ПОЧТОВЫЙ СЕРВЕР TEGU

Описание жизненного цикла

Содержание

Введение

Почтовый сервер Tegu (далее ПО) предназначен для организации сервиса электронной почты (email).

Процессы обеспечения жизненного цикла ПО направлены на достижение следующих целей:

- улучшение качества ПО;
- устранение проблем, выявленных в ходе эксплуатации ПО;
- расширение функциональности ПО.

Процессы жизненного цикла ПО обеспечиваются участием команды разнопрофильных специалистов, включающей системных инженеров, программистов, инженеров по тестированию. Высокий уровень качества ПО достигается использованием проверенных методик, формализацией процессов разработки, тестирования и ввода в эксплуатацию элементов ПО, автоматизацией процессов поддержания жизненного цикла с использованием специализированного ПО.

Основными инструментами поддержания жизненного цикла являются:

- система управления проектом разработки ПО;
- система контроля версий;
- автоматизированная система сборки ПО;
- система организации тестирования.

Система управления проектом разработки ПО предназначена для организации совместной работы над проектом разработки ПО, управления задачами по добавлению новой функциональности и исправлению ошибок, мониторинга показателей развития проекта.

Система контроля версий предназначена для обеспечения эффективной совместной работы специалистов группы разработки ПО, версионирования промежуточных состояний и релизов, а также их публикации.

Автоматизированная система сборки ПО обеспечивает сборку компонентов ПО в заданном окружении по требованию или по расписанию, позволяет в любой момент иметь актуальные сборки ПО для поддерживаемых операционных систем.

Система организации тестирования включает различные виды тестов и обеспечивает своевременное выявление ошибок, возникающих в процессе разработки ПО.

1. Компоненты инфраструктуры процесса поддержки жизненного цикла программного обеспечения

1.1 Система для управления проектами, задачами и отслеживания ошибок. Организация рабочего процесса

В качестве системы для управления проектами, задачами и отслеживания ошибок используется серверное веб-приложение Redmine https://project.mbk-lab.ru.

Redmine представляет собой коллективную среду для управления проектами, формализации процессов выполнения задач, управления требованиями, организации взаимодействия пользователей и мониторинга показателей выполнения проектов. Redmine является гибкой и конфигурируемой системой, позволяет настроить рабочие процессы отдельно для каждого проекта.

Для обеспечения процессов жизненного цикла ПО в системе Redmine создан отдельный проект, в рамках которого определены шаблоны различных видов задач, описаны компоненты рабочего процесса разработки и тестирования ПО.

Новые задачи в проекте разработки ПО возникают при необходимости добавить новую функциональность или устранить ошибки, выявленные в процессе тестирования или эксплуатации. Работы по добавлению новой функциональности ведутся в соответствии с планом работ по развитию ПО (планом релизов). Задачи по исправлению ошибок создаются в случае выявления ошибок в ходе ручного или автоматизированного тестирования либо в процессе эксплуатации ПО.

В соответствии с планом релизов системные инженеры разрабатывают частное техническое задание на добавление новой функциональности, в которое включают список функциональных и сопутствующих им нефункциональных требований. В системе управления задачами в рамках проекта разработки ПО создается новая задача, к которой прикладывается частное техническое задание (список требований), необходимые нормативные документы (стандарты, регламенты и т.п.) и иные вспомогательные материалы.

Задачи по исправлению ошибок создаются инженерами по тестированию и специалистами по качеству программного обеспечения по результатам ручного или автоматизированного тестирования ПО BOS v.1.1.0, а также инженерами службы технической поддержки по заявкам пользователей. В описание таких задач включают сведения, необходимые для воспроизведения ошибки, к которым, как правило, относятся описание сетевой топологии, характеристики и конфигурации сетевых устройств, характеристики потоков сетевого трафика, протоколы работы устройств, перечень вводимых команд.

Помимо этого, к задаче по исправлению ошибок могут прилагаться ссылки на стандарты и регламенты, описывающие ожидаемое поведение. Вновь созданная задача получает статус "Новая" и назначается руководителю группы разработки ПО.

Руководитель группы разработки ПО анализирует описание задачи с точки зрения достаточности приведенной в нем информации для начала процесса внесения изменений в

исходные коды. При необходимости он обращается к системным инженерам, инженерам по качеству и тестированию ПО, инженерам технической поддержки с целью уточнения описания задачи.

После этого задаче назначается исполнитель из числа программистов-разработчиков и специалист по контролю качества из числа инженеров по тестированию. В случае необходимости привлечения для решения задачи сразу нескольких исполнителей, она разбивается на подзадачи. Деление, как правило, производится таким образом, чтобы каждая подзадача сводилась к логически изолированным изменениям в исходных кодах ПО, которые могли бы быть выполнены и протестированы вне связи с другими подзадачами. Такой подход обеспечивает возможность параллельной работы с подзадачами группы из нескольких исполнителей. Если же выполнение какой-либо задачи (подзадачи) блокирует выполнение других задач, то это фиксируется с использованием соответствующих инструментов Redmine.

Получив уведомление о назначении задачи, исполнитель проверяет достаточность информации и доступность ресурсов. В положительном случае меняет статус задачи на "Принята".

В момент фактической работы над задачей исполнитель меняет ее статус на "В работе".

Разработчик руководствуется правилами работы с системой контроля версий, которые приведены в разделе 1.2 Организация работы с системой контроля версий. Для работы над задачей разработчик создает отдельную ветку (branch) в репозитории исходных кодов и в процессе решения задачи работает только на этой ветке. Такой подход позволяет временно изолировать изменения (commits), связанные с решением данной конкретной задачи, от других изменений в исходных кодах.

В случае, если работа над задачей невозможна в данный момент времени по таким причинам, как, например, изменение в плане работы (плане релизов), наличие более приоритетных задач или наличие неудовлетворенных зависимостей, то разработчик по согласованию с руководителем группы разработки ПО и руководителем проекта переводит задачу в статус "Отложена" с указанием конкретных причин принятия данного решения. Из данного состояния задача может быть в любой требуемый момент возвращена в работу.

В случае, если в процессе работы над задачей выявлена ее неактуальность или невозможность ее решения в силу каких-либо технических причин, программист-разработчик по согласованию с руководителем разработки и руководителем проекта переводит задачу в статус "Отклонена" с указанием конкретных причин принятия данного решения. При необходимости это решение может быть пересмотрено в будущем.

Завершив внесение изменений в исходные тексты ПО, разработчик инициирует создание тестовой сборки ПО в автоматизированной системе сборки (см. раздел 1.3 Автоматизированная система сборки).

Автоматизированная система сборки осуществляет сборку ПО в специально подготовленном окружении под заданную аппаратную платформу и версию ОС. Выполнив

сборку, разработчик изменяет статус задачи на "Тестовая сборка". Тем самым задача передается на исполнение инженеру-тестировщику.

На тестовой сборке выполняется ручное тестирование функциональности, заявленной в описании задачи, если задача связана с добавлением новой функциональности, или проверяется возможность воспроизведения ошибки, если задача направлена на устранение ошибок.

На тестовой сборке также выполняется автоматизированное поверхностное (smoke) тестирование с целью подтверждения корректности работы базовых функций ПО. В процессе проведения тестирования специалист по качеству (тестированию) ПО в основном руководствуется спецификациями, приведенными в описании задачи, но может также проводить дополнительные тесты, например, для того, чтобы оценить влияние изменений, внесенных в исходные тексты, на смежный функционал.

В процессе тестирования воспроизводятся типовые и граничные условия и оценивается корректность работы ПО с точки зрения решения поставленной задачи. В процессе проведения тестирования могут быть задействованы целевые аппаратные платформы, виртуальные машины, референсные сетевые устройства, генераторы трафика и иные средства, определяемые методикой тестирования.

Для проведения тестирования сборка ПО, полученная при помощи автоматической системы сборки, разворачивается на целевой аппаратной платформе или виртуальной машине, после чего тестовое устройство (или виртуальная машина) включается в топологию, определяемую спецификацией задачи и методикой тестирования, выполняются необходимые тесты.

В случае, если тесты выявляют некорректную работу и ошибки функционирования ПО, или ошибка, которая должна быть устранена в ходе решения задачи, снова воспроизводится на тестовой сборке, задача возвращается в разработку со статусом "Проблемы на тестовой сборке". Инженер по тестированию добавляет в описание задачи условия и технологию воспроизведения проблемы, выявленной в ходе тестирования. Разработчик анализирует выявленную проблему и вносит в исходные тексты ПО изменения, необходимые для ее устранения, после чего повторно переводит задачу в статус "Тестовая сборка". Процедуры тестирования и исправления ошибок повторяются до тех пор, пока все проблемы, возникающие на данном этапе работы, не будут устранены.

Если очередная проверка тестовой сборки не выявляет каких-либо проблем, задача возвращается разработчику со статусом "Тестовая сборка проверена".

В процессе работы над задачей и верификации тестовой сборки на основной рабочей ветке в репозитории исходных кодов могут накапливаться изменения. Это может влиять на результат решения задачи. Например, на общей ветке могут иметь место изменения (commits), конфликтующие с изменениями, выполненными в ходе решения задачи. В связи с этим, после верификации тестовой сборки разработчик инициирует процедуру получения объединенной сборки (Merged build). Для этого разработчик выполняет слияние изменений

из основной рабочей ветки в ветку, где он ведет решение текущей задачи, предварительно актуализировав состояние общей рабочей ветки в своей копии репозитория.

Разработчик при необходимости разрешает конфликты слияния и запускает сборку ПО в автоматизированной системе сборки. По завершении процесса сборки разработчик прикладывает к задаче ссылку на объединенную сборку и меняет статус задачи на "Объединенная сборка".

На объединенной сборке проводится ручное тестирование в соответствии со спецификацией задачи и регрессионное тестирование. В рамках ручного тестирования в зависимости от задачи проверяется правильность работы вновь введённой функциональности или результат исправления ошибок. В основном методика верификации объеденной сборки на данном этапе совпадает с ранее рассмотренной методикой верификации тестовой сборки. Ручное тестирование направлено на проверку корректности решения поставленной задачи в рамках объединенной сборки. В свою очередь регрессионное тестирование позволяет выявить возможные проблемы, когда изменения в исходных кодах, выполненные в рамках решения задачи, влияют на стороннюю функциональность. В случае выявления проблем в процессе тестирования инженер-тестировщик фиксирует их характер и описывает методику воспроизведения, после чего возвращает задачу на доработку со статусом "Проблемы на объединенной сборке".

Разработчик анализирует выявленные проблемы, устраняет их причины, готовит новый (исправленный) вариант объединенной сборки и отправляет его на повторное тестирование. Процесс повторяется до тех пор, пока все проблемы на объединенной сборке не будут устранены.

Если тестирование объединенной сборки не выявляет никаких проблем, задача возвращается разработчику со статусом "Объединенная сборка проверена". Разработчик готовит и отправляет в главный репозиторий исходных кодов проекта запрос на принятие изменений (Pull request). При этом задача переводится в статус "Рецензирование".

Перед внесением изменений в главный репозиторий проекта выполняется проверка качества исходных кодов (Code Review). Проверку качества обычно осуществляет один из наиболее опытных разработчиков или непосредственно сам руководитель группы разработки ПО. В ходе проверки прежде всего контролируется:

- соответствие внесенных изменений принятым стандартам оформления;
- целесообразность примененных архитектурных решений;
- наличие потенциальных источников уязвимостей;
- достаточность, актуальность и информативность комментариев в коде.

При наличии замечаний рецензент фиксирует их в комментариях к задаче и саму задачу возвращает разработчику со статусом "Необходимы изменения". Разработчик устраняет причины замечаний и отправляет задачу на повторную проверку. Процесс повторяется, пока все замечания не будут устранены. После этого задача переводится в статус "Решена".

1.2 Организация работы с системой контроля версий

В качестве системы контроля версий (git) используется система Gitea https://git.mbk-lab.ru. Под управлением системы контроля версий находятся все компоненты ПО, включая сторонние компоненты. Используется схема, при которой все основные модули программного обеспечения хранятся в едином репозитории исходных кодов. Исключение составляют сторонние компоненты, код которых хранится в отдельных репозиториях. В основном репозитории ПО хранятся также основные модули, исходный код программы, сборки для различных поддерживаемых операционных систем.

Работа с системой контроля версий организована с использованием нескольких веток (branches). Основными ветками являются ветки «Master» («Мастер-ветка») и «Development» («Разработка»). На ветке «Master» хранятся только релизные версии ПО. На ветке «Development» хранятся самые последние наработки, прошедшие верификацию с использованием функционального и регрессионного тестирования.

Ветка «Development» является основной рабочей веткой. Кроме того, в репозитории хранятся исходные коды системы сборки.

Каждый разработчик имеет свою собственную копию (fork) основного репозитория, которая автоматически или по требованию синхронизируется с основным репозиторием. Таким образом, работая со своей копией, разработчик всегда имеет доступ к актуальным изменениям в исходных кодах, зафиксированным в системе контроля версий. В ходе работ по внесению изменений в исходные коды каждый разработчик работает со своей копией репозитория, а по окончании задачи и ее верификации отправляет свои наработки в главный репозиторий.

Для решения конкретной задачи, связанной с введением новой функциональности или исправлением ошибок, в соответствии с методикой организации рабочего процесса, изложенной в разделе 1.1 Система для управления проектами, задачами и отслеживания ошибок. Организация рабочего процесса, разработчик синхронизирует локальную копию репозитория на своей рабочей машине с сервером Git, после чего выделяет отдельную ветку для решения задачи. Новая ветка «Feature» выделяется из стабильной ветки «Development».

В процессе работы над задачей разработчик фиксирует изменения в исходном коде (commits) на выделенной ветке. По окончании внесения изменений в исходные коды разработчик делает тестовую сборку (test build) из ветки «Feature» и передает ее на тестирование. В процессе верификации тестовой сборки проверяется соответствие добавленной функциональности требованиям и спецификациям, приведенным в задании на разработку, или проверяется возможность воспроизведения ошибки, если задача связана с исправлением ошибок функционирования ПО.

Если в процессе верификации тестовой сборки выявляются проблемы, разработчик анализирует причины и устраняет их путем внесения изменений в исходные коды и фиксации изменений (commits) на ветке «Feature». После устранения проблем разработчик повторно создает тестовую сборку, и процесс верификации проводится снова.

Когда все проблемы на тестовой сборке устранены, разработчик проводит слияние изменений, которые накапливаются на ветке «Development», в ветку «Feature» и создает объединенную сборку. Объединенная сборка подвергается функциональному и регрессионному тестированию. В случае выявления проблем на этом этапе разработчик устраняет их путем внесения изменений (commits) на ветке «Feature», повторно готовит объединенную сборку и передает ее на тесты. Данный процесс повторяется до тех пор, пока все проблемы, выявляемые на объединенной сборке, не будут устранены.

По завершении тестирования объединенной сборки разработчик отправляет запрос на внесение изменений из ветки «Feature» в ветку «Development» в главном репозитории. Предлагаемые изменения подвергаются рецензированию (Code review), в ходе которого оцениваются правильность оформления кода, целесообразность архитектурных решений, наличие потенциальных уязвимостей.

При наличии замечаний разработчик анализирует причины, устраняет их и фиксирует исправления (commits) на ветке «Feature». Внесение исправлений приводит к автоматическому обновлению запроса на внесение изменений. Выполняется повторное рецензирование и при необходимости вносятся дополнительные доработки.

Если рецензирование не выявляет проблем, изменения фиксируются на ветке «Development» основного репозитория. Перед выпуском очередного релиза от ветки «Development» создается ответвление - ветка «Release Candidate». На ней в автоматизированном режиме проводятся полное функциональное тестирование, регрессионное тестирование, нагрузочное тестирование и другие виды долгосрочного тестирования.

При выявлении ошибок в ходе тестирования они исправляются, изменения в коде фиксируются на ветке «Release Candidate». В ветку «Release Candidate» попадают только те изменения, которые связаны с устранением ошибок, и не попадают изменения, связанные с добавлением новой функциональности. После окончания тестирования и устранения всех ошибок ветка «Release candidate» сливается с ветками «Master» и «Development». Текущее состояние ветки «Master» помечается тегом (tag), соответствующим номеру и наименованию релиза.

В дальнейшем данный тег может использоваться для идентификации нужной сборки при организации работы технической поддержки.

1.3 Автоматизированная система сборки

Система сборки ПО является внутренней разработкой и основывается на стандартных инструментах автоматизации сборки программного обеспечения в среде операционной системы Linux.

Для обеспечения стабильной среды и автоматизации процесса сборки организован пул сборочных серверов для различных операционных систем и аппаратных архитектур.

Система сборки клонирует на сборочный сервер указанную версию, хранящуюся в репозитории, после чего производит сборку и создает бинарные пакеты, которые размещаются в репозитории бинарных пакетов и могут использоваться для обновления программного обеспечения пользователей.

Сборочные сервера интегрированы с системой контроля версий и настроены таким образом, чтобы сборка запускалась каждый раз при добавлении изменений в ветки «Development» и/или «Master» в основном репозитории исходных кодов в полуавтоматическом режиме.

Это позволяет в любой момент времени иметь актуальные рабочие и релизные сборки. Обычно процесс принудительной сборки запускают разработчики для получения тестовых (test build) и объединённых (merged build) сборок в соответствии со схемой организации рабочего процесса, изложенной в разделе 1.1 Система для управления проектами, задачами и отслеживания ошибок. Организация рабочего процесса.

1.4 Система автоматизированного тестирования

Система автоматизированного тестирования ПО является собственной разработкой, выполненной на базе программного обеспечения с открытым исходным кодом Python.

Система автоматизированного тестирования - это программное обеспечение, посредством которого осуществляется создание, отладка, выполнение и анализ результатов прогона тест-скриптов. Test Scripts — это наборы инструкций для автоматической проверки определенной части программного обеспечения.

Тестирование программных систем состоит из динамической верификации поведения программ на конечном наборе тестов. При этом тесты выбираются из обычно выполняемых действий прикладной области и обеспечивают проверку соответствия ожидаемому поведению системы.

Система автоматизированного тестирования позволяет производить:

- Тестирование производительности. Нагрузочное, стрессоустойчивое, тестирование на стабильность.
- Регрессионное тестирование. Выполняет проверку ПО на корректность функциональности, выпущенной и протестированной в предыдущей версии.
- Конфигурационное тестирование – выполнение одних и тех же тестов ПО на различных операционных системах.

Обычно после добавления очередной функциональности и выполнения тестов в ручном виде, этот тест автоматизируется и впоследствии используется в формате автоматического тестирования в виде скрипта.

2. Процессы устранения неисправностей и совершенствования программного обеспечения

2.1 Исправление ошибок

Процесс исправления ошибок начинается в одном из следующих случаев:

- обнаружена ошибка в процессе тестирования;
- поступило обращение пользователя в службу технической поддержки.

В случае обнаружения ошибки на этапе тестирования, если тестирование проводилось в рамках работы над уже существующей задачей, информация об ошибке и условиях ее воспроизведения указывается в комментариях к этой задаче. В остальных случаях в системе управления проектами создается новая задача с описанием ошибки и условий ее воспроизведения.

Если ошибка возникла в процессе работы над существующей задачей, разработчик сразу приступает к ее устранению. В случае новой задачи, в зависимости от критичности выявленной ошибки, определяется, когда задача должна быть принята в работу.

С обращениями пользователя работает специалист отдела технической поддержки (2-я линия). Он воспроизводит проблему по исходным данным, полученным от пользователя. При необходимости отправляет ему запрос на уточнение данных или осуществляет удаленное подключение к клиентскому устройству для сбора необходимой информации.

Далее описываются шаги по воспроизведению ошибки и задача назначается разработчикам в отдел разработки ПО. Проблема анализируется разработчиками, после чего они вносят необходимые изменения в исходные коды ПО и совместно со специалистами отдела тестирования проводят верификацию изменений в соответствии с методикой организации работы.

2.2 Выпуск релиза

Выпуск основных релизов ПО, содержащих новую функциональность, осуществляется один раз в полгода. В промежутке между основными релизами могут при необходимости выпускаться промежуточные релизы, которые, как правило, включают исправления наиболее критичных ошибок функционирования ПО.

Перед выпуском релиза из ветки «Development» основного репозитория исходных кодов выделяется ветка «Release Candidate» (см. раздел 1.2 Организация работы с системой контроля версий). Из ветки «Release Candidate» создается сборка ПО, которая подвергается разным видам тестирования.

При подготовке релиза выполняют:

- полное функциональное (регрессионное) тестирование с использованием виртуальных машин и целевых аппаратных платформ;
- нагрузочное тестирование;
- тестирование на стабильность;

- тестирование на совместимость, где проверяется корректность работы в среде различных операционных систем;
- тестирование установки.

В ручных тестах инженер проверяет корректность установки ПО на целевую платформу в соответствии с инструкцией по установке.

При наличии ошибок, выявленных в процессе тестирования, в исходные коды вносятся исправления, которые фиксируются (commits) на ветке «Release Candidate». После успешного завершения тестирования и отладки выполняется слияние изменений в ветку «Master» и производится обновление репозитория бинарных пакетов для доступа пользователей.

2.3 Процесс поставки

Процесс поставки определяет, как пользователи получат обновление ПО.

Возможны два основных варианта получения новых версий ПО:

- в виде бинарных пакетов;
- в виде исходных кодов.

Установка ПО подробно описана в статье Описание функциональных характеристик, инструкция по установке и эксплуатации

Обновление бинарных пакетов на стороне пользователя происходит в полуавтоматическом режиме: достаточно подключить репозиторий и выполнить получение обновлений.

Если же пользователь не нашел бинарные файлы для выбранной операционной системы, то может воспользоваться исходным кодом, который необходимо скомпилировать согласно инструкции в текущей операционной системе.

3. Техническая поддержка

Техническая поддержка предусматривает работу с обращениями пользователей, а также выпуск обновлений ПО (включая исправления критических ошибок).

Право на получение технической поддержки может быть приобретено при покупке сертификата на техническую поддержку.

Заказчик может получить следующие виды технической поддержки:

- базовая техническая поддержка;
- расширенная техническая поддержка (Заказчик должен приобрести Сертификат на техническую поддержку);
- выполнение дополнительных работ, выходящих за рамки технической поддержки, возможно по отдельному, договору с Заказчиком (необходимо согласование ТЗ).

Виды работ по Базовой техподдержке:

1. Консультирование Заказчика в части технологий почтовой системы.
2. Предоставление рецептов HowTo по отдельным сценариям использования.

Виды работ по Расширенной техподдержке:

1. Консультирование Заказчика в части технологий почтовой системы.
2. Предоставление рецептов HowTo по отдельным сценариям использования.
3. Анализ существующей программно-аппаратной инфраструктуры Заказчика с целью определения оптимальной стратегии внедрения ПО.
4. Подбор аппаратного комплекта для реализации системы электронной почты.
5. Практическая помощь в установке, настройке и обслуживании.
6. Практическая помощь в миграции на новое ПО.
7. Практическая помощь в оптимизации работы ПО.
8. Мониторинг 24x7 состояния инфраструктуры Заказчика.
9. Практические мероприятия по восстановлению функциональности ПО после сбоев.
10. Обновление ПО (по требованию).

4. Персонал, необходимый для обеспечения поддержки

Сотрудники организации (программисты, тестировщики, специалисты по сопровождению) обладают необходимым набором знаний для работы со всеми компонентами, входящими в состав ПО, и инструментами для работы с ними.

5. Оборудование

Разработчик располагает всем необходимым собственным оборудованием, на базе которого построены все инструменты и хранилища. Все серверы организации находятся на территории Российской Федерации.