**Лекция 1 Введение в безопасное программирование. Жизненный цикл безопасной разработки программного обеспечения (SSDLC)**

**Зачем и от кого нужно защищать программное обеспечение компьютерных систем**

Причины, приводящие к функционально непригодному результату:

сбои компьютерных систем, ошибки программистов и операторов, дефекты в программах (преднамеренные и непреднамеренные).

* Кто потенциально может осуществить преднамеренно практическое внедрение программных дефектов деструктивного воздействия в исполняемый программный код ?
* Каковы возможные мотивы действий субъекта, осуществляющего разработку таких дефектов?
* Как можно идентифицировать наличие программного дефекта?
* Как можно отличить преднамеренный программный дефект от программной ошибки?
* Каковы наиболее вероятные последствия активизации деструктивных программных средств при эксплуатации компьютерных систем?

**Жизненный цикл программного обеспечения компьютерных систем**

Software Development Lifecycle (SDLC) - жизненный цикл разработки программного обеспечения.

Secure Software Development Lifecycle (SSDLC) - жизненный цикл безопасной разработки программного обеспечения

*Жизненный цикл ПО* определяется как период времени, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент его полного изъятия из эксплуатации.

Основным нормативным документом, регламентирующим состав процессов ЖЦ ПО, является международный стандарт ISO/IEC 12207: 2017.

* *ПО* или *программный продукт* определяется как набор компьютерных программ, процедур и, возможно, связанной с ними документации и данных.
* *Программа* – набор операторов, который может быть представлен как единое целое в некоторой вычислительной системе и который используется для управления поведением этой системы.
* *Процесс* определяется как совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные.

***Основные процессы ЖЦ***

* **Процесс приобретения** (acquisition process)
* **Процесс поставки** (supply process)
* **Процесс разработки** (development process)

 *анализ, проектирование* и *реализация* (*программирование*)

* **Процесс эксплуатации** (operation process)
* **Процесс сопровождения** (maintenance process)

*Сопровождение* – внесение из­менений в ПО в целях исправления ошибок, повышения произво­дительности или адаптации к изменившимся условиям работы или требованиям.

***Вспомогательные процессы ЖЦ***

* **Процесс документирования** (documentation process)
* **Процесс управления конфигурацией** (configuration management process)

*Конфигурация ПО* – совокупность его функциональных и физических ха­рактеристик, установленных в технической документации и реали­зованных в ПО

* **Процесс обеспечения качества** (quality assurance process)
* **Процесс обеспечения качества** (quality assurance process)

*Качество ПО* – совокупность свойств, которые характе­ризуют способность ПО удовлетворять заданным требованиям

* **Процесс верификации** (verification process)

*Верификация* – процесс определения того, отвечает ли текущее состояние разработки, достигнутое на данном этапе, требованиям этого этапа.

*Верификация* в узком смысле означает формальное доказательство правильности ПО.

* **Процесс аттестации** (validation process)

Под *аттестацией ПО* обычно понимается подтвер­ждение и оценка достоверности проведенного тестирования ПО.

* **Процесс совместной оценки** (joint review process)
* **Процесс аудита** (audit process)

*Аудит* – это ревизия (проверка), проводимая компетентным органом (лицом) в целях обеспечения независимой оценки степени соответствия ПО или процессов установленным требованиям.

* **Процесс разрешения проблем** (problem resolution process)

***Организационные процессы ЖЦ ПО***

* **Процесс управления** (management process)
* **Процесс создания инфраструктуры** (infrastructure process)
* **Процесс усовершенствования** (improvement process)
* **Процесс обучения** (training process)

*Модели ЖЦ* ПО

*Модель ЖЦ* - структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ.

*Стадия создания ПО* - часть процесса создания ПО, ограниченная некоторыми временными рамками и заканчивающаяся выпуском конкретного продукта (моделей ПО, компонентов, документации).

Стадии создания ПО обычно выделяются в следующие группы:

* Формирование требований к ПО;
* Анализ;
* Проектирование;
* Реализация (Кодирование);
* Тестирование и Отладка;
* Внедрение и сопровождение;
* Снятие с эксплуатации.

**Жизненный цикл функциональной надежности ПО**

К настоящему времени в области создания безопасного ПО известно определенное количество документов, оформленных в виде корпоративных, отраслевых и международных стандартов и содержащих указания для разработчиков ПО и «лучшие практики», которые рекомендуется внедрять в жизненном цикле ПО с целью создания ПО с минимально возможным количеством уязвимостей и формирования среды обеспечения оперативного устранения выявленных пользователем ПО проблем (уязвимостей ПО).

При формировании требований к программному обеспечению (ПО) прежде всего должна быть выбрана модель жизненного цикла (ЖЦ) обеспечения его функциональной надежности. На рис. 1 приведена широко распространенная в международной практике V-диаграмма ЖЦ разработки функционально надежного ПО. По этой диаграмме по нисходящей ветви четко прослеживается последовательность формирования требований, архитектуры ПО, последовательность проектирования системы ПО, составных программных модулей и кодирование программных модулей.



Рис.1. V-диаграмма жизненного цикла обеспечения функциональной надежности ПО

**Анализ требований к системе** подразумевает определение ее функциональных возможностей, пользовательских требований, требований к надежности и безопасности, требований к внешним интерфейсам и т.д. Требования к системе оцениваются исходя из критериев реализуемости и возможности проверки при тестировании на основе предварительной оценки рисков.

**Анализ требований к ПО** предполагает определение следующих характеристик для каждого компонента ПО:

* функциональных возможностей, включая характеристики производительности и среды функционирования компонента;
* внешних интерфейсов;
* спецификаций функциональной надежности и безопасности;
* эргономических ребований;
* требований к используемым данным;
* требований к установке и приемке;
* требований к пользовательской документации;

требований к эксплуатации и сопровождению.

**Проектирование архитектуры системы** на высоком уровне заключается в определении компонентов ее оборудования, ПО и операций, выполняемых эксплуатирующим систему персоналом. Архитектура системы должна соответствовать требованиям, предъявляемым к системе, а также принятым проектным стандартам и методам.

**Проектирование архитектуры ПО** включает следующие задачи (для каждого компонента ПО):

* трансформацию требований к программным средствам в архитектуру, определяющую на высоком уровне структуру программных средств и состав его компонентов;
* разработку и документирование программных интерфейсов программных средств и баз данных;
* разработку предварительной версии пользовательской документации;
* разработку и документирование предварительных требований к тестам и плана интеграции программных средств.

**Проектирование архитектуры системы** на высоком уровне заключается в определении компонентов ее оборудования, ПО и операций, выполняемых эксплуатирующим систему персоналом. Архитектура системы должна соответствовать требованиям, предъявляемым к системе, а также принятым проектным стандартам и методам.

**Проектирование архитектуры ПО** включает следующие задачи (для каждого компонента ПО):

* трансформацию требований к программным средствам в архитектуру, определяющую на высоком уровне структуру программных средств и состав его компонентов;
* разработку и документирование программных интерфейсов программных средств и баз данных;
* разработку предварительной версии пользовательской документации;
* разработку и документирование предварительных требований к тестам и плана интеграции программных средств.

Под верификацией программного средства подразумеваются действия по определению того, что выходные данные каждой фазы жизненного цикла отвечают требованиям предыдущей фазы. Эти действия должны осуществляться посредством анализа и испытаний.

В процессе верификации проверяются следующие условия:

* непротиворечивость требований к системе и степень учета потребностей пользователей;
* возможности поставщика выполнить заданные требования;
* соответствие выбранных процессов жизненного цикла программных средств условиям договора;
* адекватность стандартов, процедур и среды разработки процессам жизненного цикла программных средств;
* соответствие проектных спецификаций программных средств заданным требованиям;
* корректность описания в проектных спецификациях входных и выходных данных, последовательности событий, интерфейсов, логики и т.д.;
* соответствие кода проектным спецификациям и требованиям;
* тестируемость и корректность кода, его соответствие принятым стандартам кодирования;
* корректность интеграции компонентов программных средств в систему;
* адекватность, полнота и непротиворечивость документации.

**Валидация.**

Завершающая стадия разработки — это аттестация (валидация) программного обеспечения, под которой понимаются процедуры получения доказательственной базы для подтверждения соответствия ПО заданным требованиям по функциональной надежности.

Процесс валидации (*validation process*) предусматривает определение полноты соответствия заданных требований и созданной системы или программного продукта их конкретному функциональному назначению. Под *аттестацией* обычно понимаются подтверждение и оценка достоверности проведенного тестирования ПО. Валидация должна гарантировать полное соответствие ПС спецификациям, требованиям и документации, а также возможность его безопасного и надежного применения пользователем. Валидацию рекомендуется выполнять путем тестирования во всех возможных ситуациях и использовать при этом независимых специалистов. Валидация может проводиться на начальных стадиях жизненного цикла программных средств или как часть работы по приемке ПО.

**Жизненный цикл обеспечения безопасности ПО**

На рис.2. приведена схема жизненного цикла обеспечения безопасности ПО



Рис. 2. Жизненный цикл безопасного ПО

Совокупность стадий данной модели ЖЦ целесообразно делить на две части, существенно различающиеся особенностями процессов, технико-экономическими характеристиками и влияющими на них факторами.

В первой части ЖЦ производится системный анализ, оценка рисков, проектирование, разработка, тестирование и испытания ПО. Номенклатура работ, их трудоемкость, длительность и другие характеристики на этих стадиях существенно зависят от объекта и среды разработки. Изучение подобных зависимостей для различных классов ПО позволяет прогнозировать состав и основные характеристики графиков работ для новых версий ПО.

Этой совокупности стадий ЖЦ ПО соответствует процесс внесения в разрабатываемые программы определенных защитных функций. Этот процесс называется *обеспечением функциональной безопасности ПО* и характеризуется необходимостью предотвращения его модификации за счет внедрения вычислительных процессов априорного типа.

Вторая часть ЖЦ, отражающая поддержку эксплуатации и сопровождения ПО, относительно слабо связана с характеристиками объекта и среды разработки. Номенклатура работ на этих стадиях более стабильна, а их трудоемкость и длительность могут существенно изменяться и зависят от массовости применения ПО. Для любой модели ЖЦ обеспечение высокого качества программных комплексов возможно лишь при использовании регламентированного технологического процесса на каждой из этих стадий.

Стадиям эксплуатации и сопровождения ПО соответствует процесс *обеспечения эксплуатационной безопасности ПО.* Этот процесс характеризуется необходимостью защиты программ от вычислительных процессов апостериорного типа. Последние могут внедряться за счет злонамеренного использования методов исследования программ. Кроме того, на стадии обеспечения эксплуатационной безопасности ПО применяются методы защиты программ от несанкционированного исследования и копирования