[8] *Czarnecki K., Eisenecker U.* Generative Programming: Methods, Tools, and Applications. Reading, Mass.: Addison Wesley Longman, 2000. 864 p.

[9] *Day D., Priestley M., Schell D. A.* Introduction to the Darwin Information Typing Architecture // Toward portable technical information. http://www.106.ibm.com/developerworks/xml/library/x-dita1.

[10] *Eriksson M., B orstler J., Borg K.* The PLUSS Approach — Domain Modeling with Features, Use Cases and Use Case Realizations // Proc. Software Product Line Conference'2005, LNCS 3714. 2005. P. 33–44.

[11] *Greenfield J., Short K., Cook S., Kent S.* Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. Indianapolis, Indiana: Wiley Publishing, Inc., 2004. 696 p.

[12] *Griss M., Favaro J., d'Alessandro M.* Integrating Feature Modeling with RSEB // IEEE Proceedings of Fifth International Conference on Software Reuse. 1998. P. 76–85

[13] *Harrison N.* The Darwin Information Typing Architecture (DITA): applications for globalization // 2005 IEEE International Professional Communication Conference Proceedings. 2005.

[14] *Jarzabek S., Bassett P., Hongyu Zhang, Weishan Zhang.* XVCL: XML-based variant configuration language // 25th International Conference on Software Engineering. 2003. P. 810–811.

[15] *Kang K., Cohen S., Hess J., Novak J.* et al. Feature-Oriented Domain Analysis (FODA) Feasibility Study // Technical Report, CMU/SEI–90–TR–21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh. 1990.

[16] *Krueger C.* Easing the Transition to Software Mass Customization // Proc. 4th Int'l Workshop on Software Product Family Eng. 2001. P. 282–293.

[17] *Kyo C. Kang J. L., Donohoe P.* Feature-Oriented Product Line Engineering // IEEE Software. July/August 2002. P. 58–65.

[18] *Lee K., Kang K., Lee J.* Concepts and Guidelines of Feature Modeling for Product Line Software Engineering // Proc. 7th Int'l Conf. Software Reuse, LNCS 2319. 2002. P. 62–77.

[19] *Marques M.* Single-sourcing with FrameMaker // TECHWR-L Magazine Online. http://www.techwr-l.com/techwhirl/magazine/ technical/singlesourcing.html.

[20] *Tolvanen J., Kelly S.* Defining Domain-Specific Modeling Languages to Automate Product Derivation: Collected Experiences // Proc. Software Product Line Conference'2005, LNCS 3714. 2005. P. 198–209.

[21] *Walsh N., Muellner L.* DocBook: The Definitive Guide. O'Reilly, 2003.

Опыт усовершенствования и стандартизации процесса создания ПО цифровых телефонных станций

В. И. Кияев
kiyaev@tercom.ru

Д. М. Кищенко
denis.m.kishenko@intel.com

И. С. Окомин
ilya.s.okomin@intel.com

В статье представлен опыт усовершенствования процесса разработки программного обеспечения для цифровых телефонных станций в департаменте телекоммуникаций ЗАО «ЛАНИТ-ТЕРКОМ», отделе разработки рабочего места оператора (РМО). Улучшение процесса разработки именно в этом отделе департамента стало критичным для всего проекта, поскольку (а) требования клиентов к продукту в целом сильно варьировались именно для этой его части; (б) большое количество интеграционных проблем всего проекта было сосредоточено здесь же; (в) именно в рамках РМО было целесообразно создать процедуры и программные средства иммитационного тестирования всего продукта. В результате удалось успешно решить непростые задачи по стандартизации и совершенствованию качества процесса, распространению и сопровождению программных продуктов среди клиентов (около 500 АЦТС в разных городах России).

Введение

Совершенствование процесса разработки (Software Process Improvement) — это изучение существующего производства и внесение изменений в целях улучшения качества процесса, продукта и/или снижения издержек и времени разработки [7]. Особенность этой деятельности заключается в том, что она, как правило, происходит «на ходу» — в то же самое время, когда компания продолжает работать и выпускать продукцию (за исключением периода формирования и настройки процессов при основании или реинжиниринге компании), т. е. компанию нельзя «закрыть на реконструкцию» [3]. То, как компания конкретно должна осуществить это, не прописано в имеющихся в этой области стандартах [2, 14, 15, 11] и является уникальным опытом, который каждая компания накапливает, превращает в знания и реализует самостоятельно.

Улучшение процессов разработки ПО особенно актуально для российских компаний в силу специфичности их положения на мировом программном рынке, а также из-за различных внутренних особенностей [16, 17]. По организации эффективных процессов наши компании существенно отстают от западных и, в частности, американских компаний, несмотря на отдельные успехи[1].

В силу этих причин описание успешного опыта российских компаний в области усовершенствования процессов разработки является чрезвычайно актуальным. В этой области существуют публикации, например [12, 13, 8, 6], однако их явно недостаточно и, к сожалению, отсутствует единое всероссийское сообщество, которое бы собирало, анализировало, обсуждало и распространяло подобные материалы.

В настоящей работе представлен опыт совершенствования процесса разработки ПО для автоматических цифровых телефонных станций (АЦТС) в департаменте телекоммуникаций ЗАО «ЛАНИТ-ТЕРКОМ». Наш коллектив на базе НИИ математики и механики ЛГУ и ЛНПО «Красная заря» в 80-х годах начал разработку ПО для АЦТС. Для коммерциализации и развития этих и других работ была образована компания ЗАО «ЛАНИТ-ТЕРКОМ». В результате было создано ПО для многих телефонных станций военного и гражданского назначения. К настоящему моменту нашим главным продуктом является ПО для АЦТС «КВАНТ-Е-СОКОЛ», аппаратуру для которой изготавливает завод «Сокол» (г. Белгород). Мы выпустили и установили около 500 таких станций по разным городам и областям России в общей сложности более чем для 400 000 абонентов.

Сейчас департамент телекоммуникаций ЗАО «ЛАНИТ-ТЕРКОМ» насчитывает 6 отделов. Переход от выпуска отдельных уникальных разработок к серийному производству телефонных станций, распространяемых по всей России, потребовал от нас существенной реорганизации процесса разработки ПО.

При этом одним из самых приоритетных направлений всего проекта оказалось улучшение качества разработки программных компонент для ПО рабочих мест операторов станций (РМО) ввиду следующих причин:

- большой вариативности требований заказчиков к этим компонентам (они, единственные из всех компонент ПО АЦТС, имеют пользовательский интерфейс, и этот интерефейс для разных заказчиков постоянно приходится «чуть-чуть» изменять — в результате создается большое количество различных версий компонент для разных клиентов, которые нужно систематизировать, тестировать, доставлять заказчику и сопровождать в процессе постоянной эксплуатации);

- в рамках этого подпроекта пришлось решать интеграционные проблемы по соответствию версий РМО версиям других компонент, контролировать корректность генерации данных для станции;

- в рамках этого же подпроекта пришлось решать также задачу имитации нагрузки для всей станции и соответствующего тестирования всех ее программных компонент (этот модуль впоследствие стал частью РМО)[2].

[1] Так, в 2003 году первая в мире российская компания Luxoft получила двойной сертификат 5-го уровня стандартов CMM и CMMI, в 2004 году 5-й уровень CMMI получило Санкт-Петербургское отделение корпорации Моторола.

[2] Мы действовали в рамках парадигмы Organization Pull [18], т. е. разрешали самые насущные задачи совершенствования процесса в рамках одного отдела всего департамента. Парадигма Technology Push, подразумевающая комплексную реорганизацию всего процесса в соответствии с определенными технологиями и стандартами, на текущий момент для нас оказалась неприемлемой, поскольку мы не могли существенно приостановить и детально исследовать процесс разработки, что необходимо при таком подходе. Разумеется, мы прикладывали максимум усилий, чтобы сделать наш опыт общим достоянием всего

Ещё раз повторим — в течение пяти лет было поставлено более пятисот АЦТС в разные регионы России. Каждая из этих станций требовала индивидуального сопровождения — например, практически никогда не было возможости обновить сразу всё программное обеспечение, так как станции постоянно находятся под рабочей нагрузкой. В связи с этим заказчики могли расчитывать лишь на исправление выявленных ошибок и на добавление нужных функций. Таким образом, вполне традиционная задача управления версиями продукта превратилась в нашем случае в существенно более трудную задачу одновременного мультиверсионного контроля нескольких сотен вариантов на работающих станциях.

В данной работе мы представляем описание улучшения процесса в отделе разработки РМО. Мы активно пользовались опытом других подразделений ЗАО «ЛАНИТ-ТЕРКОМ», в частности департамента реинжиниринга [12, 13, 8, 6, 9]. Следуя рекомендациям стандарта CMM [14], мы начали улучшение процесса с наладки двух важнейших для нас ключевых областей CMM 2-го уровня — конфигурационного управления (версионный контроль, управление сборкой) и обеспечения качества разработки ПО (стандартизация процесса тестирования, наладка регрессионного тестирования). Необходимо отметить, что мы использовали CMM и ряд других стандартов не с целью немедленной сертификации нашего процесса, а для того, чтобы реально изменить процесс к лучшему[3].

1. ПО автоматических цифровых телефонных станций

1.1. Состав программного обеспечения АЦТС

Комплексное программное обеспечение АЦТС можно условно разбить на три взаимосвязанные части.

департамента, поскольку и в других отделах проводилось постепенное улучшение процессов, но не так интенсивно, как в отделе ПО РМО.

[3] Действительное улучшение процесса и деятельность по получению какого-либо сертификата качества (CMM/CMMI, ISO 9000 и др.) — это связанные, но не тождественные виды деятельности. Получение сертификата часто оказывается формальностью, «процессом на бумаге», а действительное улучшения процессов — это системное совершенствование производства, управления и изменение менталитета работников компании [1] — всё это может происходить постепенно, «набирая обороты» для того, чтобы создать «задел» для дальнейших работ по получению соответствующего сертификата качества.

1. Функциональное программное обеспечение (ФПО) — программное обеспечение, записываемое в АЦТС на управляющие микросхемы памяти с помощью специального устройства, которое называется «программатор» (в дальнейшем такая операция будет называться «прожигание»). ФПО отвечает за установление всех видов соединений/разъединений, обеспечение дополнительных услуг, сканеры технических средств, обеспечение технического обслуживания, включая начальный запуск станции.

2. Системное программное обеспечение (СПО) — программное обеспечение, «прожигаемое» в АЦТС, которое включает в себя программные средства, обеспечивающие работу ФПО в распределенной вычислительной среде (операционная среда распределенного комплекса вычислительных средств, программы обмена по локальной сети, диспетчер объектов).

3. Программное обеспечение рабочего места оператора (РМО) для технического обслуживания и эксплуатации АЦТС, которое функционирует на непосредственно подключенном к АЦТС или удаленном рабочем месте, в качестве которого используется персональная ЭВМ. Отметим, что существенную часть РМО составляет база данных АЦТС. И если программное обеспечение станции является в основном универсальным, одинаковым для всех станций данного типа, то БД индивидуальны для каждой АЦТС. БД формируется при запуске станции в эксплуатацию, а затем может изменяться как по требованию пользователей, так и обслуживающего персонала. Разработчикам отдела РМО удалось в большой части автоматизировать процесс генерации БД, однако её сопровождение является важным аспектом системы версионного контроля.

Конечный продукт представляет собой интеграцию вышеперечисленых компонент ПО, взаимодействующих на разных уровнях. Первые две части ПО являются встроенными, третья часть — интерфейсная.

1.2. Комплекс РМО

Одной из особенностей такого ПО АЦТС является то, что одна его часть функционирует непосредственно в станции, обеспечивая

работу с внутренними абонентами и взаимодействие с телефонной сетью, которую она обслуживает, другая часть размещается на компьютере, соединенном со станцией (РМО).

Комплекс РМО позволяет выполнять основные задачи обслуживания телефонной станции во время ее работы: управление, тестирование, диагностику и мониторинг различных программных и аппаратных элементов, а также сбор статистической и тарификационной информации и т. п. Кроме того, комплекс поддерживает ряд дополнительных возможностей: удаленную работу со станцией через протокол TCP/IP, авторизацию пользователей, протоколирование действий оператора, выполнение задач по расписанию, синхронизацию времени и т. д.

РМО имеет серверную и клиентскую части. Серверная часть реализована отдельным приложением *«Серверы РМО»*, которое представляет собой набор нескольких служб, работающих в фоновом режиме. Приложение периодически общается с ПО станции, выполняя текущие задачи:

- «подгрузку» в станцию необходимых программных модулей по требованию ПО станции;
- получение от встроенного ПО станции сообщений о текущем состоянии станции, а также сообщений межобъектного взаимодействия и демонстрацию их оператору;
- синхронизацию текущего суточного времени между ПК и станцией и т. д.

Клиентская сторона представляет собой пакет взаимодействующих между собой Windows-приложений, объединенных общим приложением *«Клиент РМО»*. С помощью «Клиента РМО» осуществляется подключение к определенной станции — тем самым предоставляется возможность производить обмен сообщениями приложений со станцией. Приложения клиентской части служат для выполнения следующих задач мониторинга и диагностики АТС: работа и изменение данных абонента, мониторинг состояния оборудования АТС, обслуживание и управление абонентскими линиями и пр. Некоторые программы, входящие в состав клиентской части РМО, используют «Серверы РМО» для получения необходимой им информации.

Каждая станция комплектуется обслуживающим компьютером с конфигурацией, достаточной для функционирования пакета программ РМО. Одно РМО может обслуживать несколько АЦТС.

2. Проблемы процесса разработки РМО

В ходе инспекции существующего положения дел с целью формирования видения ситуации «как есть» (as is) мы провели тщательное исследование и анализ предметной и функциональной областей деятельности отдела РМО. В качестве инструментов исследования применялись опрос и анкетирование сотрудников, изучение имеющейся методической и технической документации, требований стандарта CMM для реализации соответствующих ключевых областей процесса, моделирование процессов и процедур [14]. К сожалению, рамки настоящей статьи не позволяют в полной мере описать исследовательскую составляющую часть работы, что само по себе представляет самостоятельный методологический интерес. В результате многоаспектного исследования были выделены и идентифицированы следующие существенные системные проблемы разработки РМО АЦТС.

1. Сам процесс разработки ПО РМО в целом не был описан и формализован в виде даже простой модели, показывающей взаимосвязь различных составляющих процесса.
2. Не было единой дисциплины версионного контроля исходных текстов проекта.
3. Жизненный цикл версии РМО был обозначен нечетко. Такие важные стадии, как начало разработки новой версии, «замораживание» функциональности, тестирование, сопровождение и закрытие версии были «размыты», либо вовсе отсутствовали.
4. Существовали значительные трудности с распространением и учетом версий среди сотен клиентов: обновления высылались в виде отдельных файлов, не было централизованной процедуры регистрации обновлений. Через некотрое время оказывалось невозможным определить точную конфигурацию продукта у клиента и воспроизвести ошибки — в результате очень часто приходилось ездить в командировки, чтобы разбираться с ситуациями на месте.

5. Не был упорядочен версионный контроль, не автоматизирована сборка продуктов для всего множества клиентов, что сильно затрудняло одновременную разработку нескольких версий продукта, а также превращало процесс создание новой вресии в трудный и очень «нервный» процесс.

6. Отсутствовала регулярная проверка целостности версий для каждого клиента и регрессионное тестирование, что приводило к постоянным проблемам при сборке проекта и затягивало его реализацию, а также регрессионное тестирование.

7. Часто возникала проблема несоответствия версий ФПО, БД АЦТС и РМО в рамках одной поставки.

8. Отсутствовала единая система обработки протоколов замечаний с завода и с объектов, где были установлены версии ПО. Эти замечания представляют большую ценность, так как они описывают ошибки и ситуации (коллизии), возникающие в процессе эксплуатации реальной АЦТС в полной комплектации, работающей под нагрузкой.

Таким образом, в ходе проведённого исследования ситуации «как есть» было выявлено достаточно большое количество системных причин, создающих предпосылки для организационной и производственной неразберихи, потери реального контроля ситуации, что в конечном счёте с неизбежностью приводит к реализации некачественного продукта. Формирование и анализ состояния «как есть» позволило нам построить видение перспективы «как должно быть» (as to be), разработать конкретные планы воплощения этой перспективы и в большой степени реализовать их. На рис. 1 и 2 представлена общая схема процесса разработки ПО РМО.

3. Совершенствование процесса

3.1. Управление жизненным циклом РМО и версионный контроль

Прежде всего мы решили навести порядок среди активов разработки — исходных текстов проекта. К этому времени в нашем проекте уже использовалось средство версионного контроля

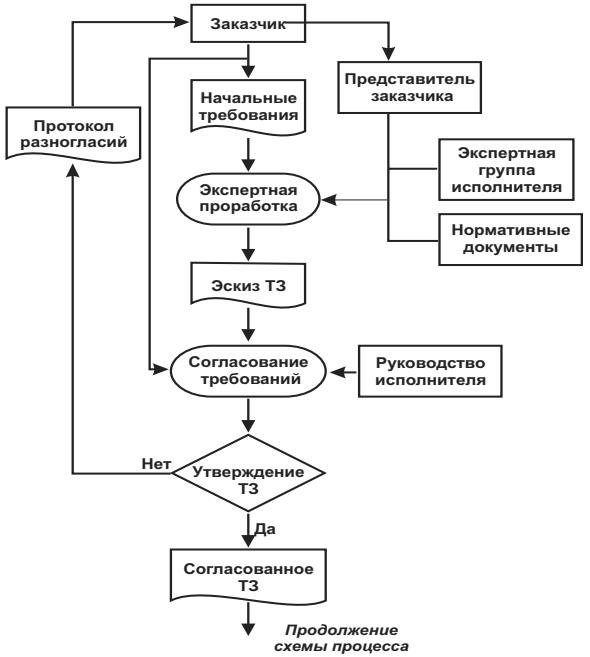

Рис. 1. Общая схема процесса разработки ПО РМО

`Microsoft Visual SourceSafe` (`VSS`), но не существовало единой, согласованной для всех клиентов и для всех составных частей ПО РМО политики версионирования. Это было вызвано рядом особенностей разработки ПО для телефонных станций, что постепенно привело к сложившемуся неудовлетворительному стилю работы.

Среди разработчиков существует практика исправления ошибок путем простой замены старых модулей (приложений, библиотек и др.) на исправленные или модифицированные. Это происходит в тех случаях, когда механизм сопровождения версий недостаточно отработан. Кроме того, в целях проверок или при попытках устранить ошибку неопытные пользователи РМО могут случайно или намеренно изменить текстовые файлы РМО без согласования с разработчиками, что может привести к сбоям в работе программной системы.

Рис. 2. Общая схема процесса разработки ПО РМО (окончание)

В результате выполненных исследований стало понятно, что главной концептуальной основой процесса версионирования является жизненный цикл версии продукта. В нашем проекте, в силу исторических причин, существовало два разных продукта, которые мы условно назвали РМО 1 и РМО 1.5 (рис. 3).

Каждый из этих продуктов развивается — «наращивается» функциональность и исправляются ошибки. Новая функциональность часто появляется в ответ на запрос конкретного клиента по текущему проекту, ошибки часто исправляются также по «критическим» ситуациям и замечаниям представителей заказчика. Мы интегрируем все эти изменения в следующую версию РМО и стремимся как можно скорее перевести всех наших клиентов на эту версию, чтобы не поддерживать

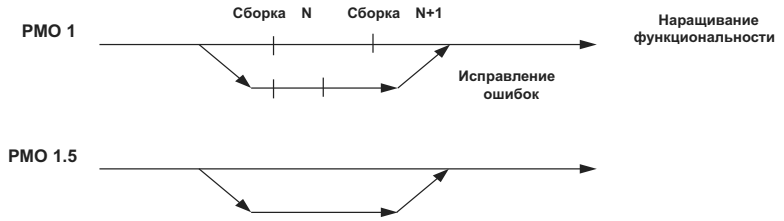

Рис. 3. Схема развития программного продукта РМО

отдельные варианты продукта для каждого клиента в отдельности.

Однако бывает, что развитие функциональности в интересах какого-либо клиента происходит достаточно долго, и тогда мы создаем для него отдельную версию продукта, а в прежней только исправляем ошибки. После того как новая версия стабилизируется, мы объединяем две этих «ветки». При существовавшей схеме версионного контроля было довольно трудно установить момент стабилизации и отследить все «узкие места» объединения веток отдельных клиентов в базовую версию продукта.

Подобные действия сильно усложняют обнаружение и исправление ошибок в клиентском РМО, поскольку на базовом «стволе» разрастается пышная версионная «крона», и в большинстве случаев невозможно идентифицировать имеющуюся у заказчика версию. Для проверки состояния версии и перехода к более качественному сопровождению продукта была разработана и внедрена система контроля версий РМО. Суть заключается в том, чтобы поставить под версионный контроль как РМО в целом, так и каждый файл, входящий в клиентскую поставку, в отдельности и оснастить РМО средствами проверки состояния версии и обнаружения «исправленных» модулей.

Для реализации системы контроля в поставку РМО был добавлен дополнительный контрольный файл, который содержит информацию обо всех файлах РМО. Система устроена достаточно просто, но приносит хороший результат. Контрольный файл генерируется автоматически при сборке версии и содержит список файлов, входящих в инсталляцию, где для каждого файла указан тип верификации и дополнительные параметры. Системой контроля предусмотрено два способа верификации.

1. Контрольная сумма содержимого файла. Контрольная сумма вычисляется во время сборки версии PMO и заносится в контрольный файл. При проверке состояния версии PMO реальная контрольная сумма сравнивается с записанной. Метод применим для всех видов файлов.

2. Запись номера версии внутри файла — «прошивка» версии. В определенное место файла, после его сборки, записывается информация о версии. Если файл был создан, минуя систему сборки случайно или умышленно, в нем будет отсутствовать такая запись. Метод применим только для бинарных файлов типа `.exe`, `.dll` и `.bpl`. Преимущество метода в том, что проверка версии занимает меньше времени, чем при подсчете контрольной суммы файла.

Однако не все файлы можно верифицировать указанным методом, поскольку их содержимое может изменяться в процессе работы PMO, например, административные файлы со списком пользователей и паролями. Кроме информации об отдельных файлах, контрольный файл содержит такие данные, как версия PMO и дата сборки. Для таких ситуаций функции проверки состояния версии реализованы в специальной библиотеке, входящей в общую поставку PMO — теперь с её помощью пользователь может получить исчерпывающую информацию о составе версии и ее состоянии.

Используя разработанный механизм и опыт другого проекта нашей компании — `RescueWare` [12, 8, 9] — мы формализовали и разработали четкую структуру проекта в `VSS`, а также наладили автоматическую сборку проекта на основе `VSS`.

3.2. Управление сборкой

Одной из важных задач конфигурационного управления является управление сборками системы (Build Management). Сборкой называют процесс компиляции и связывания программных компонентов в единую исполняемую программу [7]. Одним из наиболее популярных средств сборки является утилита `make`. Существуют и более сложные системы сборки, например, продукты `OpenMake`, `ANT`, `FinalBuild`. Однако практическое использование этих инструментов ограничено по следующим причинам: при разработке систем сборки крупных проектов таких стандартных средств обычно

Group	Count	Commit	Developed	Installation
PMO 1	92 + 341	- 2		
PMO 1.5	96 + 594			

PMO 1

File (92)	Commit	Developed	Owner
Abnfind.exe	6	6	Любимов Борис
AbnStates.exe	23	23	Дмитрий Козлов
Abonent.exe	63	63	Баканов Антон
AbonentISDN.exe	15	15	Баканов Антон
Access.bpl	28	28	Баканов Антон
ADCSubscribers.exe	3	3	Баканов Антон
AlarmPanel.exe	1	1	Кищенко Денис
Chat.exe	3	3	Мясников Андрей
Client.exe	29	29	Сулима Феликс
CMMEdit.exe	3	3	Бугайченко Дмитрий
CmmExec.exe	20	20	Наганов Михаил
CmmExplorer.exe	11	11	Наганов Ми

PMO 1.5

File (96)	Commit	Developed	Owner
Abnfind.exe	6	6	Любимов Борис
AbnStates.exe	23	23	Дмитрий Козлов
Abonent.exe	63	63	Баканов Антон
AbonentISDN.exe	15	15	Баканов Антон
Access.bpl	28	28	Баканов Антон
AlarmPanel.exe	1	1	Кищенко Денис
BoardTester.bpl	2	2	Бугайченко Дмитрий
Chat.exe	3	3	Мясников Андрей
Client.exe	29	29	Сулима Феликс
CMMEdit.exe	3	3	Бугайченко Дмитрий

Рис. 4. Пример отчёта сборки версии

недостаточно, так как процесс сборки в каждом проекте имеет много индивидуальных особенностей. Для создания специализированных «сборщиков» проще написать и отладить несколько скриптов, чем осваивать и использовать новый продукт. Поэтому во многих крупных проектах создаются собственные средства сборки — примерами могут служить open-source-проект `Mozilla`[4], проект ЗАО «ЛАНИТ-ТЕРКОМ» `RescueWare` [12, 8, 9]. У нас был именно такой случай.

Функциональной особенностью созданного нами «сборщика» была разработка специального, удобного формата отчетов, которые создавались в процессе сборки и рассылались всем участникам проекта — пример такого отчёта представлен на рис. 4. На основе такого автоматического инструмента мы наладили процедуру регулярной сборки (Continues Integration) [10].

[4] http://www.mozilla.org.

3.3. Распространение версий

Одной из существенных проблем, как было отмечено выше, были значительные трудности с распространением «рабочих» версий среди множества клиентов и учетом текущих изменений этих версий. Для наведения порядка с учетом обновлений, высылаемых клиентам, были предприняты следующие шаги.

1. Был создан Интернет-сайт, с которого каждый клиент мог сам «скачать» последнюю версию комплекса РМО.

2. Большие организационные усилия были потрачены на убеждения и разъяснения клиентам того, что все обновления целесообразно получать только в рамках новой целостной версии продукта, а не в виде отдельных файлов, высылаемых по электронной почте.

3. Было решено в некоторых случая все-таки идти навстречу клиентам и высылать им обновленные отдельные файлы: иногда клиент не может ждать выхода новой версии продукта, потому что, например, у него проявилась какая-то существенная критическая ошибка, которую нужно срочно исправить, чтобы его система продолжала нормальное функционирование[5]. Для того, чтобы впоследствии можно было воспроизвести таким образом обновленную конфигурацию продукта у этого клиента (например, при анализе и исправлении следующих выявленных ошибок), использовали систему поддержки версионной информации в работающей (бинарной) конфигурации нашего продукта (см. раздел 3.1). Теперь, если у клиента возникают проблемы или вопросы относительно работы ПО, он может воспользоваться специально разработанной и встроенной в РМО функцией «Послать письмо в службу поддержки», которая автоматически выполнит проверку состояния версии и сформирует письмо с полученными результатами в виде файла-отчета.

[5]Конечно, «физически» клиент, как правило, может дождаться новой версии, где эта ошибка будет исправлена, но в целях установления доверительных отношений с заказчиком психологически иногда оказывается очень полезно немедленно исправить ошибку в том варианте ПО, который в настоящий момент находится у него.

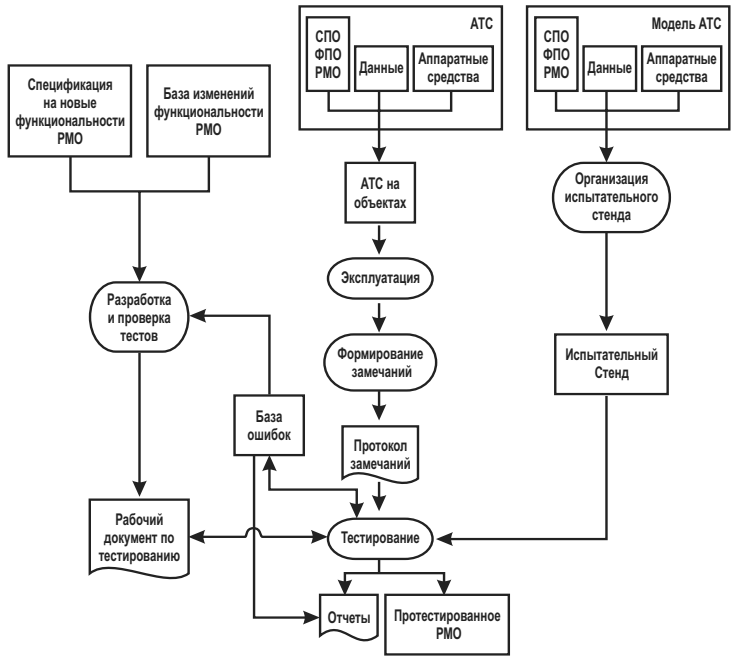

Рис. 5. Процесс тестирования комплекса РМО

3.4. Общая организация тестирования

За время разработки и использования комплекса РМО сложилась схема, согласно которой происходит проверка качества продуктов и разработка новых тестов. В рамках улучшения процесса тестирования мы решили формализовать процесс в виде блок-схемы (рис. 5), попутно упорядочив и регламентировав ее отдельные фрагменты.

В настоящее время для методологического обеспечения процесса тестирования версии РМО, что прямо требуется в ряде ключевых областей процесса в соответствии с требованиями стандарта CMM [14], используется *«Рабочий документ по тестированию»*, разработанный нами в ходе выполняемой работы. Это пакет методических документов, который состоит из двух частей: констатирующей (описательной) и тестовой (технологической). Описательная

часть может быть использована в качестве основы для разработки внутреннего стандарта. «Рабочий документ по тестированию» подробно освещает процесс проверки и тестирования версии РМО и имеет следующие разделы:

- определение и назначение программного комплекса РМО;
- ресурсы, необходимые для работы РМО;
- состав пакета РМО с распределением ответственности разработчиков;
- описание набора тестов для проверки корректности работы компонент клиентской и серверной частей РМО;
- описание процесса учета ошибок;
- описание методики тестирования, используемой в группе тестирования по части РМО;
- набор тестов для проверки конфигурационной базы данных РМО `ats.mdb`, который предполагает непосредственный анализ таблиц, выявляющий коллизии, и обеспечивает гарантию качества при корректном взаимодействии некоторых приложений РМО с ней.

В процессе жизненного цикла ПО РМО «Рабочий документ по тестированию» не остается неизменным. Он пополняется тестами для разрешения новых возникших некорректных ситуаций, при этом анализируется используемая методика тестирования на предмет ее оптимальности. Данный документ можно считать методологической основой для организации и реализации процесса тестирования РМО, а также для разработки и создания автоматических тестов и «внутренних стандартов» на тестирование. Когда выявляется новая ошибка, формируется способ ее проверки, затем строится тест, который заносится в набор тестов «Рабочего документа по тестированию». Шаблон теста представляет собой набор следующих обязательных элементов:

- мнемоничное название, отражающее суть теста;
- начальные условия, необходимые для проведения теста;

- тело теста, включающее в себя последовательность действий, которые проверяют возможную конфликтную ситуацию в работе ПО;
- результат, ожидаемый после выполнения последовательности действий.

Помимо тестирования коллизий, существует тестирование интерфейсной и функциональной частей компонент РМО, что также реализовано в виде набора тестов. По мере изменения компонент ПО разрабатываются новые тесты, покрывающие эти изменения. Полученные тесты автоматически заносятся в «Рабочий документ по тестированию» и при тестировании последующих версий пакет РМО тестируется уже в расширенном варианте, при этом выделенный сотрудник отдела контролирует этот процесс.

В целом же в отделе РМО были реализованы следующие организационно-функциональные меры по наладке тестирования.

1. Создана штатная группа тестирования, работники которой заняты только тестированием.
2. В проекте развернута база данных ошибок и налажена работа с ней.
3. Выделен работник отдела, ответственный за координацию всего процесса тестирования, в частности, отвечающий за создание заданий на тестирование. Создан регламент по созданию таких заданий в письменной форме: при получении замечаний с завода, при подготовке новой функциональности компонент РМО и т. д.
4. Создан документ о регрессионном тестировании РМО [5], подробно описывающий то, что нужно сделать (стандартные шаги тестировщика) при проверке компонент РМО на «нерегресс».
5. Создан регламент по проверке всех компонент РМО и их совместимости с компонентами ФПО при подготовке данных для станции. Сотрудникам департамента телекоммуникаций, выполняющим эту работу, вменено в обязанность следовать этому регламенту (важно отметить, что эту работу выполняют не только в отделе РМО).

6. Создан имитатор нагрузки, который является частью РМО (используется в департаменте телекоммуникаций и не входит в поставку заказчику) и позволяет всем подразделениям департамента тестировать свои компоненты в условиях, максимально приближенных к «боевым».

3.5. Имитатор станционной нагрузки

В процессе тестирования ПО АЦТС на тестовом стенде для проверки некоторых функциональностей необходимо создать режим работы, максимально приближающий нагрузку реально работающих АЦТС.

Для создания автоматической имитации абонентских соединений, обеспечивающих станцию нагрузкой, разработано программное приложение «Имитатор станционной нагрузки». С помощью имитируемых соединений можно получать данные статистики и тарификации в больших объемах (большие серии разговоров, формирование записей о длительных разговорах), чем можно задать при ручном установлении соединений. Например, для проверки корректной работы приложений просмотра тарификации или телетрафика достаточно всего лишь наличия тестовых баз данных, которые можно сгенерировать «вручную». Но для проверки серверов этих служб, а следовательно, и процесса размещения и изменения данных в памяти станции, необходима непосредственная работа АЦТС.

Отметим, что для решения проблемы создания искусственной нагрузки существуют программно-аппаратные комплексы АВИСТЕН-М и ПРИЗМА-М[6]. Разработанное нами решение выгодно отличается от указанных тем, что оно является исключительно программной реализацией нагрузки без использования дополнительного дорогостоящего оборудования. Реализованная функциональность процесса тестирования ПО РМО позволяет расширить возможности имитатора нагрузки до отдельного тестового комплекса, позволяющего диагностировать результаты соединений по различным параметрам. В зависимости от типа соединения можно формировать различные виды нагрузки на разные элементы АЦТС: внутримодульная нагрузка, внутристанционная межмодульная нагрузка, межстанционная нагрузка и т. д.

[6] http://www.loniis.ru:8101/RUS/products/service/service01.htm, http://www.loniis.ru:8101/RUS/products/service/service03.htm.

Во время работы «Имитатора станционной нагрузки» осуществляется автоматическая генерация данных серий имитации, инициация и управление протоколом имитации, получение результатов серии для их дальнейшего анализа. Приложение интегрировано в комплекс с клиентской частью РМО, что позволяет работать с базами данных различных станций, производить обмен сообщениями со станцией, а также использовать другие приложения, входящие в пакет РМО. Интерфейс «Имитатора станционной нагрузки» представлен в виде окна с закладками, определяющими тип нагрузки, с полями для определения параметров, характерных для каждого типа нагрузки, нескольких функциональных кнопок и окна просмотра результата прохождения серии.

С использованием имитации разговора стала возможной автоматическая проверка соединений и разговорных трактов для различных типов соединений, характерных для заданных станций, проверка сигнализации в шлейфе. Стало возможным проверять функциональность ПО АЦТС для станции, находящейся под нагрузкой. Всё это значительно (до 35–40%) сократило время тестирования и уменьшило число сотрудников, занимавшихся раньше проведением тестирования станции «под нагрузкой».

4. Заключение

Описанные в статье исследования и системные улучшения процесса разработки ПО РМО позволили нам решить задачу выпуска и распространения АЦТС для резко возросшего числа клиентов, имеющих к тому же большую географическую распределенность. Идентификация процессов и процедур, созданные модели, внутренние стандарты, регламенты, более чёткое распределение ответственности за реализацию и контроль процедур привели к тому, что процесс разработки ПО РМО в упомянутых выше ключевых областях 2-го уровня CMM стал управляемым («дисциплинированным»), что и требует стандарт [14]. К настоящему времени прекратился поток жалоб на качество компонент РМО и, в целом, на ПО АЦТС. Конечно, улучшение качества процесса разработки ПО происходило не только в нашем отделе, но и в других отделах департамента, а также и на заводе-изготовителе аппаратуры при установке ПО. Но именно в нашем отделе налаженные процедуры оказались

наиболее формализованными и пригодными для стандартизации. Мы считаем, что в отделе получен чрезвычайно полезный опыт и создан хороший задел для дальнейшего совершенствования процесса разработки сложного комплексного программного обеспечения в рамках всего департамента.

В заключение мы выражаем искреннюю признательность доценту Д. В. Кознову, принявшему деятельное участие в подготовке текста статьи, директору департамента телекоммуникаций ЗАО «ЛАНИТ-ТЕРКОМ» А. А. Цепляеву и сотруднику этого департамента Б. Любимову, сделавшим ценные замечания и дополнения, которые были учтены при подготовке статьи.

Список литературы

[1] *Бергстрем С., Робер Л.* Rational Unified Process — путь к успеху: руководство по внедрению RUP (пер. с англ.). М.: Кудиц-образ, 2004. 256 с.

[2] Информационные технологии. Процессы жизненного цикла программного обеспечения. ISO/IEC 12207. 1995.

[3] *Кознов Д. В.* Программная инженерия. Часть 1. Учебное пособие. СПб. Изд-во СПбГУ, 2005. 40 с.

[4] *Ольхович Л. Б., Кознов Д. В.* Метод автоматической валидации UML-спецификаций на основе OCL // Программирование. № 6. 2003. С. 44–50.

[5] *Окомин И.* План тестирования PMO. Внутренняя документация ЗАО «ЛАНИТ-ТЕРКОМ», департамент телекоммуникаций. 2003. 84 с.

[6] *Попова Т. Н., Тиунова А. Е., Кожарина Л. А.* Опыт использования визуального представления информации для контроля качества в процессе создания программного обеспечения // Системное прорамммирование. СПб.: 2004. С. 225–249.

[7] *Сомервилл И.* Инженерия программного обеспечения. Изд. 6-е (пер. с англ.). М.: Вильямс, 2002. 623 с.

[8] *Терехов А. Н., Эрлих Л. А., Терехов А. А.* История и архитектура проекта RescueWare // Автоматизированный реинжиниринг программ. СПб.: Изд-во СПбГУ, 2000. С. 7–19.

[9] Технологический процесс: логическая схема. ЗАО «ЛАНИТ-ТЕРКОМ». Отдел Software Reengineering. 2002.

[10] *Cusumano M. A., Selby R. W.* Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People. Free Press, 1995. 320 p.

[11] IEEE Recommended Practice for Software Requirements Specifications // SWE Standards Committee of the IEEE Computer Society Pub., 1993.

[12] *Kiyaev V., Sobolev I., Terekhov A. A., Fedotov B.* Formalization and Automation of Global Software Development Processes // Proc. 2nd International Workshop «New Models of Business. Managerial Aspects and Enabling Technolog». 2002. P. 150–160.

[13] *Kiyaev V. I., Terekhov A. A.* Software Process Improvement in Russian Company: A Case Study // Proc. 1st International Workshop «New Models of Business. Managerial Aspects and Enabling Technology». 2001. P. 122–130.

[14] *Paulk M. C., Weber C. V., Curtis B.* e.a. The Capability Maturity Model: Guidelines for Improving the Software Process. Addison-Wesley, 1995.

[15] Software Process Improvement Capabilities and Determination (SPICE). ISO/IEC TR 15504-CMM. 1998.

[16] *Terekhov A. A.* An Overview of the Russian IT Industry. Baltics IT T Review. N 3 (30). 2003. P. 22—26.

[17] *Terekhov A. A.* The Russian Software Industry // IEEE Software. Vol. 18. № 6. 2001. P. 98—101.

[18] *Zmud R. W.* An Examination of «Push-Pull» Theory Applied to Process Innovation in Knowledge Work // Management Science. Vol. 30(6). 1984. P. 727–738.