Основы программной инженерии

Казанский федеральный
УНИВЕРСИТЕТ

Тема 1. Предмет и основные принципы программной инженерии

Матренина Ольга Михайловна
Ст.преподаватель КАДиТП ИВМиИТ КФУ

https://kpfu.ru/main?p_id=47391
OMMatrenina@kpfu.ru
+7 (843) 233-77-60

Предмет и основные принципы программной инженерии

- Основные понятия и определения программной инженерии
- История становления программной инженерии
- Современная программная инженерия как наука и отрасль практической деятельности
- Программное обеспечение (software) и программные продукты. Классификация ПО
- Стандарты программной инженерии
- Области знаний программной инженерии. SWEBOK
- Методы программной инженерии
- Инструменты программной инженерии. CASE-инструменты

Программная инженерия (Software engineering)

- инженерная дисциплина, охватывающая все аспекты создания **программных продуктов**
- применение систематизированного, научного и рассчитываемого подхода к созданию, функционированию и сопровождению **программного обеспечения** (стандарт IEEE Std 610.12-1990)

История становления программной инженерии

1960-70 гг. – первый кризис программирования. Стоимость программ приблизилась к стоимости оборудования, а сроки разработки возрастали

1968 г. – закрытая конференция НАТО по проблемам «кризиса» разработки ПО, впервые звучит термин “Software Engineering”

1970-80 гг. – формирование целостного взгляда на Программную инженерию как область профессиональной деятельности; формирование терминологии, областей знаний, развитие стандартов, методов и инструментов

1980–90 гг. - начало информационно-технологической революции, вызванной взрывным ростом использования информационных средств: персональный компьютер, локальные и глобальные вычислительные сети, мобильная связь, электронная почту, Internet, ...;
возрастает конкуренция, сложность и количество заказов на программы

Настоящее время – развитие стандартов, инструментов и методов программной инженерии

Современный взгляд на эволюционные периоды разработки ПО (The Standish Group – CHAOS Report 2020) и перспективы развития менеджмента IT-проектов

- Первый период **«the Wild West»**, примерно с 1960 по 1980 год
- Второй период **«Waterfall»** - с 1980 по 2000 год
- Третий период **«Agile»** начался примерно в 2000 году, и, по прогнозам, он скоро закончится

- Четвертый период **«Infinite Flow»** (бесконечного потока) в перспективе продлится не менее 20 лет
  - нет бюджетов проектов, планов проектов, менеджеров проектов или мастеров Scrum
  - будет бюджет pipeline, который представляет собой чистые прямые затраты на продукцию
  - будут расходы на управление конвейером, что снизит текущие накладные расходы по проектам на 90%

    Это будет достигнуто за счет сокращения и ликвидации большей части текущей деятельности по управлению проектами. Функциональное описание работы войдет в конвейер и будет полностью пригодным для использования.

    Изменения будут происходить непрерывно, но с небольшими приращениями, чтобы все было актуальным, полезным и более приемлемым для пользователей, а не пугало их результатом «большого взрыва»

Критерии успешности проектов

- **Полнота и качество решения**
  - Выполнен весь объем работ
  - Реализованы все необходимые фичи
  - Все возможности реализованы с надлежащим качеством
- **Время**
  - Проект реализован вовремя
  - Каждый этап проекта реализован вовремя
- **Бюджет**
  - Проект уложился в планируемый бюджет

Статистика успешности IT-проектов

https://standishgroup.myshopify.com/collections/frontpage/products/copy-of-chaos-report-beyond-infinity-digital-version

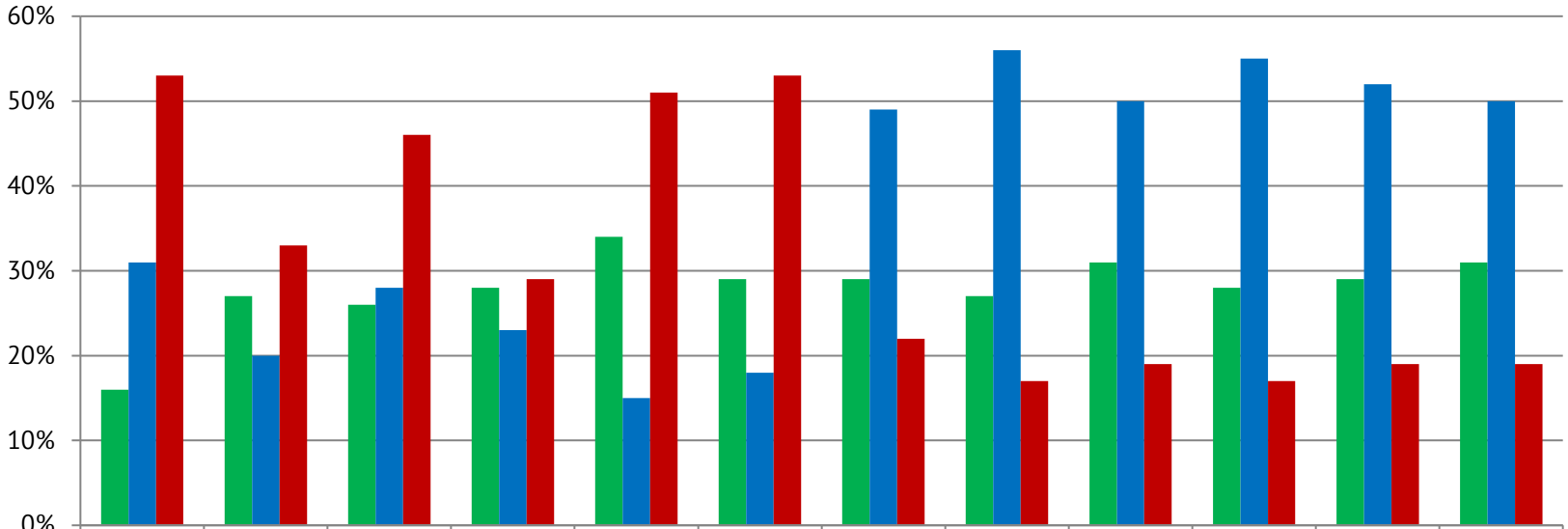
В **1994 г.** в США из 30000 проектов:

- **16,2% успешные** (завершились в срок, не превысили запланированный бюджет и реализовали все требуемые функции)
- **52,7% проблемные** (завершились с опозданием, расходы превысили запланированный бюджет, требуемые функции не были реализованы в полном объеме, качество получаемого программного обеспечения не устраивало потребителей)
- **31,1% провальные** (были аннулированы до завершения)

В **2020 г.**:

- **31% успешные**
- **50% проблемные**
- **19% провальные**

Статистика завершенных IT-проектов



	1994	1996	1998	2000	2002	2004	2011	2012	2013	2014	2015	2020
Успешные	16%	27%	26%	28%	34%	29%	29%	27%	31%	28%	29%	31%
Спорные	31%	20%	28%	23%	15%	18%	49%	56%	50%	55%	52%	50%
Провальные	53%	33%	46%	29%	51%	53%	22%	17%	19%	17%	19%	19%

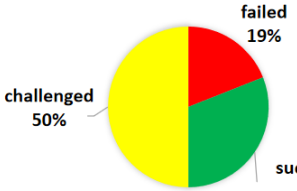
https://standishgroup.myshopify.com/collections/frontpage/products/copy-of-chaos-report-beyond-infinity-digital-version

Факторы успешности IT-проектов

https://standishgroup.myshopify.com/collections/frontpage/products/copy-of-chaos-report-beyond-infinity-digital-version

Project Success
Quick Reference Card

Based on CHAOS 2020: Beyond Infinity Overview. January'2021, QRC by Henny Portman



Modern measurement
(software projects)

Good Sponsor, Good Team, and Good Place are the only things we need to improve and build on to improve project performance.

The Good Place is where the sponsor and team work to create the product. It's made up of the people who support both sponsor and team. These people can be helpful or destructive. It's imperative that the organization work to improve their skills if a project is to succeed. This area is the hardest to mitigate, since each project is touched by so many people. Principles for a Good Place are:

- The Decision Latency Principle
- The Emotional Maturity Principle
- The Communication Principle
- The User Involvement Principle
- The Five Deadly Sins Principle
- The Negotiation Principle
- The Competency Principle
- The Optimization Principle
- The Rapid Execution Principle
- The Enterprise Architecture Principle

Successful project Resolution by Good Place Maturity Level:

highly mature	50%
mature	34%
moderately mature	23%
not mature	23%

The Good Team is the project's workhorse. They do the heavy lifting. The sponsor breathes life into the project, but the team takes that breath and uses it to create a viable product that the organization can use and from which it derives value. Since we recommend small teams, this is the second easiest area to improve. Principles for a Good Team are:

- The Influential Principle
- The Mindfulness Principle
- The Five Deadly Sins Principle
- The Problem-Solver Principle
- The Communication Principle
- The Acceptance Principle
- The Respectfulness Principle
- The Confrontationist Principle
- The Civility Principle
- The Driven Principle

Successful project Resolution by Good Team Maturity Level:

highly mature	66%
mature	46%
moderately mature	21%
not mature	1%

The Good Sponsor is the soul of the project. The sponsor breathes life into a project, and without the sponsor there is no project. Improving the skills of the project sponsor is the number-one factor of success – and also the easiest to improve upon, since each project has only one.
Principles for a Good Sponsor are:

- The Decision Latency principle
- The Vision Principle
- The Work Smart Principle
- The Daydream Principle
- The Influence Principle
- The Passionate Principle
- The People Principle
- The Tension Principle
- The Torque Principle
- The Progress Principle

Successful project Resolution by Good Sponsor Maturity Level:

highly mature	67%
mature	33%
moderately mature	21%
not mature	18%

Что влияет на успешность проекта?

- **Заказчик**
  - Заинтересованность в проекте
  - Понимание целей
  - Готовность к коммуникациям
  - Бюджет
  - …
- **Разработчики**
  - Квалификация участников команды
  - Инфрастуктура
  - Качество организации процессов разработки
- **Решаемая задача**
  - Масштаб
  - Сложность
  - Типовая ли задача или нестандартная
- **Прочие факторы**

Особенности индустрии ПО и типичных проблемы в реализации IT-проектов

- *нематериальность создаваемых продуктов* и неприменимость традиционных инженерных методов и практик к организации процесса разработки
- *незрелость собственно процессов разработки ПО*;
  трудности в процессах планирования и управления разработкой ПО
- *отставание методов разработки и инструментов* от быстрого прогресса в технологиях аппаратных платформ
- *размытые границы проектов*, работа в условиях неопределенных или изменяющихся требований к продукту
- *недостаток квалифицированных кадров*
- *разобщенность и разнородность отдельных групп разработчиков* по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств
- *зависимость от качества коммуникации* внутри команды и с заказчиком
- *временная протяженность проекта*, обусловленная сложностью проекта, ограниченными возможностями коллектива разработчиков, масштабами организации-заказчика и различной степенью готовности подразделений заказчика к внедрению программной системы

**Многие программные проекты не укладывается во время и в бюджет,
идеальный проект – скорее исключение, а не правило**

Больше проект – больше проблем

Размер проекта	Разработчики	Время (месяцы)	% успеха
Менее $750K	6	6	55%
От $750K до $1.5M	12	9	33%
От $1.5M до $3M	25	12	25%
От $3M до $6M	40	18	15%
От $6M до $10M	+250	+24	8%
Более $10M	+500	+36	0%

[данные SEI, 2000 г.]

Сложность современных ПП **– один разработчик не в силах охватить все аспекты системы, ее сложность превышает возможности человеческого интеллекта.**

Сложные программные системы требуют усилий коллектива специалистов для *разработки и сопровождения*, а также имеют тенденцию к *эволюции* в процессе их использования.

Особенности индустрии ПО и типичных проблемы в реализации IT-проектов

Особенности, обусловленные сложностью современных ПП

- *сложность реальных предметных областей* из которых приходит заказ на разработку
- *сложность описания программной системы,* обусловленная большим количеством функций, процессов, элементов данных и сложными взаимосвязями между ними;
  недостаточная проработанность способов описания поведения крупных дискретных систем
- *отсутствие полных аналогов,* ограничивающее возможность использования типовых проектных решений и прикладных систем при создании новой программной системы
- *наличие (как правило) в программной системе совокупности тесно взаимодействующих подсистем,* имеющих локальные задачи и цели функционирования, которые могут быть противоречивыми
- *необходимость интеграции* существующих и вновь разрабатываемых программных систем и приложений
- *функционирование сложной программной системы в неоднородной среде* на нескольких аппаратных платформах и в различных операционных средах, часто трудно совместимых

Программное обеспечение и программные продукты

- **Компьютерная программа** — это набор инструкций,
  которые могут быть выполнены компьютером для решения определенной задачи

- **Программное обеспечение (ПО)** — это набор компьютерных программ,
  процедур и связанной с ними документации и данных,
  необходимых для эксплуатации этих программ (ISO/IEC 12207 - ЖЦ ПО);
  совокупность программ системы обработки информации и программных документов, необходимых для их эксплуатации (ГОСТ 19.781-90 ЕСПД)

- **Программный продукт (ПП)** — это ПО, поставляемое конечному потребителю:
  совокупность отдельных программ,
  их документации,
  гарантий качества,
  рекламных материалов,
  мер по обучению пользователей,
  распространению и сопровождению этих программ
  …

Классификация ПО по функциональности (потребительский подход)

Программный продукт – ПО, поставляемое конечному потребителю.

«Коробочное» ПО

создается компанией по производству ПО по ее же инициативе,
продается на открытом рынке,
продается любому потребителю

- Ориентировано на рынок в целом
- Задачу формулирует разработчик
- Риски выше у исполнителя

Индивидуальное ПО «на заказ»

создается по инициативе заказчика,
пользоваться будет только заказчик,
возможна разработка
- или собственным IT-отделом а не софтверной компанией
- или согласно контракту, с учетом специфики заказчика

- Ориентировано на конкретного заказчика
- Задачу ставит заказчик
- Риски выше у потребителя

Другие способы классификации ПО

- **по целевому назначению**
  - встроенное ПО (software-embedded systems), системы реального времени (real-time systems) или социотехнические системы (sociotechnical systems)
  - программно насыщенные ПС (software-intensive systems) - манипуляции большими массивами информации для поддержки решений или приобретения знаний
  - вычислительно-ориентированные ПС (computing-intensive systems) - решение трудных задач, моделирование сложных систем путем расчетов и имитации
  - общего назначения – решение повседневных задач работы с информацией широкого круга пользователей
- **по доступности исходного кода**
  - открытый код (open source)
  - закрытый код (closed)
- **по типу лицензии**
  - проприетарные, частные (proprietary software)
  - свободные (libre software или free software)
- **по способу оплаты, ...**

Требования к хорошему ПО

Функциональность – ПО должно без чрезмерных усилий обеспечивать решение задач пользователя, для которых оно было разработано.

Сопровождаемость – ПО должно быть написано таким образом, чтобы оно могло развиваться, чтобы соответствовать изменяющимся требованиям пользователей. Включает наличие и понятность документации, ее соответствие исходному коду.

Надежность:

- защищенность от угроз
- отказоустойчивость – возможность восстановления
- безопасность – нет серьезных последствий при сбоях

Эффективность использования системных ресурсов, загрузки памяти, процессорного времени, сети и т.д.

Удобство (usability) – должно быть удобно для тех пользователей, на которых оно рассчитано.

Что такое стандарт?

англ. standard - норма, образец, мерило

- утверждаемый компетентным органом нормативно-технический документ, устанавливающий комплекс норм, правил по отношению к предмету стандартизации
  Пример: ГОСТ ЕСПД – единая система программной документации

- типовой образец, эталон, модель, принимаемые за исходные для сопоставления с ними других предметов
  Пример: эталоны мер и весов (эталон метра, Париж, палата мер и весов)

Что стандартизируют?

-материально-технические предметы (продукцию, эталоны, образцы веществ)
  Пример: электрическая розетка – вилка

- нормы, правила, требования организационно-методического и общетехнического характера
  Пример: Государственные образовательные стандарты для вузов

Какие бывают стандарты?

- **Корпоративные**
  Разрабатываются крупными фирмами для повышения качества своей продукции. На основе собственного опыта и с учетом требований мировых стандартов. Не сертифицируются, но обязательны внутри корпорации. Могут иметь закрытый характер.
- **Отраслевые стандарты**
  Действуют в пределах организаций некоторой отрасли (министерства). Разрабатываются с учетом мирового опыта и специфики отрасли. Являются обязательными для отрасли. Подлежат сертификации.
- **Государственные стандарты (ГОСТы)**
  Принимаются государственными органами, имеют силу закона. Разрабатываются с учетом мирового опыта или на основе отраслевых стандартов. Могут иметь как рекомендательный, так и обязательный характер (стандарты безопасности). Для сертификации создаются государственные или лицензированные органы сертификации.
- **Международные стандарты**
  Разрабатываются специальными международными организациями на основе мирового опыта и лучших корпоративных стандартов. Имеют рекомендатель-ный характер. Право сертификации получают организации (государственные и частные), прошедшие лицензирование в международных организациях.

Кто разрабатывает стандарты программной инженерии?

- **ISO** - International Organization for Standardization – Международная организация по стандартизации. Разрабатывает стандарты почти во всех областях деятельности, в том числе и в IT.
- **SEI** - Software Engineering Institute - Институт Программной Инженерии. Исследования в области программной инженерии с упором на разработку методов оценки и повышения качества ПО. Стандарты по качеству ПО и зрелости организаций, разрабатывающих ПО.
- **IEEE** - *Institute of Electrical and Electronics Engineers*) - Институт инженеров по электротехнике и электронике. Поддержка научных и практических разработок в электронике и вычислительной техники. Стандартизация.
- **IEC** - *International Electrotechnical Commission* - Международная электротехническая комиссия (МЭК). Некоммерческая организация по стандартизации в области электрических, электронных и смежных технологий
- **ACM** - Association for Computing Machinery –Ассоциация по вычислительной технике. Всемирная научная и образовательная организация в области вычислительной технике.
- **PMI** - Project Management Institute - Международный Институт Проектного Менеджмента. Некоммерческая организация, занимается стандартами управления проектами, повышением квалификации специалистов.

Основные стандарты программной инженерии

- SWEBOK - Software Engineering Body of Knowledge - Свод знаний по программной инженерии - содержит описания состава знаний по 10 разделам (областям знаний) программной инженерии
- ACM/IEEE CC2001 - Computing Curricula 2001 – Академический образовательный стандарт в области компьютерных наук
- ISO/IEC/IEEE 15288:2015(E) «Systems and software engineering — System life cycle processes» – является базовым стандартом системной инженерии. в России действует официальный перевод ГОСТ Р ИСО/МЭК 15288-2005.
- ISO/IEC 12207 - Information Technology - Software Life Cycle Processes - ГОСТ Р ИСО/МЭК 12270 - Процессы жизненного цикла программных средств. Стандарт содержит определения основных понятий программной инженерии (в частности программного продукта и жизненного цикла программного продукта), структуры жизненного цикла как совокупности процессов, детальное описание процессов жизненного цикла.
- SEI CMM - Capability Maturity Model (for Software) - модель зрелости процессов разработки программного обеспечения. ISO/IEC 15504 - Software Process Assessment - Оценка и аттестация зрелости процессов создания и сопровождения ПО. Является развитием и уточнением ISO 12207 и SEI CMM. Содержит расширенное по отношению ISO 12207 количество процессов жизненного цикла и 6 уровней зрелости процессов. Дается подробное описание схемы аттестации процессов, на основе результатов которой может быть выполнена оценка зрелости процессов и даны рекомендации по их усовершенствованию.
- PMBOK - Project Management Body of Knowledge - Свод знаний по управлению проектами. Содержит описания состава знаний по следующим 9 разделам (областям знаний) управления проектами
- ГОСТ 34. Информационная технология. Автоматизированные системы
- ГОСТ 19. Единая система программной документации

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 Програмной инженерии, экономика и описание профессиональных навыков по ПИ)

Основные области знаний программной инженерии

SWEBOK Guide V3.0

- **Software requirements** – программные требования
- **Software design** – проектирование ПО
- **Software construction** – конструирование программного обеспечения
- **Software testing** - тестирование
- **Software maintenance** – эксплуатация (поддержка) программного обеспечения

1. Ядро SWEBOK содержит пять основных областей знаний программной инженерии, 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.

Основные области знаний программной инженерии

SWEBOK Guide V3.0

Организационные области знаний программной инженерии

SWEBOK Guide V3.0

вспомогательные процессы и модели управления программными проектами

- **Software configuration management** – конфигурационное управление
- **Software engineering management** – управление в программной инженерии
- **Software engineering process** – процессы программной инженерии
- **Software Engineering Models and Methods** — модели и методы разработки
- **Software Engineering Professional Practice** — описание критериев профессионализма и компетентности

Области знаний ПИ, добавленные в стандарт V3.0

SWEBOK Guide V3.0

- **Software Quality** — качество ПО
- **Software Engineering Economics** — экономические аспекты разработки ПО
- **Computing Foundations** — основы вычислительных технологий, применимых в разработке ПО
- **Mathematical Foundations** — базовые математические концепции и понятия, применимые в разработке ПО
- **Engineering Foundations** — основы инженерной деятельности

Дополнительные области знаний

SWEBOK Guide V3.0

- **Computer engineering** — разработка компьютеров
- **Computer science** — информатика
- **Management** — общий менеджмент
- **Mathematics** — математика
- **Project management** — управление проектами
- **Quality management** — управление качеством
- **Systems engineering** — системное проектирование

Связь программной инженерии со смежными областями знаний

- Mathematics
- Computer engineering
- Computer science
- Systems engineering
- Management
- Project management
- Quality management

Методы программной инженерии

Методы ПИ (*англ. software engineering methods*) упорядочивают деятельность, связанную с программной инженерией, чтобы систематизировать действия по разработке ПО и увеличить их продуктивность.

- **эвристические методы (heuristic methods)**
- **формальные методы (formal methods)**
- **методы прототипирования (prototyping methods)**
- **гибкие методы (agile methods)**

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)***
- ***Инструменты проектирования (Software Design Tools)***
- ***Инструменты конструирования ПО (Software Construction Tools)***
- ***Инструменты тестирования (Software Testing Tools)***
- ***Инструменты сопровождения (Software Maintenance Tools)***
- ***Инструменты конфигурационного управления (Software Configuration Management Tools)***
- ***Инструменты управления инженерной деятельностью (SE Management Tools)***
- ***Инструменты поддержки процессов (Software Engineering Process Tools)***
- ***Инструменты обеспечения качества (Software Quality Tools)***
- ***Дополнительные аспекты инструментального обеспечения (Miscellaneous Tool Issues)***

CASE–инструменты *(Computer Aided Software Engineering)* – инструментальные программные средства, используемые для поддержки процесса создания ПО

Зачем их использовать?

- ошибок становится меньше
- качество ПО выше
- обслуживание проще

CASE-средства позволяют получить описание работы создаваемой системы раньше, чем ее построили.

С их помощью можно оптимизировать подготавливаемые решения, включая анализ требований к системе, проектирование прикладного программного обеспечения и баз данных, генерацию кода, тестирование, обеспечение качества, управление проектом, а также другие процессы.

Особенности современных CASE-средств:

- наличие мощных графических средств для описания и документирования системы, обеспечивающих удобный интерфейс с разработчиком и развивающих его творческие возможности;
- интеграция отдельных компонентов CASE-средств, обеспечивающая управляемость процессом разработки программного обеспечения;
- использование специальным образом организованных хранилищ проектных метаданных (репозитории)

Классификация Case-инструментов:

- по уровню применения – показывает отношение к стадиям и этапам разработки ПО
- по функциям – отражает внутреннюю ориентацию на те или иные процессы ЖЦ
- по категориям – степень интегрирования по выполняемым функциям

Классификация CASE–средств по уровню применения

- средства анализа предметной области проекта и поддержки требований к проектируемой системе (Upper CASE);
- средства проектирования (Middle CASE);
- средства разработки приложений (Low CASE);
- интегрированные CASE-средства, охватывающие несколько этапов создания ПО – от анализа требований до тестирования и выпуска эксплуатационной документации.

Классификация CASE-средств по выполняемым функциям

Тип CASE-средства	Примеры
Средства планирования	Средства системы PERT*, средства оценивания, электронные таблицы
Средства редактирования	Текстовые редакторы, редакторы диаграмм, тестовые процессоры
Средства управления изменениями	Средства оперативного контроля за требованиями, системы управления изменениями
Средства управления конфигурацией	Системы управления версиями ПО, средства построения систем
Средства прототипирования	Языки программирования самого высокого уровня, генераторы пользовательских интерфейсов
Средства, ориентированные на поддержку определенных методов	Редакторы системных структур, словари данных, генераторы программного кода
Средства конструирования программ	Компиляторы, интерпретаторы
Средства анализа программ	Генераторы перекрестных ссылок, статические и динамические анализаторы программ
Средства тестирования	Генераторы тестовых данных, компараторы файлов
Средства отладки	Интерактивные средства отладки
Средства документирования	Программы разметки страниц, редакторы изображений, генераторы отчетов
Средства реинжиниринга	Восстановление ER-моделей по результатам анализа схем баз данных; восстановление диаграмм классов на основе анализа исходных кодов программ

Классификация CASE-средств по категориям

- *Вспомогательные программы,* поддерживающие отдельные процессы разработки ПО: проверка непротиворечивости архитектуры системы, компиляция программ, сравнение результатов тестов и т.п. Вспомогательные программы могут быть универсальными функционально-законченными средствами или входить в состав инструментальных средств.
- *Инструментальные средства* поддерживают определенные процессы разработки ПО, например, специфицирование требований, проектирование и т.д. Обычно инструментальные средства являются набором вспомогательных программ, которые в большей или меньшей степени интегрированы.
- *Рабочие среды разработчика* поддерживают все или большинство процессов разработки ПО. Рабочие среды обычно включают в себя несколько различных интегрированных инструментальных средств.

Тема 1. Предмет и основные принципы программной инженерии

Спасибо за внимание!

Матренина Ольга Михайловна
Ст.преподаватель КАДиТП ИВМиИТ КФУ

https://kpfu.ru/main?p_id=47391
OMMatrenina@kpfu.ru
+7 (843) 233-77-60

Организационные области знаний программной инженерии

SWEBOK Guide V3.0