Министерство образования и науки Украины
Харьковский национальный университет имени В.Н. Каразина
факультет математики и информатики
кафедра теоретической и прикладной информатики

Реферат
по предмету “Управление разработкой программных проектов“
на тему:

Управление конфигурациями ПО. Повторное использование ПО

Подготовили: Орленко Родион, Швед Ирина
Проверила: Владимирова Марина Владимировна

Харьков 2018

Содержание:

Введение

История развития дисциплины управления конфигурацией

Первым заметным шагом в развитии управления конфигурациями было изобретение микрометра в 1636 году (William Gascoigne). Это устройство сыграло важную роль в индустриальной революции и переходе к массовому производству. Этот инструмент позволил использовать взаимозаменяемые части в различных устройствах, что являлось существенной причиной для того, чтобы использовать процедуры управления конфигурацией.

Первые инженерные концепции, которые привели к становлению дисциплины управления конфигурацией, начали формироваться в начале 20-го века и обрели реальную форму в 60-х годах прошлого века.

Изначально создатели концепции управления конфигурацией преследовали цель улучшения способов разработки и сопровождения программных средств (ПС). «Отцы-основатели» управления конфигурацией хотели создать дисциплину, которая обеспечивала бы соответствие разработанного ПС потребностям пользователей, для которых это ПС разрабатывалось. Они изучили успешные проекты и обобщили опыт применения тех технологий, которые хорошо себя проявили. Другой важной целью было обеспечить простоту модификации и сопровождения ПС и (так как «отцы-основатели» в основном работали на правительственные учреждения) возможность для заказчика ПС сменить разработчика без того, чтобы заново проходить весь цикл разработки ПС с нуля.

Кроме того, в качестве дополнительной цели рассматривалось обеспечение оценки состояния проекта на основе отчетности по ключевым показателям. Они сосредоточились на достижении долгосрочных целей и не рассчитывали получить сразу очевидные преимущества от использования разрабатываемых ими технологий. Следует заметить, что преимущества такого сорта трудно выразить количественно, так как при успешном использовании управления конфигурацией в организации просто перестают растрачивать ресурсы на лишнюю работу. Например, на повторное исправление ошибки, которая уже была исправлена ранее, но появилась вновь из-за того, что при сборке ПС правильный код случайно заменили на неправильный.

«Отцы-основатели» осознали, что в первую очередь им требуется контролировать то, какие части входят в готовый продукт (под продуктом может пониматься как ПС, так и оборудование и, в широком смысле, любое изделие, состоящее из различных частей) и каким образом они взаимосвязаны, а также отслеживать изменения в отдельных частях продукта и в их взаимосвязях друг с другом. Они выбрали слово «конфигурация» для обозначения «относительного взаиморасположения частей». Слово «управление» вполне подходило по смыслу, и в итоге получилось «управление конфигурацией».

Определение. Цели. Задачи

Зачем нужно управление конфигурациями? В крупном проекте большие объемы информации меняются очень быстро и неконтролируемые изменения могут быстро ввергнуть проект в хаос. Работая над программным проектом, группа программистов, тестеров и менеджеров сталкивается с проблемой отслеживания версий программ, внесения в них изменений. Чем больше проект, тем больше

времени разработчики тратят на согласования изменений в исходных текстах и получения работающих версий программного продукта. Данная задача может усугубиться наличием в компании регионально удаленных групп-разработчиков. С этими проблемами помогает справится управление конфигурациями ПО.

Объектом управления конфигурациями является трансформирующиеся в процессе жизненного цикла физические и функциональные характеристики продукта, состав и версии его компонент (конфигурация), а **целью** управления конфигурациями – поддержание целостности и прослеживаемости его (продукта) конфигурации в ходе таких трансформаций. Под целостностью конфигурации продукта в контексте рассматриваемой дисциплины следует понимать однозначную воспроизводимость, а также физическую и функциональную совместимость всех его компонент.

Управление конфигурацией ПО — это один из вспомогательных процессов, поддерживающих основные процессы ЖЦ ПО, прежде всего процессы разработки и сопровождения ПО.

Под конфигурацией ПО понимается совокупность его функциональных и физических характеристик, установленных в технической документации и реализованных в ПО.

Объект конфигурации (Configuration Item, CI): исходные тексты, скомпилированные программы, исходные коды программ, документация, элементы аппаратуры, процедуры и материалы обучения и т.п. — базовое понятие процесса управления конфигурациями.

Задача процесса управления конфигурациями — предотвратить неконтролируемое развитие проекта, гарантируя, что все изменения учитываются и санкционируются в соответствии с принятой технологией разработки.

При разработке программных систем создается множество взаимосвязанных объектов: требований, исходных текстов, объектных файлов, описаний тестов и т. п. — согласованные совокупности которых принято называть конфигурациями, а процесс поддержки их изменений и целостности в течение жизненного цикла проекта — управлением конфигурациями. Основная цель введения в проект процесса управления конфигурациями — предотвратить неконтролируемое развитие проекта и дать гарантии того, что все вносимые изменения будут учитываться и санкционироваться. Управление конфигурациями включает в себя процедуры по идентификации элементов проекта, по управлению изменениями и поддержке трассируемости объектов, а также деятельность по поддержке аудитов состояния и контролю статуса конфигурации.

Планирование управления конфигурациями

План управления конфигурациями - документ, где излагается концепция процесса и имплементация средств автоматизации. В нем же расписываются все роли, и, что особенно важно, деятельности в зависимости от стадии жизненного цикла разработки ПО.

Менеджер Управления Конфигурациями отвечает за написание плана УК. Этот человек знает процесс разработки. Понимает цели и задачи УК. Все свои знания он излагает в плане УК. Сам управляет процессом УК.

План – живой документ, то есть находится в постоянных изменениях, читается и корректируется. Он должен храниться на видном месте, его должны все читать, так как план описывает процесс разработки, то его особенно должны читать вновь

пришедшие разработчики, тестировщики, менеджеры. Чтобы план был живым его необходимо читать и корректировать. Такая ошибка, как неправильное понимание процесса, ведет к простоям и частым доработкам продукта. План должен быть доступен и управляем в части его изменений.

Цели процесса менеджмента конфигурации

Цель процесса менеджмента конфигурации состоит в установлении и поддержании целостности всех идентифицированных выходных результатов проекта или процесса обеспечения доступа к ним любой заинтересованной стороны.

Процесс Управления конфигурацией

Процесс управления конфигурациями отнесен к группе интегральных процессов, необходимых для обеспечения качества выполнения процессов разработки и их выходных данных. Интегральные процессы выполняются одновременно с процессами разработки и обеспечивают непрерывную поддержку разработки.

Основные цели процесса управления конфигурациями согласно ГОСТ 51904 состоят в том, чтобы обеспечить:

- определяемую и управляемую конфигурацию ПО на протяжении всего жизненного цикла;
- целостность при тиражировании исполняемого объектного кода для производства ПО или, в случае необходимости, его повторной генерации для проведения исследований или модификации;
- управление входными и выходными данными процесса в течение жизненного цикла, что гарантирует непротиворечивость и повторяемость работ в процессах;
- контрольную точку для проверки, оценки состояния и контроля изменений посредством управления элементами конфигурации и определения базовой линии;
- контроль над тем, чтобы дефектам и ошибкам было уделено внимание, а изменения были зарегистрированы, утверждены и реализованы;
- оценку соответствия программного средства требованиям;
- надежное физическое архивирование, восстановление и сопровождение элементов конфигурации.
- возможность всегда вернуться к предыдущей версии;

В результате успешного осуществления процесса менеджмента конфигурации:

a) определяется стратегия менеджмента конфигурации;

b) определяются составные части, нуждающиеся в менеджменте конфигурации;

c) устанавливается базовая линия конфигураци;

d) осуществляется управление изменениями в составных частях, находящихся под менеджментом конфигурации;

e) осуществляется управление конфигурацией составных частей, входящих в выпуск;

f) статус составных частей, на которые распространяется менеджмент конфигурации, становится доступным на протяжении всего жизненного цикла.

Технологии

Для управления конфигурацией определения системы сейчас используют информационные системы:

- VCS (version control system) — система управления версиями в программной инженерии;
- PDM-система (product data management) — система хранения информации проекта-design;
- PLM-система (product life cycle management) — система управления жизненным циклом, это PDM + система управления изменениями и поддержка интерфейсов в другие информационные системы других стадий жизненного цикла — системы закупок, например;
- EAM-система (enterprise asset management) — система управления активами, используется для учёта установленного оборудования на стадии эксплуатации.

По ISO/IEC 12207 процесс управления конфигурацией является процессом применения административных и технических процедур на всем протяжении жизненного цикла программных средств для:

- обозначения, определения и установления состояния (базовой линии) программных объектов в системе;
- управления изменениями и выпуском объектов; описания и сообщения о состояниях объектов и заявок на внесение изменений в них;
- обеспечения полноты, совместимости и правильности объектов;
- управления хранением, обращением и поставкой объектов.

Данный процесс состоит из следующих работ:

- подготовка процесса;
- определение конфигурации;
- контроль конфигурации;
- учет состояний конфигурации;
- оценка конфигурации;
- управление выпуском и поставка.

Основные понятия управления конфигураций

Управление конфигурацией включает в себя следующие понятия:

- **базис** (configuration baseline) — исходная (утвержденная) конфигурация. Базис определяется на следующих этапах:

Базис	**Определяется на этапе**	**Тип спецификации**	**Характеристики**	**Описываемый элемент**
Функциональный (Functional)	Выбор концепции	A	Функциональные спецификации	Система
Физический (Allocated)	Техническое проектирование	B	Проектная документация	Элемент конфигурации
Продукции (Product)	Техническое проектирование	C, D, E	Производственно-технологическая документация	Элемент конфигурации

- **версия/ревизия** (version/revision);
- **элемент конфигурации** (configuration item, CI) — элемент системы, который является основой для описания и формального управления проектированием системы, базовая часть системы, которая проектируется, конструируется и создается силами одной организации. Характеристики и интерфейсы CI с другими составными частями должны быть определены и контролироваться, чтобы гарантировать надлежащее функционирование CI в составе системы в целом. Различают:
    - аппаратные элементы конфигурации (hardware CI — HWCI)
    - элементы конфигурации программного обеспечения компьютера (computer software CI — CSCI)
- **управление интерфейсами** - управление взаимодействием составных частей системы друг с другом и с окружением системы, функция системной инженерии.
- **управление изменениями** - процесс запроса, определения возможности, планирования, реализации и оценки внесения изменения в систему. Главные задачи управления изменениями — их трассировка и контроль влияния на связанные сущности.

Условно можно выделить 3 основных уровня конфигурации:

- конфигурация;
- актив(ПК, ноутбуки и другая техника, которая находится на финансовом контроле, но не требует управления);

- материально-технический ресурс(различные мелкие периферийные устройства, такие как клавиатуры, мышки, флешки и тому подобное).

Практики

Управление конфигурацией, в том числе и управление изменениями невозможно рассматривать отдельно от управления данными (управления информацией, интеграции данных жизненного цикла и т.д.). Сюда попадают следующие практики:

- практика выпуска (release) инженерных артефактов (например, выпуск чертежей) — можно обсуждать, является ли hand-over данных входящим в эту практику, или должен рассматриваться отдельно
- практика выпуска заказных спецификаций (BOM, bill of materials)
- практика запросов на изменения
- практика изменения проекта
- практика управления данными (чтобы нужные заинтересованным сторонам данные оказывались у них в нужное время. Да, "управление требованиями", как часть инженерии требований, отвечающая именно за то, чтобы требования адекватно хранились и адекватно предоставлялись по запросам заинтересованных сторон — это часть именно этой практики. Ибо нет никаких особенностей именно у "управления требованиями" в отличии их от управления любыми другими данными).

Дисциплина управления конфигурацией имеет следующие основные основные практики:

- **Идентификация** — поддержка инженерных разбиений (классификаций, кодировок) и именования/кодировки отдельных конфигурационных единиц (configuration items).
- **Конфигурационный учет/регистрация** — административное обеспечение взаимного соответствия:
    - проекта (включая требования),
    - исполнительной документации (as built, "что мы думаем о реальной системе"),
    - самой системы "в железе и бетоне".

  Обычно обеспечивается наличием конфигурационной базы данных (CMDB — configuration management database) и административными процедурами по ее ведению (в т.ч. по назначению ведущего учёт (регистратора), передаче ведения учёта от регистратора регистратору, делегированию полномочий по учёту в порядке распределенной учётной деятельности и т.д.)

- **Контроль версий** (version/revision control):

  обеспечение того, что базис (утвержденная для каких-то целей конфигурация) собирается из взаимно соответствующих версий частей системы (будь то версии проектной или исполнительной документации, или же версии самой системы "в железе и бетоне"). Софтверщикам с их CVS и SVN против git и Mercurial должно быть понятно, о чем это.

Базовые концепции и элементы

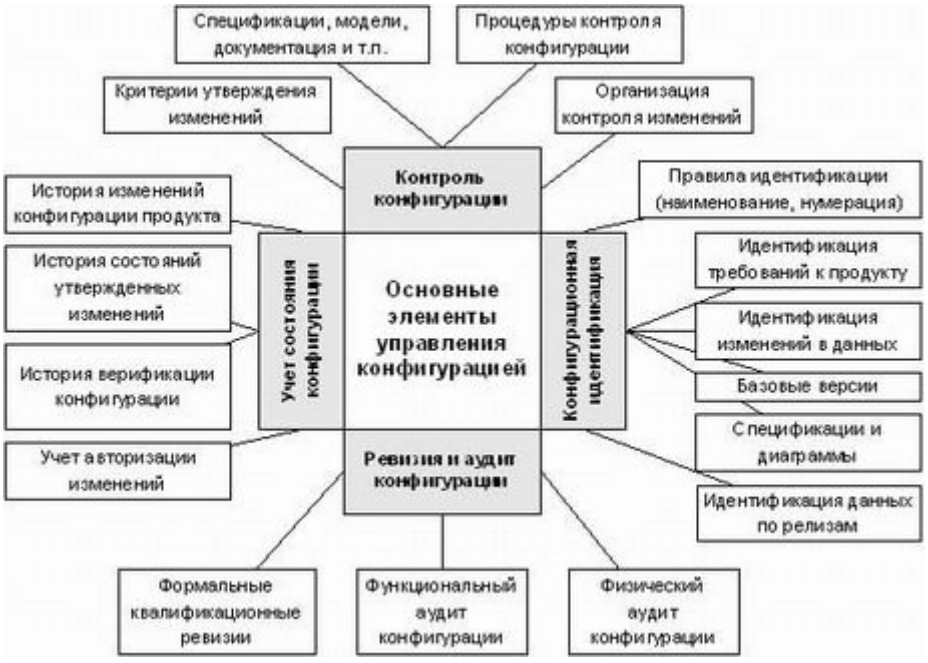
Основатели дисциплины управления конфигурацией собрали вместе, упорядочили и дали названия всем этим техникам, используемым при разработке, так что в итоге получилась отдельная дисциплина – управление конфигурацией.

Во время формирования дисциплины управления конфигурацией в ней были воплощены следующие важные концепции:

- Документы создаются для описания продукта и являются средством управления конфигурацией продукта.
- Изменения в продукте контролируются посредством контроля изменений в документации.
- Изменения в продукте не производятся до тех пор, пока они не сделаны в документации.
- До того, как быть реализованными в документации и продукте, изменения должны быть формально утверждены.
- Все изменения должны отслеживаться.
- Конфигурационные объекты (продукты), документы и их версии нумеруются и именуются единообразно и недвусмысленно (или уникально).
- Ведется отчетность о состоянии изменений, документов и продуктов.
- Каждый документ периодически сравнивается с соответствующим ему документом верхнего уровня на предмет выявления несоответствий.
- Продукт в целом сравнивается со своим описанием (конфигурационной идентификацией) и должен этому описанию соответствовать.

Используя введенную выше терминологию управления конфигурацией, эти концепции были сгруппированы в следующие четыре элемента управления конфигурацией:

1. Конфигурационная идентификация
2. Контроль конфигурации
3. Учет состояния конфигурации
4. Ревизия и аудит конфигурации



Конфигурационная идентификация

Идентификация конфигурации ПО проводится путем выбора элемента конфигурации ПО и документирования его функциональных и физических характеристик, а также оформления технической документация на элементы конфигурации ПО.

Конфигурационная идентификация основывается на следующих составляющих:

- правила идентификации и нумерации – что и каким образом идентифицируется;
- идентификация требований к продукту – каким образом идентифицируются требования к ПС;
- идентификация изменений в данных – каким образом идентифицируются изменения в данных;
- базовые версии – создаются для фиксации стабильных состояний системы и используются как кандидаты на релиз ПС;
- спецификации и диаграммы – документы, описывающие конфигурационную спецификацию ПС и диаграммы, используемые для этих же целей;
- идентификация данных по релизам – методы, позволяющие однозначно сопоставить элементы конфигурации ПС и их версии с определенным релизом ПС.

Контроль конфигурации

Контроль конфигурации ПО состоит в проведении работ по координации, утверждению или отбрасыванию реализованных изменений в элементы конфигурации после формальной ее идентификации, а также оценке результатов. Ревизия конфигурации — процесс проверки соответствия документа нижнего уровня всем требованиям верхнего.

Контроль конфигурации включает:

- критерии утверждения изменений – определяют формальные критерии, на основании которых принимается решение об утверждении или отклонении предложенного изменения;
- спецификации, модели, документация и т.п. – все эти элементы конфигурации подвержены изменениям и находятся в сфере действия контроля конфигурации;
- процедуры контроля конфигурации – утвержденные процедуры, которым должны следовать участники проекта;
- организация контроля изменений – организационная составляющая процесса, определяющая ответственность участников проекта при выполнении процедур контроля конфигурации.

Учет статуса конфигурации

Учет статуса конфигурации ПО проводится в виде комплекса мероприятий для определения уровня изменений в конфигурацию, аудита конфигурации в виде комплекса мероприятий по проверке правильности внесения изменений в конфигурацию ПО. Информация и количественные показатели накапливается в соответствующей БД и используются при управлении конфигурацией, составлении отчетности, оценке качества и выполнении других процессов ЖЦ. Учет состояния конфигурации — процесс подготовки отчетов о текущем состоянии продукта и состоянии утверждённых изменений.

Учет состояния конфигурации предполагает:

- ведение истории изменений конфигурации продукта – определяет кто, когда и какие изменения делал;
- ведение истории состояний утвержденных изменений – показывает, как менялись состояния утвержденных изменений от момента утверждения и до момента завершения их отработки;
- ведение истории верификации конфигурации – хранит данные о всех проведенных верификациях и их результаты;
- учет авторизации изменений – указывает на то, кто отвечает за сделанные изменения.

Ревизия и аудит конфигурации

Аудит конфигурации – это деятельность, которая выполняется для оценки продукта и процессов на соответствие стандартам, инструкциям, планам и процедурам. Аудит определяет степень удовлетворения элемента конфигурации заданным функциональным и физическим характеристикам системы. Иными словами, аудит конфигурации — процесс проверки соответствия готового продукта или его части документации.Ревизия конфигурации — процесс проверки соответствия документа нижнего уровня всем требованиям верхнего.

Ревизия и аудит конфигурации включает:

- формальные квалификационные ревизии – определяют соответствие элементов конфигурации предъявляемым к ним формальным требованиям, например, соответствие определенному шаблону документа;
- функциональный аудит конфигурации – определяет соответствие конфигурации ПС функциональным требованиям, предъявляемым к продукту;
- физический аудит конфигурации – определяет наличие или отсутствие отдельных элементов в составе конфигурации.

Менеджмент документации активов

Цель процесса менеджмента документации программных средств заключается в разработке и сопровождении зарегистрированной информации по программным средствам, созданной некоторым процессом.

В результате успешного осуществления процесса менеджмента документации программных средств:

a) разрабатывается стратегия идентификации документации, которая реализуется в течение жизненного цикла программного продукта или услуги;

b) определяются стандарты, которые применяются при разработке программной документации;

c) определяется документация, которая производится процессом или проектом;

d) указываются, рассматриваются и утверждаются содержание и цели всей документации;

e) документация разрабатывается и делается доступной в соответствии с определенными стандартами;

f) документация сопровождается в соответствии с определенными критериями.

Инженерия повторного использования компонентов

Во введении в стандарт IEEE Std. 1517-99 “IEEE Standard for Information Technology – Software Lifecycle Process – Reuse Processes” даётся следующее понимание повторному использованию в программном обеспечении: “Реализация повторного использования программного обеспечения подразумевает и влечёт за собой нечто большее, чем просто создание и использование библиотек активов. Оно требует формализации практики повторного использования на основе интеграции процессов и деятельности по повторному использованию в сам жизненный цикл программного обеспечения.”

Инженерия повторного использования компонентов (ПИК) - это систематическая и целенаправленная деятельность по подбору реализованных программных артефактов и представленных в виде ПИК, анализу их функций для добавления в качестве готовых в проектируемую систему и их интеграция с другими компонентами.

Согласно стандарту ISO/IEC-12207 эта деятельность классифицируется как организационная и планируемая инженерная деятельность, которая заключается в выявлении общих и специфических черт компонентов для принятия решений об их использовании в разработке новых ПС .

При этом предполагается, что имеется каталог, с помощью которого можно понять, какие ПИК, как готовые детали, имеются и как их можно соединить в программную конструкцию. Именно эта сторона характеризует повторное использование как систематическую и целенаправленную деятельность по созданию и использованию каталога ПИК.

Исследования и разработки в области инженерии программирования в направлении повторного использования компонентов (ПИК), готовых для применения в других областях человеческой деятельности привели к тому, что сформировалось два направления применения готовых ПИК :

-прикладная инженерия (application engineering) - процесс производства конкретных новых приложений из ПИК (модулей, программ, подпрограмм и др.), ранее созданных самостоятельно либо в среде программной системы или как отдельные элементы многоразового использования в инженерии другой ПрО;

-инженерия ПрО (domain engineering) включает методы разработки, поиска, классификации, адаптации, сбора ПИК и создания из них или из готовых частей систем семейства домена, которые сохраняют наработанный опыт по реализации одного домена для применения его в другом крупном домене. Необходимое условие этой инженерии - системные инструментальные средства поддержки методов накопления ПИК и внедрения их в новые подсистемы семейства или самого домена.

Первое направление фактически характеризует создание одиночных ПС из разного рода ПИК, а второе ставит задачу создания программных систем домена и их совокупностей с выделением отдельных частей ПС при проектировании, обладающими общими свойствами и характеристиками и способными к многоразовому использованию в других доменах этой совокупности.

Систематическое повторное использование - это капиталоемкий подход, который предусматривает наличие двух процессов в ЖЦ разработки ПС.

Первый процесс - это создание ПИК путем:

- изучения спектра решаемых задач ПрО, выявление среди них общих свойств и функций;
- построения компонентов, реализующих выявленные функции в виде ПИК;
- разработка каталога для хранения изготовленных компонентов и организации
- поиск необходимых компонентов по запросам пользователей.

Для успешной реализации данного процесса необходимо иметь определенный опыт в решении нескольких подобных между собой задач, позволяющий определить заложенные общие черты и различия, чтобы найти решение для их реализации, а также разработать приемы настройки на характерные для каждой задачи особенности.
Второй процесс - конструирование новых систем из готовых компонентов путем:

-понимания сущности новой системы (домена), определения целей ее создания и предъявляемых к ней требований;

-поиска в каталоге готовых компонентов, которые кажутся подходящими для их использования в новой системе;

-сопоставления цели новой разработки с возможностями найденных ПИК и принятия решений о целесообразности и месте их применения в системе;

-интеграция ПИК в новую разработку с обеспечением интерфейса с подсистемами и другими компонентами.

Первый процесс требует вложения капитала, второй - получение прибыли за счет экономии трудозатрат от применения готовых ПИК. Инвестиции в повторное использование требуют оценки эффективности вложения капитала, прогнозирования сроков и объемов возврата этого вложения, оценки рисков и др. Бизнес повторного использования, как любой бизнес, требует специальных условий по менеджменту всей инженерной деятельности инженерии систем из ПИК. Критерии успеха такого бизнеса определяются следующими предпосылками:

1. повторное использование готовых компонентов требует меньших трудозатрат, чем разработка их как новых разовых продуктов;
2. поиск пригодных для использования компонентов требует меньше усилий, чем произвести реализацию необходимых функций для целей проектируемой системы;
3. настройка компонентов на новые условия среды применения должна обеспечиваться меньшими трудозатратами, чем новая разработка.

Основная парадигма ПИК - "писать - один раз, выполнять - много раз, где угодно". Архитектура, в которую встраивается готовый ПИК, поддерживает стандартные механизмы для работы с компонентами как со строительными блоками. Чтобы обеспечить высокий уровень использования ПИК, они должны обладать такими основными свойствами: функциональность, удобство использования и качество реализации.

Разновидности ПИК. В качестве ПИК могут использоваться формализованные артефакты деятельности разработчиков ПС, которые отражают некоторую функциональность для применения в новых разработках. Под *артефактом* понимается реальная порция информации, которая может создаваться, изменяться и использоваться при выполнении деятельности, связанной с разработкой ПС различного назначения.
Артефактами могут быть:

- промежуточные продукты процесса разработки ПС (требования, постановки задач, архитектура и др.);

- описания результатов процесса разработки ПС (спецификация, модели, каркас и т.п.)

- готовые компоненты ПС или отдельные части системы;

- продукции, фреймы, диаграммы, паттерны и т.п.

К компонентам ПИК выдвигаются такие требования, как независимость от конкретной платформы, наличие стандартного интерфейса и параметров настройки на новую среду, возможность их взаимодействия в системе без внесения в них изменений.

Разработке ПС с помощью ПИК соответствует модель ЖЦ со следующими общими этапами:

-анализ объектов и отношений реализуемой ПрО для выявления ПИК,

-обладающих общими свойствами, присущими группам объектов этой области;

-адаптация имеющихся в базе репозитория ПИК, разработка новых

-функциональных компонентов, не представленных в этой базе и доведение их до уровня ПИК;

-разработка интерфейсов компонентов и их размещение в репозитарии интерфейсов системы;

-интеграция ПИК и их интерфейсов с другими элементами создаваемой системы и формирование конфигурации этой системы.

Повторные компоненты могут быть прикладными и общесистемными.

Прикладные компоненты выполняют отдельные задачи и функции прикладной области деятельности домена (бизнес-домены, коммерция, экономика и т.п.), которые могут использоваться в дальнейшем, как готовые в качестве прикладных систем в других доменах с аналогичными функциями.

К *общесистемным компонентам* относятся компоненты общего и универсальные назначения, а также общесистемные сервисные средства, которые обеспечивают системное обслуживание и предоставляют разные виды сервисов для многих создаваемых программных систем разного назначения.

К компонентам общего назначения относятся: трансляторы, редакторы тестов, системы генерации, интеграции, загрузчики и др. Они используются всеми прикладными системами в процессе их проектирования и выполнения. Универсальные системные компоненты обеспечивают функционирование любых (в том числе и прикладных) компонентов, обмен данными и передачу сообщений между всеми видами систем и компонентов, расположенных в разных средах и платформах компьютеров. К ним относятся ОС, СУБД, сетевое обеспечение, электронная почта и др.

Связь между прикладными и общесистемными средствами осуществляется через стандартные интерфейсы, обеспечивающие взаимодействие разных типов компонентов через механизмы передачи данных и сообщений.

COTS компоненты (commercial off – the – shelf**)**

Применение готовых программных компонентов, именуемых на международном рынке commercial off – the – shelf или COTS —готовые программные компоненты, в которых реализованы различные методы и алгоритмы, предназначенные для решения одной и той же задачи в составе программной системы.

COTS требует учета совместимости компонентов. Программная система реализуется согласно схеме блоков восстановления с согласованием, что позволяет обеспечить высокий уровень надежности.

Одним из актуальных на сегодняшний день подходов к созданию надежного программного обеспечения является применение блоков восстановления. Программное обеспечение, разрабатываемое согласно данному подходу, отличается применением избыточных программных компонент, предназначенных для решения одной и той же задачи. В контексте программного обеспечения схемы блоков восстановления данные программные компоненты называются альтернативами. Фактически блоки восстановления комбинируют основные идеи контрольных точек и рестарта для компонент программного обеспечения таким образом, что различные альтернативы используются только после того, как обнаруживается ошибка.

Контрольные точки определяются во время проектирования и тестирования программного обеспечения и служат для того, чтобы восстановить состояние ПО, если в альтернативе произойдет сбой или она вернет неверный результат. Оценка результата производится посредством приемочного теста. В том случае, если результат работы альтернативы оказался неверным, происходит откат и выполнение другой альтернативы.

На международном рынке программных продуктов это программные компоненты получили название COTS – компонентов (от англ. Commercial off – the – shelf – « коробочный программный продукт»). Реализация в программных компонентах различных методов и алгоритмов, предназначенных для решения одной и той же задачи, позволяет избежать возникновения идентичных ошибок в этих компонентах и тем самым обеспечить отказоустойчивое исполнение всего программного обеспечения.

Компоненты могут быть представлены в виде библиотек, часть из них может быть разработана самостоятельно, а часть – различными поставщиками. Однако в некоторых случаях программные компоненты могут быть несовместимы из-за проблем интерфейса, спецификации или реализации. В связи с этим возникает необходимость учета совместимости различных программных компонент. Применение блока восстановления с согласованием позволяет достичь высокой надежности программной системы. Использование программной избыточности согласно данному подходу гарантирует высокий уровень надежности, но требует дополнительных ресурсов.

Преимущества повторного использования ПО

Преимущество	Описание
Повышение надежности	Компоненты, повторно используемые в других системах, оказываются значительно надежнее новых компонентов. Они протестированы и проверены в разных условиях работы. Ошибки, допущенные при их проектировании и реализации, обнаружены и устранены еще при первом их применении. Поэтому повторное использование компонентов сокращает общее количество ошибок в системе

Уменьшение проектных рисков	Для уже существующих компонентов можно более точно прогнозировать расходы, связанные с их повторным использованием, чем расходы, необходимые на их разработку. Такой прогноз – важный фактор администрирования проекта, так как позволяет уменьшить неточности при предварительной оценке сметы проекта
Эффективное использование специалистов	Часть специалистов, выполняющих одинаковую работу в разных проектах, может заниматься разработкой компонентов для их дальнейшего повторного использования, эффективно применяя накопленные ранее знания
Соблюдение стандартов	Некоторые стандарты, такие как стандарты интерфейса пользователя, можно реализовать в виде набора стандартных компонентов. Например, можно разработать повторно используемые компоненты для реализации различных меню пользовательского интерфейса. Все приложения предоставляют меню пользователям в одном формате. Использование стандартного пользовательского интерфейса повышает надежность систем, так как, работая со знакомым интерфейсом, пользователи совершают меньше ошибок
Ускорение разработки	Часто для успешного продвижения системы на рынке необходимо как можно более раннее ее появление, причем независимо от полной стоимости ее создания. Повторное использование компонентов ускоряет создание систем, так как сокращается время на их разработку и тестирование
Сокращение объема работ по сопровождению ПО	Если кто-то разработал ПО, то он же отвечает и за его последующее развитие. Известен **парадокс компетентного разработчика ПО**: "чем больше вы работаете, тем больше работы вы себе создаете". Довольные пользователи вашей продукции начнут просить добавления новых функциональных возможностей, переноса на новые платформы. Если не надеяться "на дядю", то единственное решение парадокса - стать некомпетентным разработчиком, - чтобы никто больше не был заинтересован в вашей продукции. В этой книге подобное решение не поощряется.
Своевременность	При использовании уже существующих компонентов нужно **меньше** разрабатывать, а, следовательно, ПО создается **быстрее**.
Инвестирование	Создание повторно используемого ПО позволяет сберечь плоды знаний и открытий лучших разработчиков, превращая временные ресурсы в постоянные.

Совместимость.	Если использовать хорошую современную ОО-библиотеку, то ее стиль повлияет, за счет естественного "процесса диффузии", на стиль разработки всего ПО. Это существенно помогает повысить качество программного продукта.

Для успешного проектирования и разработки ПО с повторным использованием компонентов должны выполняться три основных условия:

1. Возможность поиска необходимых системных компонентов. В организациях должен быть каталог документированных компонентов, предназначенных для повторного использования, который обеспечивал бы быстрый поиск нужных компонентов.

2. При повторном использовании необходимо удостовериться, что поведение компонентов предсказуемо и надежно. В идеале все компоненты, представленные в каталоге, должны быть сертифицированы, чтобы подтвердить соответствие определенным стандартам качества.

3. На каждый компонент должна быть соответствующая документация, цель которой – помочь разработчику получить нужную информацию о компоненте и адаптировать его к новому приложению. В документации должна содержаться информация о том, где используется данный компонент, и другие вопросы, которые могут возникнуть при повторном использовании компонента.

Успешное использование компонентов в приложениях Visual Basic, Visual C++ и Java продемонстрировало важность повторного использования. Разработка ПО, основанная на повторном использовании компонентов, становится широко распространенным рентабельным подходом к разработке программных продуктов.

Вместе с тем подходу к разработке ПО с повторным использованием компонентов присущ ряд недостатков и проблем, которые препятствуют запланированному сокращению расходов на разработку проекта.

Проблемы повторного использования

Проблема	Описание
Повышение стоимости сопровождения системы	Недоступность исходного кода компонента может привести к увеличению расходов на сопровождение системы, так как повторно используемые системные элементы могут со временем оказаться не совместимыми с изменениями, производимыми в системе

Недостаточная инструментальная поддержка	CASE-средства не поддерживают разработку ПО с повторным использованием компонентов. Интегрирование этих средств с системой библиотек компонентов затруднительно или даже невозможно. Если процесс разработки ПО осуществляется с помощью CASE-средств, повторное использование компонентов можно полностью исключить
Синдром "изобретения велосипеда"	Некоторые разработчики ПО предпочитают переписать компоненты, так как полагают, что смогут при этом их усовершенствовать. Кроме того, многие считают, что создание программ "с нуля" перспективнее и "благороднее" повторного использования написанных другими программ
Содержание библиотеки компонентов	Заполнение библиотеки компонентов и ее сопровождение может стоить дорого. В настоящее время еще недостаточно хорошо продуманы методы классификации, каталогизации и извлечения информации о программных компонентах
Поиск и адаптация компонентов	Компоненты ПО нужно найти в библиотеке, изучить и адаптировать к работе в новых условиях, что "не укладывается" в обычный процесс разработки ПО

Из перечисленного выше следует, что повторное использование компонентов должно быть систематическим, плановым и включенным во все организационные программы организации-разработчика.

Процессы повторного применения программных средств(стандарт)

Группа процессов повторного применения программных средств состоит из трех процессов, которые поддерживают возможности организации использовать повторно составные части программных средств за границами проекта. Эти процессы уникальны, поскольку, в соответствии с их природой, они используются вне границ какого-либо конкретного проекта.

Процессами повторного применения программных средств являются:

a) процесс проектирования доменов;

b) процесс менеджмента повторного применения активов;

c) процесс менеджмента повторного применения программ.

Подробно рассмотрим процесс повторного применения активов.

Менеджмент повторного применения активов

Цель процесса менеджмента повторного применения активов заключается в управлении жизненным циклом повторно применяемых активов от концепции до отмены применения.

В результате успешного осуществления процесса менеджмента повторного применения активов:

a) документируется стратегия менеджмента активов;

b) формируется схема классификации активов;

c) определяются критерии приемки активов, сертификации и прекращения применения;

d) приводится в действие механизм хранения и поиска активов;

e) регистрируется использование активов;

f) контролируются изменения в активах;

g) пользователи активов оповещаются о выявленных проблемах, выполненных модификациях, созданных новых версиях и удалениях активов из мест хранения и механизмов поиска.

Верификация и валидация ПО

Верификация и валидация, как методы, обеспечивают соответственно проверку и анализ правильности выполнения заданных функций и соответствия ПО требованиям заказчика, а также заданным спецификациям.

Основным методом повышения надежности ПО является верификация. Под верификацией программного обеспечения подразумевается процесс, направленный на подтверждение соответствия ПО заданным требованиям путем различного рода проверок и обеспечения объективных доказательств. Следует подчеркнуть, что процесс верификации должен носить комплексный характер, охватывающий все этапы разработки ПО (анализ требований, проектирование, кодирование, интеграцию), а также изменения, вносимые при сопровождении ПО. Необходимость верификации ПО для технических комплексов критического использования установлена в стандартах по программной инженерии, а также в различных отраслевых стандартах.

Согласно требованиям к процессам жизненного цикла ПО, процесс должен завершаться верификацией соответствующего продукта. Процесс верификации ПО включает следующие этапы:

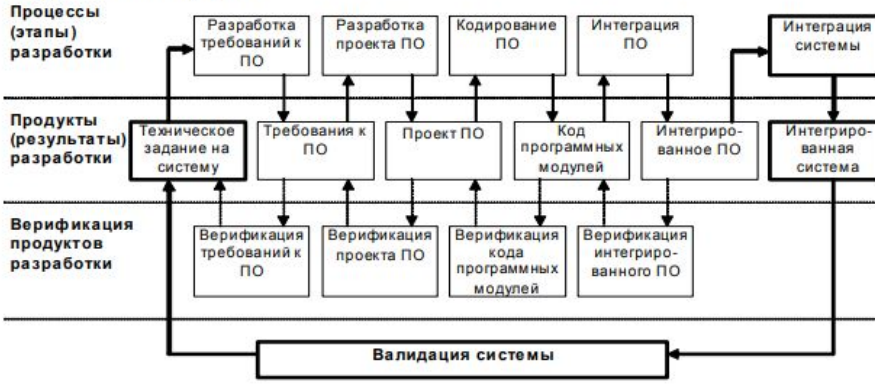
– верификация требований к программному обеспечению;

– верификация проекта программного обеспечения;

– верификация кода программных модулей;

– верификация интегрированного программного обеспечения

Структура процессов верификации и валидации программного обеспечения:



Процесс верификации ПО должен быть документирован. Перед началом верификации должен быть выпущен план верификации ПО. Отдельные части плана верификации ПО, имеющие самостоятельное значение (программы, методики испытаний, планы тестирования и т.п.), могут быть выпущены в виде отдельных документов.

Если отдельные части или функции ПО могут быть проверены только в составе системы (в процессе валидации), то это должно быть обосновано и отражено в плане верификации.

По результатам верификации должен быть выпущен отчет по верификации ПО. Отдельные части отчета по верификации ПО, имеющие самостоятельное значение (отчеты по испытаниям, журналы испытаний, протоколы испытаний и т.п.), могут быть выпущены в виде отдельных документов. Вся документация по разработке и верификации ПО должна быть изложена в доступной форме, понятной специалистам, не участвовавшим в проведении разработки и верификации ПО.

Как должен выглядеть отчет согласно требованиям стандарта IEEE 1012-1998 :

Введение
1. Объект верификации
2. Объем и цель верификации
3. Методы и средства, использовавшиеся при верификации
4. Порядок и особенности проведения верификации
5. Отчеты по верификации
5.1. Структура отчетов по верификации
5.2. Соответствие отчетов методикам верификации
6. Анализ результатов верификации
6.1. Результаты испытаний
6.2. Анализ недостатков и принятые меры
6.3. Результаты повторных испытаний
7. Общие выводы по верификации

В то же время отчет по валидации ПО должен содержать следующие разделы:

− конфигурация, используемая при тестировании

− оборудование, используемое при испытаниях, и сведения о его калибровке;

− используемые имитационные модели;

− список тестируемых входных данных;

− список тестируемых выходных данных;

− используемые методы имитации энергетической установки и ее систем или интерфейсных компонентов, которые использовались в процедуре верификации;

− дополнительные данные (синхронизация, последовательность событий и др.);

− соответствие критериям приемки, указанным в плане валидации;

− протокол регистрации выявленных отклонений, в котором приводится описание типа ошибки и выполненные корректирующие действия.

Фактически, отчет о валидации допустимо представлять в виде комплекта документов, состоящего из программ-методик испытаний и протоколов по результатам проведенных испытаний, так как все вышеприведенные пункты отчета о валидации должны в той или иной форме содержаться в этих документах. При этом отчеты о валидации должны быть представлены в виде, который позволяет проверить их лицам, непосредственно не участвовавшим в процедуре валидации.

Инспекция ПО

Инспекция ПО - это статическая проверка соответствия программы заданным спецификациями, проводится путем анализа различных представлений результатов проектирования (документации, требований, спецификаций, схем или исходного кода программ) на процессах ЖЦ. Просмотры и инспекции результатов проектирования и соответствия их требованиям заказчика обеспечивают более высокое качество создаваемых ПС.

При инспекции программ рассматриваются документы рабочего проектирования на этапах ЖЦ совместно с независимыми экспертами и участниками разработки ПС. На начальном этапе проектирования инспекция предполагает проверку полноты, целостности, однозначности, непротиворечивости и совместимости документов с исходными требованиями к программной системе. На этапе реализации системы под инспекцией понимается анализ текстов программ на соблюдение требований стандартов и принятых руководящих документов технологии программирования.

Эффективность такой проверки заключается в том, что привлекаемые эксперты пытаются взглянуть на проблему "со стороны" и подвергают ее всестороннему критическому анализу.

Гарантирование качества.

Гарантия качества (quality assurance): Все запланированные и систематические действия, выполняемые в рамках системы качества и продемонстрированные надлежащим образом для обеспечения соответствующей уверенности в том, что объект полностью удовлетворяет требованиям к качеству.

Существуют как внутренние, так и внешние цели гарантии качества:

a) внутренняя гарантия качества: в пределах организации гарантия качества обеспечивает уверенность руководству организации;

b) внешняя гарантия качества: в контрактных ситуациях гарантия качества обеспечивает уверенность заказчику или другим сторонам.

Некоторые действия по управлению качеством и гарантии качества взаимосвязаны.

До тех пор, пока требования к качеству полностью не удовлетворяют потребностям пользователя, гарантия качества не может обеспечить необходимой уверенности.

Квалификационное тестирование ПО в системном контексте.

Цель процесса квалификационного тестирования системы заключается в подтверждении того, что реализация каждого системного требования тестируется на соответствие и система готова к поставке.

Квалификационное тестирование системы должно проводиться в соответствии с квалификационными требованиями, установленными для системы. Должны обеспечиваться гарантии проверки выполнения каждого системного требования и готовности системы к поставке. Результаты квалификационного тестирования должны быть документированы.

Система должна быть оценена с учетом перечисленных ниже критериев:

a) тестовое покрытие системных требований;

b) соответствие ожидаемым результатам;

c) осуществимость функционирования и сопровождения.

Цель процесса квалификационного тестирования программных средств заключается в подтверждении того, что укомплектованный программный продукт удовлетворяет установленным требованиям.

В результате успешного осуществления процесса квалификационного тестирования программных средств:

a) определяются критерии для комплектованных программных средств с целью демонстрации соответствия с требованиями к программным средствам;

b) комплектованные программные средства верифицируются с использованием определенных критериев;

c) записываются результаты тестирования;

d) разрабатывается и применяется стратегия регрессии для повторного тестирования комплектованного программного средства при проведении изменений в программных составных частях.

Для повторного применения тестирования комплексированного программного средства при проведении изменений в программных составных частях должна быть разработана стратегия регрессии.

В результате успешного осуществления процесса квалификационного тестирования системы:

a) разрабатываются критерии для оценки соответствия системным требованиям;

b) комплексированная система тестируется, используя определенные критерии;

c) документируются результаты тестирования;

d) гарантируется готовность системы для поставки.

Вывод

Таким образом, в реферате подробно проработана тема “управление конфигурациями ПО, повторное использование ПО” с помощью большого количества интернет-ресурсов и стандарта, список которых приведен в разделе Список использованных ресурсов.

Данная тема считается особенно актуальной в данное время, поскольку позволяет контролировать весь процесс жизненного цикла программного обеспечения и уберечь его от не контролированного развития проекта, гарантируя, что все изменения учитываются и санкционируются в соответствии с принятой технологией разработки.

В результате написания реферата были подробно рассмотрены основные понятия разделов управления конфигурациями ПО и повторного использования ПО, также цели, задачи, объекты управления конфигурациями, рассмотрены элементы управления, такие как конфигурационная идентификация, контроль конфигурации, учет состояния конфигурации, а также ревизия и аудит конфигурации; определения и компоненты инженерии повторного использования ПО, преимущества и недостатки повторного использования, а также в рамках данной темы рассмотрены процессы верификации и валидации программного обеспечения, которые обеспечивают соответственно проверку и анализ правильности выполнения заданных функций и соответствия ПО требованиям заказчика и заданным спецификациям; инспекции ПО, гарантирование качества, и квалификационное тестирование.

Стандарт, на который опирается данный реферат, используя устоявшуюся терминологию, устанавливает общую структуру процессов жизненного цикла программных средств, на которую можно ориентироваться в программной индустрии, определяет процессы, виды деятельности и задачи, которые используются при приобретении программного продукта или услуги, а также при поставке, разработке, применении по назначению, сопровождении и прекращении применения программных продуктов.

Стандарт ISO/IEC 12207 был опубликован 1 августа 1995 года и явился первым международным стандартом, содержавшим представительный набор процессов ЖЦ, действий и задач в отношении ПО, которое рассматривалось как часть большей системы, а также применительно к программным продуктам и услугам. За стандартом ISO/IEC 12207 в ноябре 2002 года последовал стандарт ISO/IEC 15288, посвященный процессам ЖЦ систем. Широта применения ПС привела к тому, что ПО и процессы его разработки не могли рассматриваться в отрыве от систем, но только как составная часть системы и процесса её создания. В Дополнениях к стандарту ISO/IEC 12207 были введены цель процесса и его выходы и определена эталонная модель процесса, отвечающая требованиям стандарта ISO/IEC 15504-2. Международный стандарт ISO/IEC 12207:2008, представляет собой переработанные и исправленные дополнения к стандарту ISO/IEC 12207 и является первым шагом в стратегии SC7 по гармонизации спецификаций, имеющей целью создание полностью интегрированного набора процессов ЖЦ систем и программных средств и руководства по их применению.

Список использованных ресурсов:

ftp://ftp.vt.tpu.ru/study/Malchukov/public/ERS/GOST/12207-2010.pdf - стандарт

http://www.cyb.univ.kiev.ua/library/books/lavrishcheva-4.pdf - МЕТОДЫ И СРЕДСТВА ИНЖЕНЕРИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

http://www.k-press.ru/cs/2007/2/QA/QA.asp - Конфигурационное управление проектами разработки программного обеспечения

https://www.intuit.ru/studies/courses/2190/237/lecture/6134 - методы и средства программного обеспечения

https://www.intuit.ru/studies/courses/2190/237/lecture/6130 - методы и средства программного обеспечения

http://www.immsp.kiev.ua/publications/articles/2006/2006_3/Skljar_03_2006.pdf - РЕАЛИЗАЦИЯ ПРОЦЕССА ВЕРИФИКАЦИИ ДЛЯ РАЗРАБОТКИ НАДЕЖНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

http://jurnal.vniiem.ru/text/104/6.pdf - ОСОБЕННОСТИ ВЕРИФИКАЦИИ И ВАЛИДАЦИИ ПРОГРАММНО-ТЕХНИЧЕСКИХ КОМПЛЕКСОВ ВТОРОГО КЛАССА БЕЗОПАСНОСТИ В СОСТАВЕ ЭЛЕКТРООБОРУДОВАНИЯ СУЗ РЕАКТОРОВ ВВЭР

file:///C:/Users/%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C-%D0%BF%D0%BA/Downloads/5_CIT_vol_5_1_Smolarova_p_33.pdf - Software reuse: Principles, Patterns, Prospects

http://sewiki.ru/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D0%B5%D0%B9 - Управление конфигурацией

http://getbug.ru/uchet-sovmestimosti-cots-komponentov-pri-formirovanii-izbyitochnyih-programmnyih-sredstv/ - Учет совместимости COTS компонентов при формировании избыточных программных средств

https://www.ibm.com/developerworks/ru/library/plan-uk/index.html - Разработка плана управления конфигурацией

https://it-guild.com/info/blog/kak-vnedrit-protsess-upravleniya-konfiguratsiey-configuration-management-chast-1/ - Как внедрить процесс управления конфигурацией
https://ru.wikipedia.org/wiki/ISO/IEC_12207:2008#%D0%A0%D0%B0%D0%B7%D0%B2%D0%B8%D1%82%D0%B8%D0%B5_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%B0 - ISO/IEC 12207:2008, статья Википедии
ГОСТ Р ИСО/МЭК 12207-2010 - стандарт