Реферат

на тему:

Управление конфигурациями ПО. Повторное использование ПО

Подготовили: Орленко Родион, Швед Ирина

#### **История развития дисциплины управления конфигурацией**

Первым заметным шагом в развитии УК 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».

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.

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.

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.

#### **Базовые концепции и элементы**

Нельзя сказать, что никто до этого не использовал таких методов работы. Разработка и раньше велась параллельно с документированием. Ревизии документов проводились и раньше. Тестирование и испытания продукции на предмет соответствия требованиям проводились и до этого. Отчетность использовалась в проектах и раньше. Основатели дисциплины управления конфигурацией сделали другое – они 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:

1. Документы создаются для описания продукта и являются средством управления конфигурацией продукта.
2. Изменения в продукте контролируются посредством контроля изменений в документации.
3. Изменения в продукте не производятся до тех пор, пока они не сделаны в документации.
4. До того, как быть реализованными в документации и продукте, изменения должны быть формально утверждены.
5. Все изменения должны отслеживаться.
6. Конфигурационные объекты (продукты), документы и их версии нумеруются и именуются единообразно и недвусмысленно (или уникально).
7. Ведется отчетность о состоянии изменений, документов и продуктов.
8. Каждый документ периодически сравнивается с соответствующим ему документом верхнего уровня на предмет выявления несоответствий.
9. 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 (концепция 1)
2. Контроль конфигурации (концепции 2, 3, 4, 5, 6)
3. Учет состояния конфигурации (концепция 7)
4. Ревизия и аудит конфигурации (концепции 8 и 9)

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 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 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.

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.

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

В результате успешного осуществления процесса менеджмента конфигурации:

a) определяется стратегия менеджмента конфигурации;

b) определяются составные части, нуждающиеся в менеджменте конфигурации;

c) устанавливается базовая линия конфигурации;

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.

По [ISO/IEC 12207](http://sewiki.ru/ISO/IEC_12207) процесс управления конфигурацией является процессом применения административных и технических процедур на всем протяжении [жизненного цикла](http://sewiki.ru/%D0%96%D0%B8%D0%B7%D0%BD%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%86%D0%B8%D0%BA%D0%BB) 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.

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.

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.

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.”

**Таблица 12.1. 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 |

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 (табл. 12.2), которые препятствуют запланированному сокращению расходов на разработку проекта.

**Таблица 12.2. 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.

Одна из характерных черт инженерной деятельности в промышленности - использование готовых решений и деталей. В программировании промышленное использование готовых решений и программных продуктов еще не стало повседневной практикой, а сформировались признаки этой инженерной деятельности .  
  
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.

### **Инженерия повторного использования компонентов**

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 деятельностьклассифицируется 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.

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 репозитария 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 (бизнесдомены, 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.

# **Ожидаемые преимущества**

Повторное использование может обеспечить прогресс на следующих направлениях:

**[x]. Своевременность (timeliness)** (в том смысле, который определен при обсуждении показателей качества: быстрота доведения проектов до завершения и продукции до рынка). При использовании уже существующих компонентов нужно **меньше**разрабатывать, а, следовательно, ПО создается **быстрее**.

**[x]. Сокращение объема работ по сопровождению ПО (decreased maintenance effort)**. 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.

**[x]. Надежность**. Получая компоненты от поставщика с хорошей репутацией, вы имеете определенную гарантию, что разработчики предприняли все нужные меры, включая всестороннее тестирование и другие методы контроля качества. В большинстве случаев можно ожидать, что кто-то уже испытал эти компоненты до вас и обнаружил все возможно остававшиеся ошибки. Заметьте, вовсе не предполагается, что разработчики компонентов умнее вас. Для них создаваемые компоненты - будь то графические модули, интерфейсы баз данных, алгоритмы сортировки - это служебная обязанность, цель работы. Для вас это лишь второстепенная, рутинная работа, поскольку **вашей** целью является создание некоторой прикладной системы в вашей собственной области деятельности.

**[x]. Эффективность**. Факторы, способствующие возможности повторного использования ПО, побуждают разработчиков компонентов пользоваться наилучшими алгоритмами и структурами данных, известными в их конкретной сфере деятельности. Однако в команде, разрабатывающей большой прикладной проект, трудно ожидать наличия специалистов по **каждой** проблеме, затрагиваемой в этом проекте. При разработке большого проекта невозможно оптимизировать все его детали. Следует стремиться к достижению наилучших решений в своей области знаний, а в остальном использовать профессиональные разработки.

**[x]. 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.

**[x]. Инвестирование**. Создание повторно используемого ПО позволяет сберечь плоды знаний и открытий лучших разработчиков, превращая временные ресурсы в постоянные.

ftp://[ftp.vt.tpu.ru/study/Malchukov/public/ERS/GOST/12207-2010.pdf](http://ftp.vt.tpu.ru/study/Malchukov/public/ERS/GOST/12207-2010.pdf)

<http://www.cyb.univ.kiev.ua/library/books/lavrishcheva-4.pdf>

<http://www.k-press.ru/cs/2007/2/QA/QA.asp>

<https://docs.google.com/document/d/1YKCiozrdjxCxAzJ0UFzg8pkdTp85o827hvVcGWuY_pc/edit#>

<https://www.intuit.ru/studies/courses/2190/237/lecture/6134>

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