ДОРИС-МАРШРУТ

ОПИСАНИЕ ПРОЦЕССОВ, ОБЕСПЕЧИВАЮЩИХ ПОДДЕРЖАНИЕ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПОДСИСТЕМЫ «ДОРИС- МАРШРУТ».

Подсистема «ДОРИС – Маршрут», предназначена для обеспечения управления расписаниями и маршрутами общественного транспорта городской агломерации. Подсистема «ДОРИС – Маршрут» является частью единого комплекса специализированного программного обеспечения и аппаратных средств (ДОРИС), предназначенного для автоматизации деятельности городских структур по управлению функционированием транспортной инфраструктурой городской агломерации.

1 Общие сведения о жизненном цикле программного обеспечения

Жизненный цикл программного обеспечения (ПО) - период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации.

К числу основных процессов жизненного цикла ПО относятся:

1) процесс заказа;
2) процесс поставки;
3) процесс разработки;
4) процесс эксплуатации;
5) процесс сопровождения.

Ответственность за выполнение работ и задач в основном процессе несет организация, создающая и реализующая данный процесс. Данная организация гарантирует реальность существования и функциональные особенности конкретного процесса.

В ГОСТ Р ИСО/МЭК 12207-2010 (Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств) определены основные группы различных видов деятельности и соответствующие им процессы, которые могут выполняться в течение жизненного цикла программных систем.

К числу основных процессов поддержки жизненного цикла ПО относятся процессы поддержки программных средств:

a) Процесс менеджмента документации программных средств цель, которого заключается в разработке и сопровождении зарегистрированной информации по программным средствам. Документация разрабатывается и делается доступной в соответствии с определенными стандартами;

b) Процесс менеджмента конфигурации программных средств цель, которого заключается в установлении и сопровождении целостности программных составных частей проекта и обеспечении их доступности для заинтересованных сторон.

c) Процесс обеспечения гарантии качества программных средств цель, которого заключается в предоставлении гарантии соответствия рабочей продукции и процессов предварительно определенным условиям и планам.

d) Процесс верификации программных средств цель, которого заключается в подтверждении того, что каждые программный рабочий продукт и (или) услуга процесса или проекта должным образом отражают заданные требования.

e) Процесс валидации программных средств цель, которого заключается в подтверждении того, что требования выполняются для конкретного применения рабочего программного продукта.

Примечание - Для валидации помимо тестирования могут использоваться другие средства, такие как анализ, моделирование, имитация и т.п.

f) Процесс ревизии программных средств цель, которого заключается в поддержке общего понимания с правообладателями прогресса относительно целей соглашения и того, что именно необходимо сделать для помощи в обеспечении разработки продукта, удовлетворяющего правообладателей. Ревизии программных средств применяются как на

уровне менеджмента проекта, так и на техническом уровне и проводятся в течение всей жизни проекта.

g) Процесс аудита программных средств цель, которого заключается в независимом определении соответствия выбранных продуктов и процессов требованиям, планам и соглашениям.

h) Процесс решения проблем в программных средствах цель, которого заключается в обеспечении гарантии того, что все выявленные проблемы идентифицируются, анализируются, контролируются и подвергаются менеджменту для осуществления их решения.

Не все выше перечисленные работы могут быть предусмотрены проектом.

Для разрабатываемого на предприятии ПО в качестве модели жизненного цикла для большинства проектов выбирается спиральная модель, соответствующая масштабу и сложности проекта.

Спиральная модель схожа с инкрементной моделью, но с в ней уделяется больше внимания оценки и разрешения рисков. Спиральная модель подразделяет реализацию проекта на четыре этапа: планирование проекта, оценка рисков, проектирование и разработка, проведение оценки. Проект каждый раз заново проходит через эти четыре стадии при создании новой версии или фрагмента ПО (что соответствует одному витку спирали в данной модели). Базовый виток спирали, начинающийся на этапе постановки задач, включает в себя определение требований и оценку рисков. Каждый последующий виток строится на основе базового.

Процессы, операции и задачи выполняемые в ходе жизненного цикла ПО соответствуют требованиям ГОСТ 34.601-90.

2 Общие сведения о жизненном цикле программного обеспечения подсистемы «ДОРИС - Маршрут»

2.1 Вышеперечисленные процессы поддержки жизненного цикла ПО подсистемы «ДОРИС - Маршрут» описаны в эксплуатационных документах подсистемы «Инструкция по установке и эксплуатации. Подсистема «ДОРИС - Маршрут»» (кроме процессов f) и g), которые не предусмотрены условиями договора).

2.2 На этапе планирования проекта определяется общая концепция разрабатываемого программного продукта, и на ее основе строится базовая структура проекта, оценивается его выполнимость и связанные с ним риски, описываются соответствующие подходы к конфигурационному управлению и технологиям.

Наиболее важная часть плана проекта – декомпозиция системы на совокупность составных частей в соответствии с требованиями высокого уровня системной иерархии.

Все требования к компонентам ПО, устанавливаемые на этапе определения требований, следуют из одного или нескольких требований высокого уровня.

В процессе определения требований в качестве исходных данных используются цели, поставленные в разделе плана проекта, описывающем требования высокого уровня.

Каждый программный компонент должен иметь собственную Спецификацию требований.

Требования к ПО определяют функционал программного компонента, производительность, точность, временные характеристики работы, затраты ресурсов используемого оборудования, работоспособность в нестандартных условиях и при перегрузках. В Спецификации требований к программному обеспечению описываются алгоритмы и математические методы.

2.3 Оценка рисков проекта

Риск – это любое событие, которое может помешать реализации проекта в соответствии с планом или его успешному завершению. Риски можно идентифицировать из разных источников. Некоторые из них могут быть довольно очевидными и будут выявлены до начала проекта. Другие будут идентифицированы в течение жизненного цикла проекта, и риск может быть идентифицирован любым участником проектом. Некоторые риски будут

присущи самому проекту, в то время как другие будут результатом внешних воздействий, которые полностью неподконтрольны команде проекта.

2.4 Процесс проектирования и разработки программного обеспечения подсистемы «ДОРИС - Маршрут» включает в себя следующие основные этапы:

− формирование функциональных требований;

− проектирование и формирование спецификации требований к компонентам системы;

− разработка (доработка) ПО в соответствии со сформированными спецификациями требований;

− тестирование функционирования, разработанного (доработанного ПО) на стенде;

− устранение замечаний по итогам тестирования (в случае если они были выявлены);

− корректировка документации (если доработка влечет изменение документированных функций);

− установка актуальной версии программного обеспечения.

Требования к программному продукту и к особенностям разработки программного обеспечения определяются на этапе постановки задач. На этапе оценки и разрешения рисков применяется специальный процесс для определения рисков и нахождения разных решений по их разрешению.

На данном этапе определяется архитектурный проект системы, в соответствии с которым выполняется идентификация элементов ПО и удовлетворяются заданные требования. При определении верхнего уровня архитектуры системы должны быть идентифицированы составные части технических средств, программных средств и ручных операций. Должно быть учтено что все системные требования распределяются между этими составными частями. Составные части конфигурации технических средств, программных средств и ручных операций должны последовательно идентифицироваться этими составными частями.

На этапе разработки ПО в качестве исходных данных используются элементы проектирования, описанные в принятом плане разработки. По каждому элементу определяется артефакт или набор артефактов ПО. Артефакты ПО включают в себя (но не ограничиваются ими) меню, диалоговые окна, формы для управления данными, форматы отчетных данных и специализированные процедуры, и функции.

Результатами этапа проектирования и разработки являются план конфигурационного управления, план реализации качества ПО, план реализации проекта и календарный план, содержащий подробный список запланированных работ грядущего этапа определения требований, а также предварительная оценка трудозатрат на последующих этапах.

Третий этап включает в себя непосредственно разработку ПО и его тестирование по окончании данного этапа.

2.5 Четвертый этап анализа и оценка результатов позволяет разработчику и заказчику оценить результат проекта на текущий момент, прежде чем начнется новый виток разработки.

2.6 Система управления версиями исходного кода

В процессе разработки ПО в качестве распределенной системы управления версиями исходного кода используется инструмент Git.

Git – это программное обеспечение, свободно распространяемое на условиях универсальной общедоступной лицензии GNU версии 2.

Каждый рабочий каталог в Git – это полноценный репозиторий, содержащий всю историю проекта с возможностью отслеживания версий, не зависящий от доступа к сети или центральному серверу.

Концепция Git возникла на основе опыта, полученного при управлении большим распределенным проектом разработки при работе над Linux, а также особенностей работы с файловыми системами, и острой потребности в создании жизнеспособной системы в кратчайшие сроки. Все это привело к следующим особенностям реализации:
Git позволяет быстро создавать и осуществлять слияние отдельных ветвей проекта и включает в себя специальные инструменты для визуализации и навигации по нелинейной истории разработки. Основным принципом в Git является предположение, что слияние изменения будет производиться чаще, чем его написание, так как оно распределяется для оценки несколькими разработчиками. Ветви в Git занимают очень мало места: они являются лишь ссылками на отдельные коммиты. С помощью родительского коммита может быть построена полная структура ветвей.
Git предоставляет каждому разработчику локальную копию всей истории разработки, и изменения копируются из одного такого репозитория в другой. Такие изменения импортируются в виде дополнительных ветвей разработки, и их слияние может осуществляться так же, как и для ветви, разработанной локально.

Репозитории могут быть опубликованы с помощью HTTP, FTP, rsync или протокола Git через обычный сокет, ssh или HTTP. Git также имеет эмулятор сервера CVS (системы одновременных версий), который позволяет использовать существующие клиенты CVS и плагины IDE для доступа к репозиториям Git. Репозитории подверсий и SVK можно использовать напрямую с помощью git-svn.
История хранится в Git таким образом, что идентификатор конкретной версии («коммит» в терминологии Git) зависит от полной истории разработки, предшествовавшей данному коммиту. После его публикации невозможно изменить старые версии незаметно. Эта структура похожа на дерево хешей, но с наличием дополнительных данных в узлах и листовых вершинах.

ПО разрабатывается несколькими специалистами. Когда разработчик начинает внедрение нового функционала или отладку, он забирает последнюю версию проекта из системы Git. После выполнения задачи новая версия ПО хранится на его локальном хосте. Прежде чем залить новую версию в одну из рабочих ветвей, менеджер версий ПО отправляет измененное ПО на проверку двум другим разработчикам, выбранным произвольно. После одобрения кода ПО отправляется на слияние. Менеджер версий ПО пытается совместить новые изменения с обновленными ветвями проекта. При неудачном слиянии новый код отправляется разработчикам ПО, чтобы они одобрили общее обновление. При успешном слиянии новая ветвь отправляется на тестирование. Тестировщики проверяют обновленное ПО разными методами, и при успешном прохождении тестов ветвь разработки подготавливается к релизу. Автоматическая компиляция версии релиза проекта осуществляется дважды в день. Также ПО проверяется в тестовой среде. Если все тесты и компиляция проходят успешно, версия считается готовой к релизу. В противном случае менеджер версий ПО находит проблему и возвращает ветвь разработчикам для исправления.

2.7 В процессе разработки и модернизации было задействовано 7 ведущих инженеров-программистов.

2.8 Процессы поддержки программного обеспечения подсистемы.
Цель процесса поддержки и решения проблем в Программе заключается в обеспечении гарантии качества оказанных услуг по контракту (или договору) и того, что все выявленные запросы идентифицируются, анализируются, контролируются для осуществления их решения.
В процессе поддержки и решения проблем в Программе:

a) проблемы регистрируются, идентифицируются и классифицируются в систему управления запросами (далее -- СУЗ);

b) запросы анализируются и оцениваются для определения приемлемого решения (решений);

c) выполняется решение запросов;

d) запросы отслеживаются вплоть до их закрытия;

e) известно текущее состояние всех зафиксированных запросов;

f) предоставляются регулярные версии Программы (в случае оказания услуг по сопровождению);

g) проводятся регламентные работы.

2.9.1 Режим работы службы поддержки - с 9:00 до 18:00 MSK.

2.9.2 В процессе сопровождения, гарантийного обслуживания и технической поддержки задействовано 2 инженера.

2.10 Совершенствование и модернизация программы происходит путем доработки интерфейса и внедрения новых функций. Увеличивается производительность программы, и минимизируется время на взаимодействия между органами управления интерфейса всего комплекса и пользователем. Фактическое улучшение всей программы отображается в виде изменения младших разрядов номера версии ПО с учетом доработок.

2.11 В процессе тестирования и эксплуатации программного обеспечения могут возникнуть сообщения о неисправности. В случае их возникновения необходимо осуществить процедуру передачи информации о характере ошибки в авторизованную сервисную компании ООО «НПО «ИТС СОФТ». Устранение неисправностей и техническое обслуживание может осуществлять только квалифицированный персонал, а именно сотрудники авторизованной сервисной службы компании ООО «НПО «ИТС СОФТ».

Для оформления заявки на устранения неисправности необходимо отправить заявку на электронную почту info@npo-its.ru или почтой по адресу: 119334, Россия, г. Москва, 5-й Донской проезд, д. 15, стр. 2, этаж 2, помещ./ком. 2/IV/31а

3 Термины и сокращения

Жизненный цикл программного обеспечения (ПО) - период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации. (ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств);

Модель жизненного цикла ПО - структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла. Модель жизненного цикла зависит от специфики, масштаба и сложности проекта и специфики условий, в которых система создается и функционирует (ГОСТ Р ИСО/МЭК 12207-2010);

Процесс (process) - Совокупность взаимосвязанных или взаимодействующих видов деятельности, преобразующих входы в выходы (ГОСТ Р ИСО/МЭК 12207-2010);

Проект (project) - Попытка действий с определенными начальными и конечными сроками, предпринимаемая для создания продукта или услуги в соответствии с заданными ресурсами и требованиями (ГОСТ Р ИСО/МЭК 12207-2010);

Версия (version) - Идентифицированный экземпляр составной части (ГОСТ Р ИСО/МЭК 12207-2010);

Менеджмент – управление процессом;

«ДОРИС - Маршрут» – подсистема предназначена для автоматизации управления расписаниями и маршрутами общественного транспорта.

ПО – программное обеспечение.