Программная инженерия

SWEBOK KA-11:

SE Professional Practice

Вопросы

- Профессиональные качества
- Групповая работа
- Communication Skills (человеко-человечный интерфейс ^_^)

Зачем

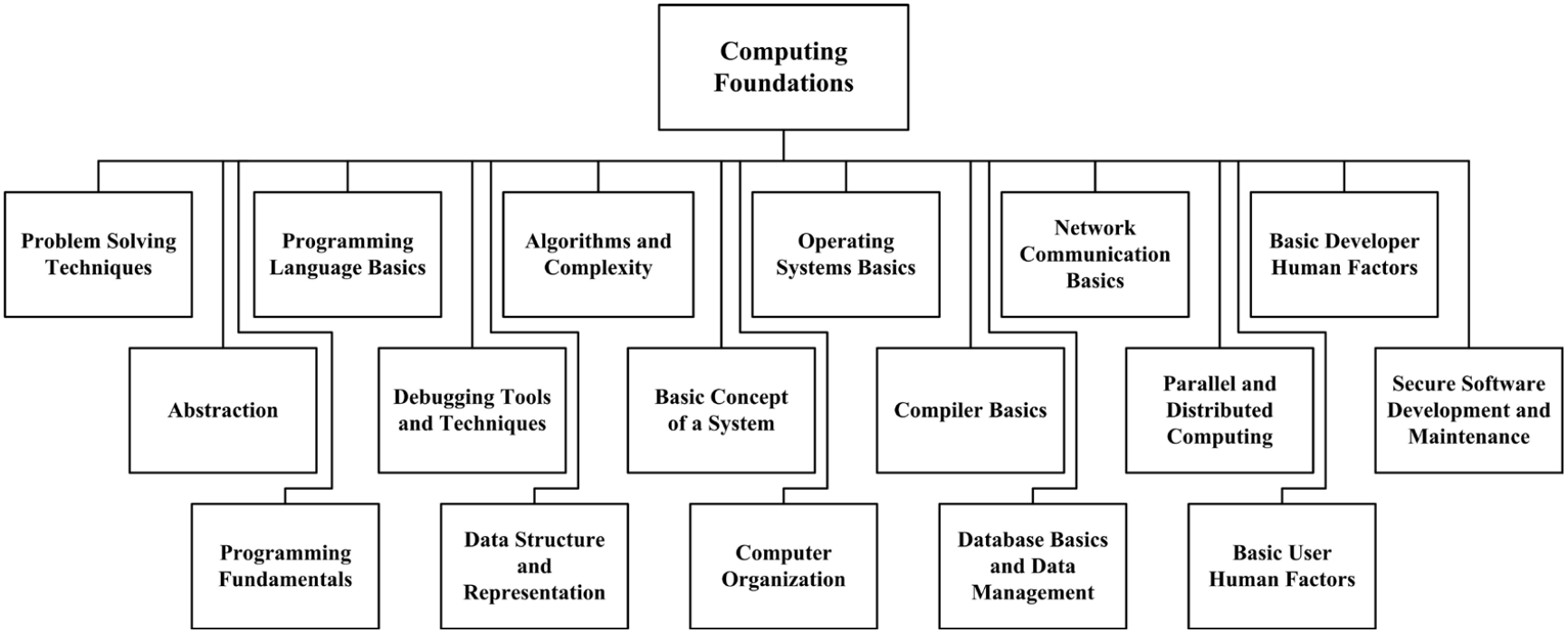
Состав раздела

SWEBOK: SE PROFESSIONAL PRACTICE (KA-11)

Причина введения КА

- Данный КА описывает среду, в которой ПО развивается (evolves) и выполняется
- Большая часть тем, затронутых в КА, также обсуждается в курсах специальных дисциплин. Однако, у таких дисциплин есть недостатки:
  - Часть тем из них не является связанной и/или важной для ПИ. Например, компьютерная графика
  - Часть тем не выделена как отдельные дисциплины и/или «размазаны» по нескольким дисциплинам

Структура КА



Практические замечания

- Всего в КА выделено 17 тем
- Для каждой будет приведено:
  - Содержание согласно КА
  - Ссылка на конкретные дисциплины и/или источники дополнительной информации

Общий подход

Формулирование

Анализ

Стратегии поиска решения

РАЗРЕШЕНИЕ ПРОБЛЕМ

Общий подход

- Разрешение проблем называется обдумывание и операции (activities), приводящие к получению ответа или выработке решения проблемы
  - Например, ПИ концентрируется на решение проблем с помощью компьютеров и ПО
- В общем случае, выделяется три этапа разрешения проблемы:
  - Формулировка [проблемы]
  - Анализ [проблемы]
  - Разработка стратегии поиска решения

Формулирование

- Сформулированная проблема должна явно указывать собственно проблему и желаемый результат ее разрешения
- Нет универсального способа. Среди наиболее общих подходов выделяют:
  - Определение источника и причины [проблемы]
  - Переформулирование проблемы
  - Анализ текущего и желаемого состояния
  - «Свежий взгляд» (fresh eye approach)

Анализ – 1/2

- Анализ проблемы помогает структурировать поиск решения
- Использует следующие виды анализа:
  - Ситуационный
    - Сначала исследуются наиболее критические или срочные аспекты ситуации
  - Проблемный (problem analysis)
    - Определяются причины возникновения проблемы
  - Анализ решений (decision analysis)
    - Определяются действия, необходимые для решения проблемы или исключения причин ее возникновения

Анализ – 2/2

- Виды анализа (продолжение):
  - Анализ потенциальных проблем (potential problem analysis)
    - Определяются действия, необходимые для предотвращения повторного возникновения старой проблемы или появления новых проблем

Разработка стратегии поиска решения

- «Лучшее» решение может быть определено различным образом:
  - Быстрее
  - Дешевле
  - Более легкое в использовании
  - Имеющее новые функции, etc.
- Необходимо:
  - Исключить пути ведущие к менее «хорошим» решениям
  - Определять задачи таким образом, чтобы направлять поиск в верном направлении
  - Установить характеристики конечного решения, показывающие его «хорошесть»

Использование компьютера

- Основные вопросы:
  - Как объяснить компьютеру что надо сделать?
  - Как представить описание проблемы в машинно-читаемом виде?
  - Как описать процесс решения проблемы на ЯП?

Уровни

Инкапсуляция

Иерархия

Альтернативы

АБСТРАКЦИИ

Абстракция

- Абстракция незаменима в решении проблем. Она применима как к процессу решения, так и к получаемому результату путем уменьшения количества одновременно «видимой» информации о концепции, проблеме, наблюдаемом явлении → «big picture»
- Важным инженерным навыком является умение выбрать верный уровень абстракции
- Абстракция позволяет распознать отношения между различными аспектами проблемы и найти более эффективное ее решение

Уровни абстракции – 1/2

- Вводя абстракцию мы сосредотачиваем внимание на одном уровне за раз, принимая что мы можем эффективно соединить этот уровень с уровнями выше и ниже
  - Однако, это не значит, что при выделении уровня мы ничего не «знаем» о соседних уровнях

Уровни абстракции – 2/2

- Обычно уровень абстракции соответствует не отдельному компоненту системы, а некоему «уровню представления» нижележащих элементов. Примеры:
  - Подсистемы
  - Сетевые уровни OSI ISO
- Взаимодействие между уровнями задает интерфейс (например API). Преимущество стандартизованных интерфейсов уровней заключается в облегчении переносимости и упрощении интеграции в дальнейшем

Инкапсуляция

- Инкапсуляция – это механизм реализации абстракции. Когда мы выбираем уровень абстракции уровни выше и ниже инкапсулируются
  - Инкапсулируемой информацией может быть концепция, проблема, наблюдаемое явление. Допустимые операции с ними – интерфейс
  - Обычно прячут часть информации о других уровнях (упрощая ее). Например, подробности устройства или способы выполнения запрашиваемых действий

Иерархия – 1/2

- При работе с информацией мы используем в разное время разные уровни абстракции. Обычно, эти разные уровни образуют иерархию
  - Выделение иерархии процесс индивидуальный и результат зависит от исполнителя
- Иерархия является естественным результатом декомпозиции
  - Анализ задач приводит к созданию иерархии задач и подзадач (WBS – work breakdown structure)

Иерархия – 2/2

- Варианты иерархий:
  - Последовательная
    - Ровно один уровень выше и ниже (исключая крайние)
  - (Чаще) Древовидная
  - (Редко) Многие-ко-многим
  - (Никогда!) Циклическая

Альтернативы

- Бывает полезным (практически всегда – прим. преподавателя) иметь одновременно несколько различных абстракции одного явления для того, чтобы рассмотреть его с разных сторон (perspectives). Например:
  - Диаграмма классов
  - Диаграмма состояний
  - Диаграмма последовательности
- Альтернативы не образуют иерархию, но дополняют друг друга для более полного описания явления

PROGRAMMING FUNDAMENTALS

Основы программирования

- Процесс разработки программ описан (внезапно!) в ПИ
- Парадигмы программирования:
  - Неструктурированное
  - Структурированное (процедурное, императивное)
  - Функциональное
  - ООП
  - Аспектно-ориентированное (aspect oriented programming)

Aspect-oriented programming

- Строится поверх ООП
- Цель АОП – изолировать вторичные и/или обеспечивающие функции от основной бизнес-логики приложения
- Основная цель АОП – избежать запутывания, вносимого ООП в виде детального описания очень сложных взаимодействий объектов

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Классификация

- По «уровню» (абстрагированию от компьютерного представления)
  - Низкоуровневые
    - Пример: ассемблер
  - Высокоуровневые
    - Пример: C++, Java
- По отношению к способу инициации вычислений
  - Декларативные
    - Пример: Prolog, Lisp
  - Императивные

СРЕДСТВА И ТЕХНИКИ ОТЛАДКИ [КОДА]

Типы ошибок

- Синтаксические
  - Самые «простые» из видов ошибок – обнаруживаются компилятором. Нарушения синтаксиса могут «перерождаться» в логические ошибки
- Логические
  - Семантические ошибки, приводящие к неверным вычислениям или неправильному поведению ПО
  - Не отлавливаются компилятором. Часть ошибок может быть выявлена статическими анализаторами кода
- Ошибки данных
  - Ввод неверных данных (неправильного формата или «не тех» данных)

Отладка

- Статическая
  - Ревизия кода
- Динамическая
  - Трассировка выполнения
- После падения
  - Анализ core dump «упавшего» процесса
- Трассировка – выполнение кода с просмотром содержимого памяти (переменных, регистров, etc)
  - Пошаговое выполнение
  - Использование точек останова (breakpoints)
  - Отслеживание изменений переменных (watch points)

Средства отладки

- Отладчики
  - Используются для динамической отладки
  - Встроены во все современные IDE
  - Функции: отслеживание выполнения, запуск / перезапуск процесса, остановка выполнения, работа с точками останова и watch points, прямое редактирование переменных (памяти), иногда, сдвиг времени назад
- Анализаторы кода
  - Используются для статической отладки
- Просмотрщики core dump

СТРУКТУРЫ ДАННЫХ И ПРЕДСТАВЛЕНИЕ

Структуры данных

- Структуры данных это абстракции, предназначенные для сбора собственно данных и связанных с ними операций
- Часто, специфические структуры данных используют для увеличения эффективности программ и алгорит
- Often, data structures are designed for improving
- program or algorithm efficiency. Examples of
- such data structures include stacks, queues, and
- heaps. At other times, data structures are used for
- conceptual unity (abstract data type), such as the
- name and address of a person. Often, a data