой компании "МНПП НАМИП".*
Контактный телефон (095) 737-33-37, 912-73-63. E-mail: bernard@namip.ru, ilya@namip.ru Http://www.namip.ru