Доклад на тему: «Методы программной инженерии в проектировании ИС»

по дисциплине: «Архитектура информационных систем».

Основные разделы

Методы программной инженерии в проектировании ИС

Методы обеспечивают проектирование, реализацию и выполнение ПО. Они накладывают некоторые ограничения на инженерию ПО в связи с особенностями применения их нотаций и процедур, а также обеспечивают оценку и проверку процессов и продуктов. Инструменты являются программной поддержкой отдельных методов инженерии ПО и обеспечивают автоматизированное выполнение задач процессов ЖЦ.

Область знаний "Методы и инструменты инженерии ПО (Software Engineering Tools and Methods)" состоит из разделов:

- инструменты инженерии ПО (Software Engineering Tools),
- методы инженерии ПО (Software Engineering Methods).

Методы инженерии

Методы инженерии ПО - это эвристические методы (*heuristic methods*), формальные методы (*formal methods*) и методы прототипирования (prototyping methods).

Эвристические методы включают:

- *Структурные методы* (structured methods), основанные на функциональной парадигме. При таком подходе системы строится с функциональной точки зрения, начиная с высокоуровневого понимания поведения системы с постепенным уточнением низкоуровневых деталей. (такой подход, иногда, также называют «проектированием сверху-вниз», прим. автора)
- Методы, ориентированные на структуры данных, которыми манипулирует ПО. Отправной точкой такого подхода являются структуры данных, которыми манипулирует создаваемое программное обеспечение. Функции в этом случае являются вторичными.
- Объектно-ориентированные методы, которые рассматривают предметную область как коллекцию объектов, а не функций; методы, ориентированные на конкретную область применения, например, на системы реального времени, безопасности и др.
- Методы, ориентированные на конкретную область применения (domain-specific methods). Такие специализированные методы разрабатываются с учетом специфики решаемых задач, например, систем реального времени, безопасности (safety) и защищенности (security).

Описание формального метода

Формальные методы основаны на формальных спецификациях, анализе, доказательстве и *верификации программ*. Формальная спецификация записывается на языке, синтаксис и семантика которого определены формально и основаны на математических концепциях (алгебре, теории множеств, логике). Различаются следующие категории формальных методов:

Формальные методы включают:

- *языки и нотации специфицирования* (specification languages and notations). Языки спецификаций могут быть ориентированы на модель, свойства и поведение. По мнению автора, ярким примером такого рода методов являются формальные методы описания требований, интерес к которым периодически возникает на протяжении всей истории программной инженерии
- *уточнение спецификации* (refinement specification); Данные подходы связаны с уточнением (трансформацией) превращения спецификаций в конечный результат, максимально близкий желаемому. В качестве результата применения таких методов рассматривается конечный - исполнимый программный продукт.
- *методы доказательства/верификации* (verification/*proving* properties), использующие утверждения (теоремы), пред- и постусловия, которые формально описываются и применяются для установления правильности *спецификации программ*.

История метода

История программной инженерии показала, что в области разработки прикладных систем, обоснованность (в частности, в силу трудоемкости) применения формальных методов не подтверждается на практике, за исключением случаев «скрытого» (неявного для разработчиков) применения определенных формальных методов на уровне внутренней реализации конкретных инструментов программной инженерии, например, в средствах моделирования и проектирования.

Эти методы применялись в основном в теоретических экспериментах и более 25 лет их практическое применение было ограничено из-за трудоемкости и экономической невыгодности. В 2005 г. проблема верификации приобрела вновь актуальность в связи с разработкой нового международного проекта по верификационному ПО "Целостный автоматизированный набор инструментов для проверки корректности ПС"

Перспективные задачи метода

- разработка единой теории построения и анализа программ;
- построение многостороннего интегрированного набора инструментов верификации на всех производственных процессах - разработка формальных спецификаций, их доказательство и проверка правильности, генерация программ и тестовых примеров, уточнение, анализ и оценка;
- создание репозитария формальных спецификаций, верифицированных программных объектов разных типов и видов.

Предполагается, что формальные методы верификации будут охватывать все аспекты создания и проверки правильности программ. Это приведет к созданию мощной верификационной производственной основы и значительному сокращению ошибок в ПО.

Методы прототипирования основаны на прототипировании ПО и подразделяются на:

- стили прототипирования, включающие в себя создание временно используемых прототипов (throwaway), эволюционное прототипирование - превращение прототипа в конечный продукт и разработка исполняемых спецификаций;
- Цели прототипирования. Примерами таких целей служат требования, архитектурный дизайн или пользовательский интерфейс
- техники оценки/исследования (evaluation) результатов прототипирования. Эти аспекты касаются того, как именно будут использованы результаты создания прототипа (например, будет ли он трансформирован в продукт, создается он для оценки нагрузочных способностей и других аспектов масштабируемости и т.п.)

Общее мнение по методам

Эти три темы не являются изолированными ,скорее они выделены исходя из их значимости и на основе определенных достаточно явных индивидуальных особенностей. Например, объектно-ориентированный подход может включать формально-технический и использовать прототипирование для проверки и аттестации. Так же как и инструменты, методы программной инженерии постоянно эволюционируют.

Инструменты инженерии ПО

Инструменты инженерии ПО обеспечивают автоматизированную поддержку процессов разработки ПО и включают множество разных инструментов, охватывающих все процессы ЖЦ.

Инструменты работы с требованиями (Software Requirements Tools) - это:

- инструменты разработки (*Requirement Development*) управления требованиями (*Requirement Management*) для анализа, сбора, специфицирования и проверки требований. Например, в модели CMMI *Staged* на 2-м уровне зрелости находится управлении требованиями, а на 3-м уровне - разработка требований;
- инструменты *трассировки требований* (Requirement *traceability* tools) являются неотъемлемой частью работы с требованиями, их функциональное содержание зависит от сложности проектов и уровня зрелости процессов.

Инструменты проектирования (Software Design Tools) - это инструменты для создания ПО с применением базовых нотаций (*SADT*/*IDEF*, UML, Microsoft DSL, Oracle и т.п.).

Инструменты конструирования ПО (Software Construction Tools) - это инструменты для производства, трансляции программ и машинного выполнения. К ним относятся:

- редакторы (*program editors*) для создания и модификации программ, и редакторы "общего назначения" (UNIX и UNIX-подобные среды);
- компиляторы и генераторы кода (compilers and *code generators*) как самостоятельные средства объединения в интегрированной среде программных компонентов для получения выходного продукта с использованием препроцессоров, сборщиков, загрузчиков и др.;
- интерпретаторы (interpreters) обеспечивают исполнение программ путем эмуляции, предоставляя для исполнения программ контролируемое и наблюдаемое окружение. Наметилась тенденотладчики (debuggers) для проверки правильности описания исходных программ и устранения ошибок;
- интегрированные среды разработки (IDE - integrated *developers environment*), библиотеки компонент (libraries components), без которых не может проводится процесс разработки ПС, программные платформы (Java, J2EE и Microsoft .NET) и платформа распределенных вычислений (CORBA и WebServices).

Инструменты тестирования (Software Testing Tools) это:

- генераторы тестов (*test generators*), помогающие в разработке сценариев тестирования;
- средства выполнения тестов (*test execution* frameworks) обеспечивают выполнение тестовых сценариев и отслеживают поведение объектов тестирования;
- инструменты оценки тестов (test evaluation tools) поддерживают оценку результатов выполнения тестов и степени соответствия поведения тестируемого объекта ожидаемому поведению;
- средства управления тестами (*test management tools*) обеспечивают инженерию процесса тестирования ПО;
- инструменты анализа производительности (*performance analysis* tools), количественной ее оценки и оценки поведения программ в процессе выполнения.

Инструменты сопровождения (Software Maintenance Tools) включают в себя:

- инструменты облегчения понимания (*comprehension* tools) программ, например, различные средства визуализации;
- инструменты реинжинерии (reengineering tools) поддерживают деятельность по реинжинерии и обратной инженерии (*reverse engineering*) для восстановления (артефактов, спецификация, архитектуры) стареющего ПО и генерации нового продукта.

Инструменты конфигурационного управления (Software Configuration Management Tools) - это:

- инструменты отслеживания (tracking) дефектов;
- инструменты управления версиями;
- инструменты управления сборкой, выпуском версии (конфигурации) продукта его инсталляции.

Инструменты управления инженерной деятельностью (Software Engineering Management Tools) состоят из:

- инструментов планирования и отслеживания проектов, количественной оценки усилий и стоимости работ проекта (Microsoft Project 2003);
- инструментов управления рисками используются для идентификации, мониторинга рисков и оценки нанесенного вреда;
- инструментов количественной оценки свойств ПО путем ведения измерений и расчета окончательного значения надежности и качества.

Инструменты поддержки процессов (Software Engineering Process Tools) разделены на:

- инструменты моделирования и описания моделей ПО (например, UML и его инструменты);
- инструменты управления программными проектами (Microsoft Project 2003);
- инструменты управления конфигурацией для поддержки версий и всех артефактов проекта.

Инструменты обеспечения качества (Software Quality Tools) делятся на две категории:

- инструменты инспектирования для поддержки просмотра (review) и аудита;
- инструменты статического анализа программных артефактов, данных, потоков работ и проверки свойств или артефактов на соответствие заданным характеристикам

Дополнительные аспекты инструментального обеспечения (Miscellaneous Tool Issues) соответствуют таким аспектам:

- техники интеграции инструментов (платформ, представлений, процессов, данных и управления) для естественного их сочетания в интегрированной среде
- метаинструменты для генерации других инструментов;
- оценка инструментов при их эволюции.

Итоги

Таким образом, метод программной инженерии — это структурный подход к созданию ПО, который способствует производству высококачественного продукта эффективным в экономическом аспекте способом.

В этом определении есть две основные составляющие:

- (а) создание высококачественного продукта и
- (б) экономически эффективным способом.

Иными словами, метод – это то, что обеспечивает решение основной задачи программной инженерии: создание качественного продукта при заданных ресурсах времени, бюджета, оборудования, людей.

Тест

1. Какого метода программной инженерии не существует?

a) Эвристического
b) Логического
c) Прототипирования
d) Формального

Тест

2. По каким причинам формальный метод ограничивался и не использовался на практике долгое время?

a) Он был слишком трудоемок
b) Он был слишком невыгоден экономически
c) Он затрагивал много ресурсов
d) Все варианты верны

Тест

3. Какие инструменты предоставляют отладку (отслеживание дефектов)?

a) Инструменты тестирования

b) Инструменты проектирования

c) Инструменты конфигурационного управления

d) Инструменты сопровождения

Тест

4. Какого подраздела в методе прототипирования не существует?

a) Задачи прототипирования
b) Цели прототипирования
c) Стили прототипирования
d) Оценки результатов прототипирования

Тест

5. Какие два основных составляющих в определении метода программной инженерии?

a) создание высококачественного продукта
b) создание дешевого продукта
c) создание экономически эффективным способом
d) создание наиболее быстрым способом
e) все вышеперечисленное