SWEBOK. Основные области знаний

Алексей Островский

Физико-технический учебно-научный центр НАН Украины

03 октября 2014 г.

Ядро знаний SWEBOK

SWEBOK (software engineering body of knowledge) — основной научно-технический документ по программной инженерии, отображающий знания и накопленный опыт специалистов по программной инженерии.

Ядру знаний SWEBOK соответствует стандарт ISO/IEC TR 19759:2005.

Версии:

- 2004 г. (SWEBOK V2) — десять областей знаний (5 основных и 5 областей управления);
- 2013 г. (SWEBOK V3) — пятнадцать областей (+ теоретические основы ПИ, экономика и описание профессиональных навыков по ПИ).

Основные области знаний SWEBOK

Требования к ПО

Определение

Требование к программному обеспечению (англ. *software requirement*) — это:

- характеристика ПО, с помощью которой конечным пользователем ПО решается какая-либо задача или достигается определенная цель;
- характеристика или свойство ПО, определенное контрактом на его разработку или другим документом (стандартом, спецификацией и т. п.).

Цель требований: определение функций, условий и ограничений, присущих ПО; спецификация данных, технического сопровождения и среды исполнения.

Виды требований

Требования к продукту и процессу — условия выполнения и режим работы ПО, ограничения на среду исполнения; определение принципов взаимодействия с другими программами.

Функциональные требования — определяют назначение и функции системы.

Нефункциональные требования — определяют условия исполнения ПО, переносимости и доступа к данным.

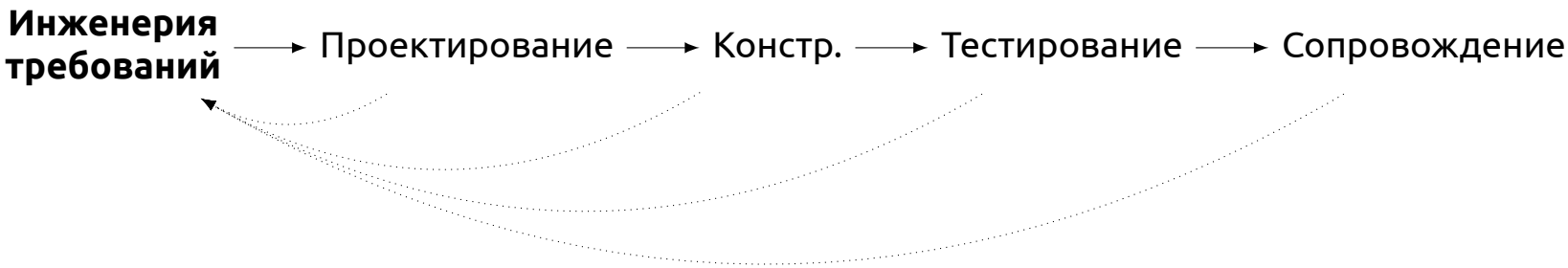
Системные требования — требования к программной системе в целом.

Инженерия требований

Определение

Инженерия требований (англ. *requirements engineering*) — процесс формулировки, документирования и поддержки требований к ПО, а также соответствующая область программной инженерии.

В классической модели жизненного цикла (англ. *waterfall model*) инженерия требований — начальный процесс разработки ПО.



В других моделях (RUP, extreme programming, Scrum) требования уточняются в процессе разработки.

Составляющие инженерии требований

определение (*elicitation*) → анализ (*analysis*) → спецификация (*specification*) → проверка (*validation*) → управление (*management*)

- Извлечение информации из договоров;
- проведение собеседований;
- согласование с заказчиком.

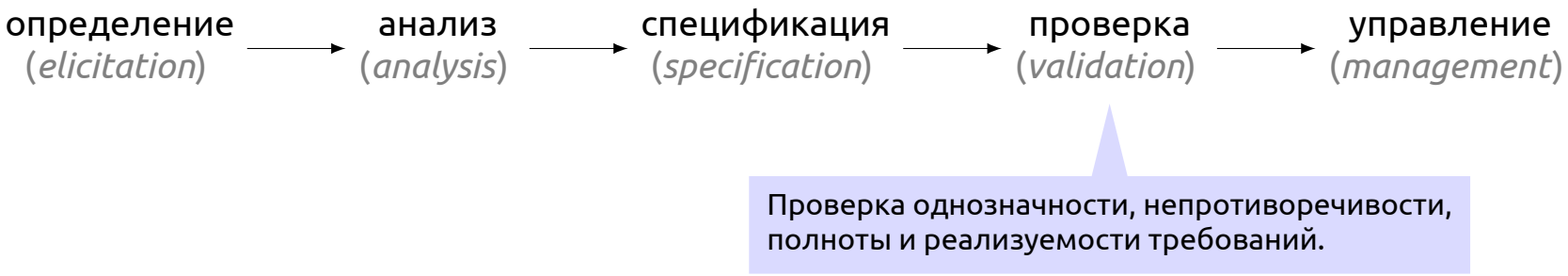
Составляющие инженерии требований

- Изучение потребностей и целей пользователей;
- требования к системе исполнения, аппаратуре и ПО;
- устранение конфликтов между требованиями;
- определение приоритетов и принципов взаимодействия с окружением.

Составляющие инженерии требований

- Формальное описание требований;
- спецификация требований к структуре ПО, функциям, качеству и документации;
- задание архитектуры и логики системы.

Составляющие инженерии требований



Составляющие инженерии требований

- Интеграция требований во все процессы ЖЦ;
- контроль реализации требований;
- необходимая корректировка требований.

Проектирование программного обеспечения

Определение

Проектирование ПО (англ. *software design*) — процесс определения архитектуры ПО, набора составляющих компонентов и их интерфейсов, прочих характеристик системы и конечного состава программного продукта.

Основные **концепции** проектирования ПО:

- **абстрагирование** (отсеивание лишней информации) и **уточнение** (построение иерархии выполнения);
- **модульность** (выделение автономных компонентов системы) и **архитектура** (общая структура системы, связывающая все компоненты);
- **структуризация** (представления взаимоотношений между данными) и **инкапусляция** (отделение реализации от представления).

Архитектура ПО

Определение

Архитектура программного проекта — высокоуровневое представление структуры системы и спецификация ее компонентов и логики их взаимодействия.

Преимущества использования архитектуры ПО:

- основа для анализа системы на ранних этапах ее разработки;
- основа для повторного использования компонентов и решений;
- упрощение принятия решений касательно разработки, развертывания и поддержки ПО;
- упрощение диалога с заказчиком;
- уменьшение рисков и снижение затрат на производство ПО.

Шаблоны проектирования

Определение

Шаблон проектирования (англ. *design pattern*) — типовой конструктивный элемент программной системы, задающий взаимодействие нескольких компонентов системы, а также роли и сферы ответственности исполнителей.

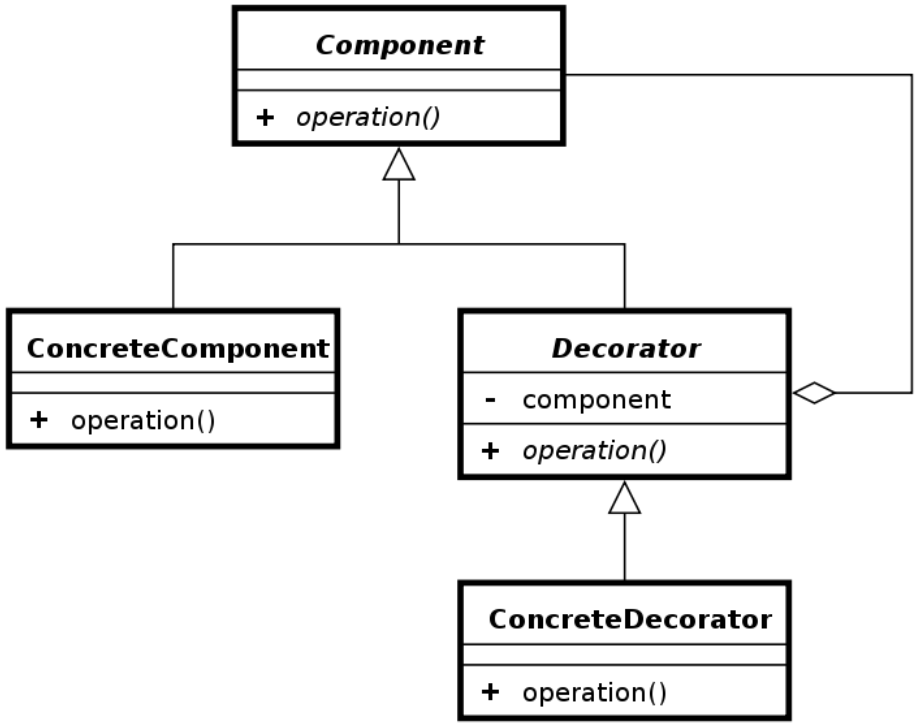
Виды шаблонов:

- порождающие (англ. *creational patterns*) — связанные с созданием объектов.
  Пример: фабрика (factory), синглтон (singleton).
- структурные (англ. *structural patterns*) — определяющие структуру композиции из нескольких объектов.
  Пример: мост (bridge), декоратор (decorator).
- поведенческие (англ. *behavioral patterns*) — определяющие поведение объектов.
  Пример: итератор (iterator).

Инструменты проектирования

Описание элементов ПО осуществляется с помощью **нотаций проектирования**.

Структурная нотация — представление основных аспектов элементов ПО, их интерфейсов и взаимосвязей.



UML-описание шаблона «Декоратор»

Инструменты:

- ADL (architecture description language);
- UML (unified modeling language);
- ERD (entity – relation diagrams);
- IDL (interface description language).

Инструменты проектирования

Описание элементов ПО осуществляется с помощью **нотаций проектирования**.

Поведенческая нотация — определенное представление динамики работы системы и ее элементов.

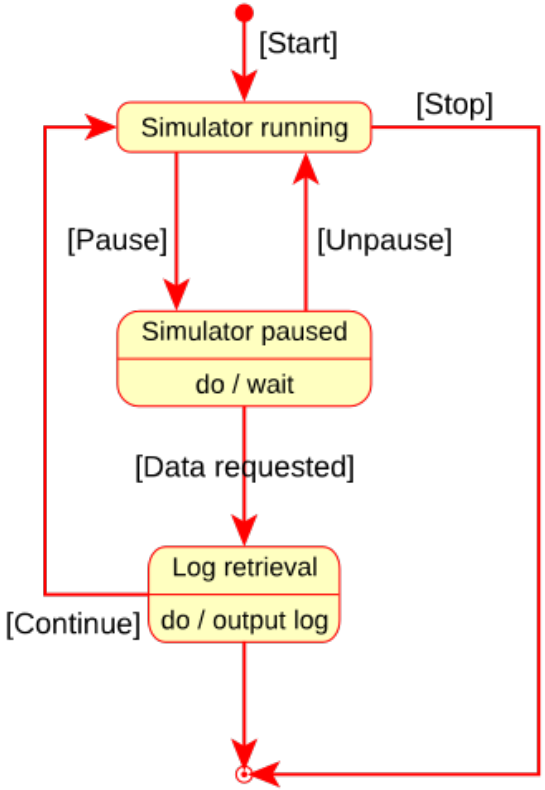


Диаграмма состояний эмулятора

Инструменты:

- диаграммы потоков данных (data flow);
- таблицы принятия решений (decision tables);
- формальные языки спецификации (Z, VDM, RAISE).

Конструирование ПО

Определение

Конструирование ПО (англ. *software construction*) — создание ПО из составных элементов (блоков, операторов, функций) и его проверка методами верификации и тестирования.

Техники конструирования: кодирование, верификация, модульное тестирование (unit testing), тестирование итеграции (integration testing), отладка (debugging).

Инструменты конструирования: языки конструирования; программные методы и инструментальные системы (компиляторы, СУБД, генераторы отчетов, системы управления версиями, конфигурацией, тестированием).

Основы конструирования ПО

- **Снижение сложности** (англ. *minimizing complexity*) — акцент на читаемости кода, а не его «красоте». Keep it simple, stupid (KISS).

  Средства достижения: стандарты кодирования.

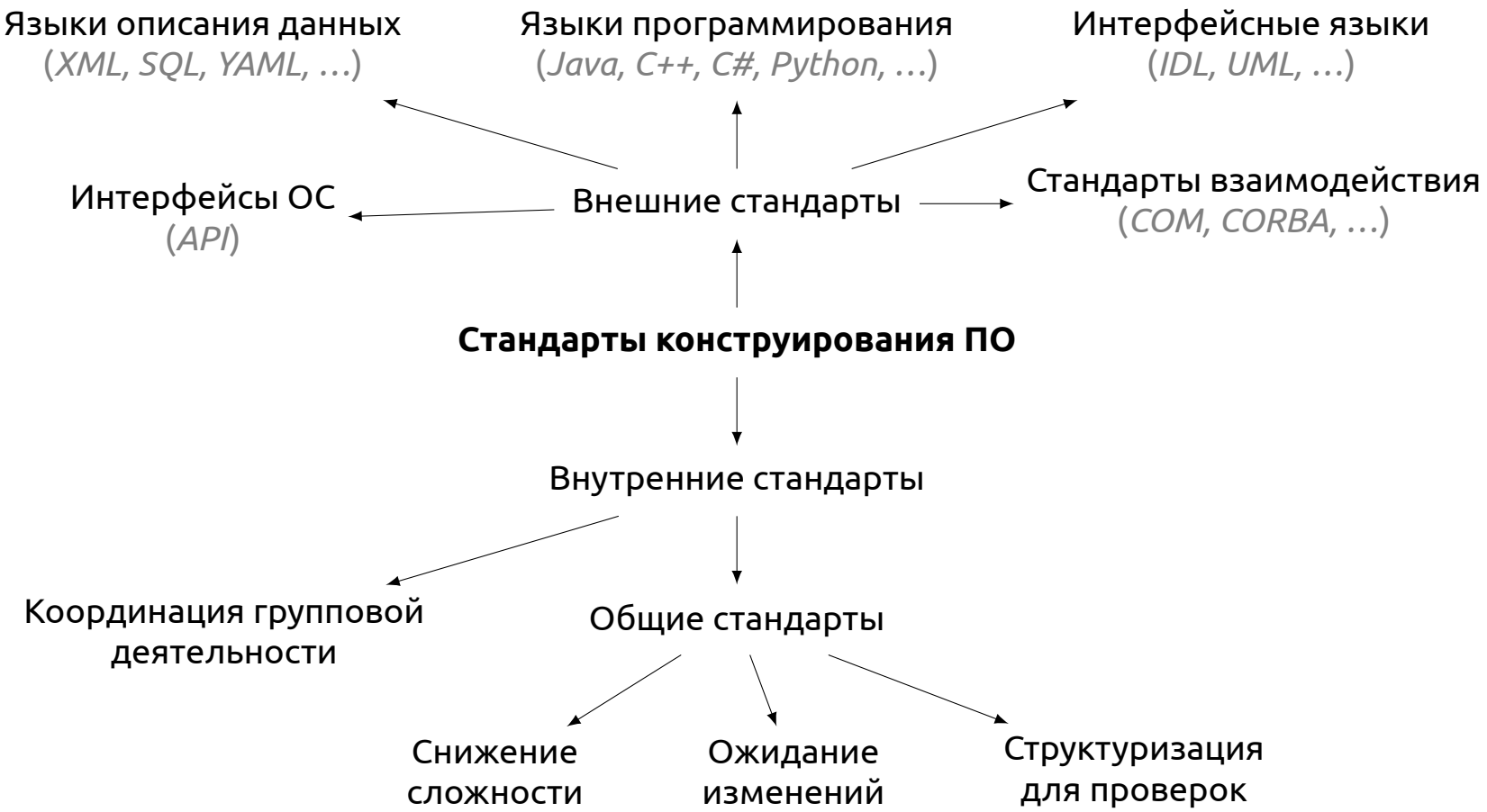
- **Ожидание изменений** (англ. *anticipating change*) — использование инструментов, позволяющих оперативно вносить изменения и дополнять код.

  Средства достижения: средства коммуникации (напр., системы документирования), диаграммы UML.

- **Структуризация для проверок** (англ. *constructing for verification*) — построение ПО, ошибки в котором выявляются на как можно более ранних этапах.

  Средства достижения: обзоры кода (code reviews), модульное тестирование, автоматизация тестирования.

Стандарты при конструировании



Управление конструированием

Определение

Управление конструированием (англ. *managing construction*) — управление процессом конструирования ПО, включающим в себя следующие этапы:

1. создание модели конструирования, зависящей от выбранной модели жизненного цикла ПО;
2. планирование конструирования — определение расписания конструкторских работ и их распределения между исполнителями;
3. измерение показателей — выработка количественных показателей (напр., объем нового / повторно использованного кода; сложность кода; число обнаруженных / исправленных ошибок) для корректировки процесса разработки.

Тестирование ПО

Определение

Тестирование ПО — это процесс проверки готовой программы в *статике* (обзоры кода, инспекции и т. п.) и *динамике* (прогон программы на тестовых данных) с целью обеспечить ее соответствие заданным требованиям.

Виды тестирования:

- модульное (unit testing);
- интеграционное (integration testing);
- системное (system testing);
- приемка (acceptance testing).

Основные понятия тестирования

Этапы возникновения сбоев в программе:

1. программист совершает *ошибку* (error, mistake);
2. ошибка приводит к *дефекту* (defect, fault, bug) в исходном коде;
3. при определенных условиях исполнения дефект приводит к *сбою программы* (program failure).

Тест — набор входных данных и прочих условий (напр., характеристики операционной системы и оборудования), которые полностью определяют ход выполнения программы.

Цель тестирования — локализация и устранение дефектов, соответствующие всем сбоям программы, обнаруженным с помощью тестов.

Прогон *всех* тестов невозможен даже для простых систем. $\Rightarrow$ Необходим отбор *информативных* тестов.

Методы тестирования

- **Белый ящик** (англ. *white box testing*), структурное тестирование — тестирование внутренних структур и операций ПО.

  Виды: тестирование API, внедрение ошибок (fault injection), покрытие кода (code coverage), мутационное тестирование (mutation testing), статическое тестирование (static testing).

- **Черный ящик** (англ. *black box testing*) — тестирование функциональности, доступной конечному пользователю ПО.

  Виды: анализ граничных значений (boundary value analysis), таблицы принятия решений (decision table testing), тестирование прецедентов (use case testing), тестирование потоков данных (data flow testing) и т. д.

- **Серый ящик** (англ. *gray box testing*) — тестирование ПО с частичным знанием о его внутренней структуре.

  Виды: тестирование интерфейсов компонентов системы, анализ обработки ошибок и т. п.

Типы тестирования

- **тестирование установки** ПО;
- **тестирование совместимости** (напр., с операционной системой и оборудованием);
- **проверка работоспособности** (англ. *sanity check*) — проверка на отсутствие тривиальных ошибок;
- **функциональное тестирование** — проверка реализации функций ПО, определенных в требованиях, и корректности их исполнения;
- **регрессионное тестирование** — повторная проверка функциональности ПО после внесения значительных изменений;

Типы тестирования (продолжение)

- **тестирование эффективности** — проверка скорости исполнения, продуктивности, используемого объема памяти и т. п.;
- **стресс-тестирование** — проверка поведения системы при превышении допустимой нагрузки;
- **альфа-** и **бета-тестирование** — тестирование системы тестировщиками разработчика (альфа) и ограниченной группой сторонних пользователей (бета);
- **тестирование безопасности**;
- **тестирование интерфейса** ПО, usability, локализации и т. д.

Управление тестированием

Основные этапы тестирования:

1. планирование процесса тестирования, составление планов, тестов, наборов данных;
2. проведение тестирования компонентов повторного использования и шаблонов;
3. генерация тестовых сценариев, соответствующих среде выполнения ПО;
4. сбор сведений об отказах ПО и выявленных исключительных ситуациях;
5. подготовка отчетов о результатах тестирования и оценка характеристик системы.

Сопровождение ПО

Определение

Сопровождение ПО (англ. *software maintenance*) — совокупность действий по обеспечению работы ПО, внесению изменений при выявлении ошибок, адаптации к новой среде исполнения, улучшения продуктивности или других характеристик ПО.

Основные вопросы сопровождения ПО:

- технические вопросы (напр., тестирование, анализ изменений);
- вопросы управления (напр., организация персонала);
- экономические вопросы (оценка стоимости сопровождения);
- измерительные вопросы (создание метрик для анализа эффективности сопровождения).

Категории сопровождения

По времени	По наличию ошибки	
	исправление	совершенствование
упреждающее	предотвращение	улучшение
ответное	корректировка	адаптация

- **корректировка** — устранение выявленных ошибок или нереализованных требований;
- **адаптация** — настройка продукта к изменившимся условиям эксплуатации;
- **предотвращение ошибок** — устранение скрытых дефектов, которые потенциально могут привести к сбоям;
- **улучшение** — увеличение продуктивности или повышение уровня сопровождения.

Методики сопровождения ПО

- **Понимание программ** (англ. *program comprehension*) — чтение и осмысление кода ПО с целью внесения в него изменений.
- **Реинженерия** (англ. *reengineering*) — усовершенствование ПО путем реорганизации или реструктуризации, а также настройка параметров и программных элементов для новой среды исполнения.
- **Реверсная инженерия** (англ. *reverse engineering*) — анализ ПО для выделения его компонент и взаимоотношений между ними.
  Используется для создания абстрактного представления ПО, чаще всего для перепрограммирования ПО для новой среды исполнения.
- **Рефакторинг** (англ. *refactoring*) — реорганизация кода программы для улучшения ее структуры, не изменяющая функциональность ПО.

Выводы

1. Ядро SWEBOK содержит пять основных областей знаний программной инженерии (инженерия требований, проектирование, конструирование, тестирование и сопровождение ПО), которые соответствуют процессам жизненного цикла ПО.
2. Основные области знаний содержат в себе как *теоретические основы*, так и систематизированные *практические навыки* разработки ПО, а также *методы управления* процессами разработки.
3. Все пять основных процессов ЖЦ тесно связаны между собой; особенности их взаимодействия сильно зависят от выбранной модели жизненного цикла.

Материалы

- Лавріщева К. М.

  Програмна інженерія (підручник).

  К., 2008. — 319 с.

- Описание стандарта SWEBOK.

  http://www.computer.org/portal/web/swebok/html/contents

Спасибо за внимание!