СИСТЕМА ЭЛЕКТРОННОГО ДОКУМЕНТООБОРОТА

СХЕМА ДИСТРИБЬЮЦИИ И ПРОИЗВОДСТВЕННЫЙ ЦИКЛ

Листов 9

2016

СОДЕРЖАНИЕ

1. ТЕРМИНЫ, СОКРАЩЕНИЯ И ОПРЕДЕЛЕНИЯ

В настоящем документе применены следующие сокращения и термины с соответствующими определениями:

Термин/сокращение	Определения
Antrunner	Утилита, которая используется как часть жизненного цикла сборки СЭДО, работающая на базе утилиты Apache Ant.
Apache Maven	Maven является инструментом автоматизация сборки, используемый в основном для Java - проектов. Maven рассматриваются два аспекта программного обеспечения: во- первых, он описывает , как строится программное обеспечение, а во- вторых, он описывает его зависимости.
Development Change (DC)	Базовое понятие процедуры сборки системы СЭДО - это последовательность изменений, вносимых произвольным образом в систему, объединенных одной целью
Git	Распределенная система контроля версий
Java Development Kit	Бесплатно распространяемый компанией Oracle Corporation комплект разработчика приложений на языке Java, включающий в себя компилятор Java (javac), стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему Java (JRE). В состав JDK не входит интегрированная среда разработки на Java, поэтому разработчик, использующий только JDK, вынужден использовать внешний текстовый редактор и компилировать свои программы, используя утилиты командной строки
Wasabi	Система инкрементной заливки изменений, используемая при сборке проекта СЭДО
Репозиторий артефактов	Хранилище библиотек, которые используются при сборке исходного кода
СЭДО\Система	Система электронного документооборота

2. ВВЕДЕНИЕ

2.1. Цель

Данный документ содержит описание схемы дистрибуции и производственного цикла СЭДО.

2.2. Краткое описание возможностей СЭДО

Система электронного документооборта является многофункциональным программно-техническим комплексом, обеспечивающим автоматизацию управления деловыми процессами в условиях распределенного использования информации.

СЭДО обеспечивает комплексную автоматизацию служб документационного управления в части обработки управленческой документации, а именно – процессов создания, согласования, утверждения, регистрации, хранения и движения управленческих документов, а также контроля исполнения резолюций и поручений.

В настоящее время СЭДО позволяет обрабатывать следующие виды документов:

Документы, участвующие в Федеративном взаимодействии:

- организационно-распорядительные документы;
- входящие документы;
- исходящие документы.

В следующих версиях Системы перечень видов обрабатываемых документов может быть расширен.

3. СХЕМА ДИСТРИБУЦИИ

3.1 Общие положения

Основной схемой распространения обновлений СЭДО является распространение в виде исходного кода.

Исходный код СЭДО хранится в репозитории исходного кода под управлением системы контроля версий Git.

Для полноценного обновления СЭДО на сервере сборки необходимо иметь:

1. Java Development Kit v1.6.x;
2. Apache Maven v3.2.2;
3. Git 2.7.2 и выше;
4. доступ к репозиторию исходного кода ООО “АйДи-Технологии Управления” (или его копию);
5. доступ к репозиторию артефактов ООО “АйДи-Технологии Управления” (или его копию).

При обновлении СЭДО используются также ряд инструментов, разработанных в ООО “АйДи-Технологии Управления”, описание которых дано ниже.

3.2 Базовые понятия жизненного цикла обновления СЭДО

3.2.1. Создание DC

Если есть необходимость в создании DC, нужно:

- сгенерировать уникальный номер DC;
- зарегистрировать DC в виде тэга в скрипте build.post.xml, лежит в корне репозитория с исходным кодом СЭДО, указав имя dc, его описание и последовательность действий в виде вложенных тэгов, поддерживаемых утилитой Antrunner.

При сборке системы, Antrunner будет искать в хранилище dc с заданным именем и если он не создан - выполнять вложенные тэги.

В настоящее время модель эволюции СЭДО представляет собой следующие шаги:

- dc "init" - первое dc системы, создается при установке системы скриптом build.init.xml;
- прогон pre-dc - заливка dc, описанных в скрипте build.pre.xml;
- прогон wasabi - внесение изменений в данные, которые могут быть описаны декларативным способом, поддерживаемым Wasabi;
- прогон post-dc - заливка dc, описанных в скрипте build.post.xml;

3.2.2. Цели создания Wasabi

1. Автоматизировать процесс внесения изменений в модель данных СЭДО (уменьшить время обновления системы, устранить человеческий фактор).
2. Получить в декларативной форме описание схемы данных СЭДО.
3. Получить инструмент анализа степени отличия произвольной площадки от целевой модели.
4. Уменьшить издержки при миграции старых версий СЭДО.
5. Получить возможность быстрой установки СЭДО "с нуля".
6. Уменьшить издержки при выкладке обновлений системы
7. Свести к минимуму отличия в схемах данных между различными площадками

3.2.3. Использование Wasabi

Для работы с Wasabi необходимо использовать утилиту wasabi-tool, входящую в состав утилит, собираемых из исходного кода СЭДО.

Утилита wasabi-tool оперирует понятиями источника и приемника данных. Работу утилиты схематично можно изобразить так:

[источник] -> [метаданные] -> [приемник]

Базовый синтаксис запуска утилиты выглядит так:

java -jar wasabi-tool-1.0.jar -s [источник] -t [приемник]

3.2.4. Сбор метаданных

Для формирования источника данных утилиты wasabi-tool разработан плагин инфраструктуры Apache Maven – wasabi-plugin, который сканирует артефакты СЭДО на предмет извлечения из них информации, необходимой Wasabi для формирования источника данных.

3.2.5. Antrunner

Antrunner был разработан для решения двух задач сборки СЭДО:

- предоставить инструмент для внесения изменений в тех случаях, когда Wasabi использовать невозможно;
- автоматизировать создание Development Changes при выкладке, чтобы снизить влияние человеческого фактора на результат сборки.

Специфической особенностью Wasabi является то, что она в первую очередь является системой анализа и сравнения данных,и лишь во вторую – средством сборки.

Данные описаны в декларативном формате, то есть Wasabi работает по принципу - нужно описать целевую схему -> и Wasabi внести изменения в текущую так, чтобы достигнуть целевой. Эта особенность делает непригодным использование Wasabi для ряда случаев, и Antrunner решает задачу внесения изменений, для случаев, когда более пригодна императивная модель внесения изменений.

С глобальной точки зрения, модель внесения изменений, используемая в СЭДО является именно императивной и состоит из следующих шагов:

- прогон скрипта build.init.xml (создание dc "init");
- прогон скрипта build.pre.xml;
- прогон Wasabi;
- прогон скрипта build.post.xml.

Скрипты build.pre.xml и build.post.xml прогоняются утилитой antrunner, и являются обычными скриптами в нотации Apache Ant, за исключением нескольких специфических тэгов, разработанных для СЭДО.

3.2.6. Основные тэги Antrunner

- GroovyScript - прогоняет groovy-скрипт;
- SqlScript - прогоняет sql-скрпт (используя execsql);
- Api - выполняет api-команду;
- DC – тэг-контейнер, который содержит в себе другие тэги, принадлежащие к этому DC. При успешном выполнении создает в базе объект типа ddt_dc, и повторно не прогоняется.

3.2.7. Параметры запуска

- -f - путь до скрипта$
- -l - путь до папки с внешними библиотеками, необходимыми для работы тэгов или скриптов/

3.3 Шаги обновления

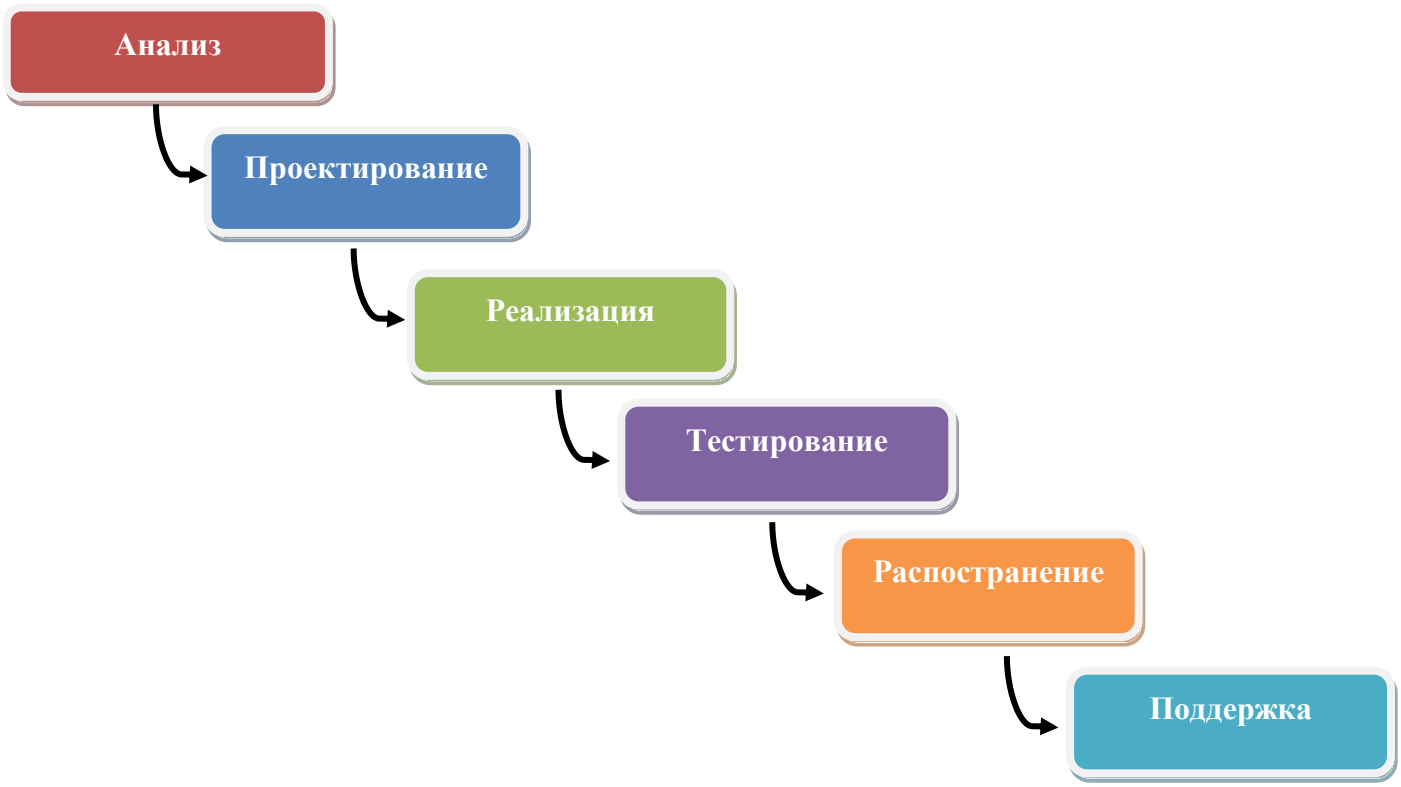
Сборка логически состоит из следующих шагов:

1. Актуализация исходного из репозитория исходного кода средствами Git.
2. Сборка и инсталляция артефактов системы в локальный репозиторий Maven.
3. Сбор метаданных Wasabi.
4. Прогон build.pre.xml.
5. Заливка метаданных Wasabi.
6. Прогон build.post.xml.
7. Деплой артефактов и их зависимостей в модули Системы.
8. Очистка кэшей Системы.
9. Полная перезагрузка Системы.

4. ПРОИЗВОДСТВЕННЫЙ ЦИКЛ

4.1 Общие сведения

Процесс разработки программного продукта сотсоит из следующих этапов:



4.2 Анализ требований к продукту

Самая важная задача при создании программного продукта — это выработка требований, или анализ требований к продукту. Заказчик чаще всего представляет весьма размытую идею о том, каким должен быть конечный результат, и не имеет представления о том, как должна работать программа. Незаконченные, нелепые, а иногда противоречащие друг другу требования распознаются хорошими инженерами на этой стадии. Частая демонстрация живого кода может уменьшить риск того, что начальные требования были не верны. Один из методов нахождения проблем такого рода — это анализ элементов программного обеспечения.

Когда общие требования получены от клиента, их необходимо уточнить и отобразить в документе. Реализованная функциональность может отличаться от определённой, в результате высокой стоимости разработки и/или непонятных требованиях к продукту. Если разработка проводится вне фирмы заказчика, то данный документ может использоваться для разрешения споров связанных с функциональностью продукта.

Анализ области работы часто является первой ступенью лестницы проектирования нового фрагмента программного обеспечения, вне зависимости от того, является ли он добавлением к уже существующему приложению или новым приложением, подсистемой или совершенно новой системой. Принимая, что разработчики не являются в начале достаточно образованными в области знаний нового программного обеспечения, первая задача — это собственно исследование этой самой области знаний. Чем лучше разработчики знают область, в которой работают, тем меньше потом возникает работы. Также её проводят для того, чтобы позже не появлялось путаницы в терминологии, и пониманием того, что делает

эта программа. Если аналитик использует неверную терминологию, то опять же возможны недопонимания, в результате того, что программа будет делать не то, что нужно.

4.3 Проектирование, реализация и тестирование

Проектирование — процесс создания общей архитектуры и алгоритмов согласно спецификациям (проектным доументам).

Реализация — это та часть процесса, во время которой программисты создают программный код продукта.

Тестирование — часть процесса, позволяющая выявить и решить различные ошибки.

В процессе выполнения этих этапов выполняется документирование. Документирование проводится для того, чтобы в будущем было проще поддерживать и улучшать программный продукт.

4.4 Распространение и поддержка

Распространение начинается после того, как выполнено полное тестирование продукта и он признан готовым к релизу.

Техническая поддержка и обучение важны, так как большой процент проектов проваливается потому, что многие разработчики не понимают, что сколько бы не было потрачено времени на программный продукт, он будет бессмысленным, если его никто не использует.

Поддержка и улучшение продукта вместе с исправлением найденных ошибок может занимать больше времени, чем собственно процесс разработки этого продукта.