Руководство проектом

Лекция 2

«Если водитель трамвая начнет искать новые пути, жди беды»

- Классическое управление проектами выделяет два вида организации человеческой деятельности: *операционная* и *проектная*
- Операционная деятельность представляет собой продолжающийся во времени и повторяющийся процесс
- Задача операционной деятельности – обеспечение нормального течения бизнеса
- В этом случае основой эффективности служат узкая специализация и повышение компетенции

Проект – основа инноваций

- Условия применимости проектной деятельности:
  - разрабатывается новый продукт,
  - внешние условия и требования к продукту постоянно меняются,
  - применяемые производственные технологии используются впервые,
  - постоянно требуются поиск новых возможностей, интеллектуальные усилия и творчество

Проект – основа инноваций

- *Проект* – это временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов
- Проекты являются временными и уникальными
- Задача проекта – достижение конкретной бизнес-цели

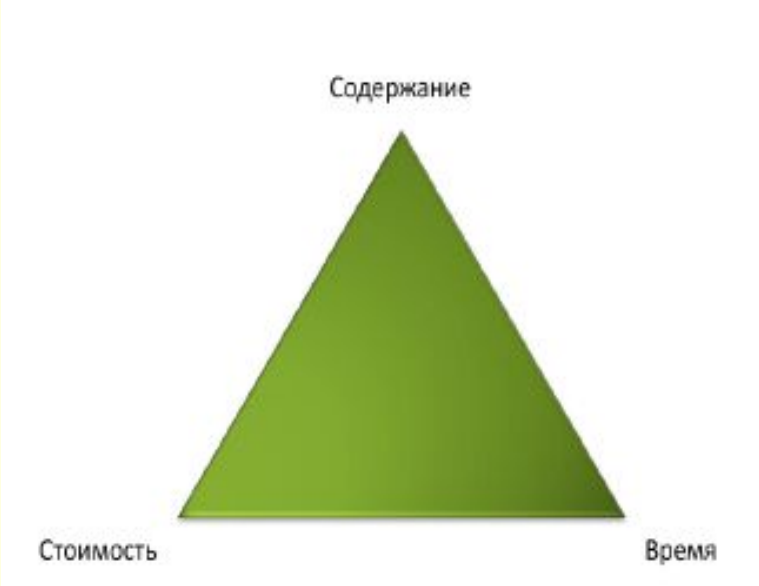
Проект – средство стратегического развития

Проект – средство стратегического развития

- Цель – описание того, что мы хотим достичь
- Стратегия – констатация того, каким образом мы собираемся эти цели достигать
- *Проекты преобразуют стратегии в действия, а цели в реальность*
- Проекты объединяются в программы; *программа* – ряд связанных друг с другом проектов
- Проекты и программы объединяются в портфели
- *Портфель* – набор проектов или программ и других работ, объединенных вместе с целью эффективного управления

Критерии успешности проекта

- Задача проекта — достижение конкретной бизнес-цели, при соблюдении ограничений «железного треугольника»: ни один из углов треугольника не может быть изменен без оказания влияния на другие



Критерии успешности проекта

- Проект считается успешным, если:
  - выполнен в соответствие со спецификациями,
  - выполнен в срок,
  - выполнен в пределах бюджета,
  - *каждый участник команды уходил с работы в 18:00 с чувством успеха*
- *Нет ничего более гибельного для проекта, чем равнодушие или уныние его участников*

Организация проектной команды

- Каждый проект разработки ПО имеет свою организационную структуру, которая определяет распределение ответственности и полномочий среди участников проекта, а также обязанностей и отношений отчетности
- *Чем меньше проект, тем больше ролей приходится совмещать одному исполнителю.*

Роли и ответственности

- Роли и ответственности участников типового проекта разработки ПО можно условно разделить на пять групп:
  - **Анализ.** Извлечение, документирование и сопровождение требований к продукту.
  - **Управление.** Определение и управление производственными процессами.
  - **Производство.** Проектирование и разработка ПО.
  - **Тестирование.** Тестирование ПО.
  - **Обеспечение.** Производство дополнительных продуктов и услуг

Группа анализа

- Включает в себя следующие роли:
  - **Бизнес-аналитик.** Построение модели предметной области (онтологии).
  - **Бизнес-архитектор.** Определяет общее видение продукта.
  - **Системный аналитик.** Отвечает за перевод требований к продукту в функциональные требования к ПО.
  - **Специалист по требованиям.** Документирование и сопровождение требований к продукту.
  - **Менеджер продукта.** Представляет в проекте интересы пользователей продукта

Группа управления

- Состоит из следующих ролей:
  - **Руководитель проекта.** Отвечает за достижение целей проекта при заданных ограничениях.
  - **Куратор проекта.** Оценка планов и исполнения проекта. Выделение ресурсов.
  - **Системный архитектор.** Разработка технической концепции системы.
  - **Руководитель группы тестирования.** Определение целей и стратегии тестирования.
  - **Ответственный за управление изменениями**, конфигурациями, за сборку и поставку программного продукта

Производственная группа

- В ее состав входят:
  - **Проектировщик.** Проектирование компонентов и подсистем в соответствие с общей архитектурой, разработка архитектурно значимых модулей.
  - **Проектировщик базы данных**.
  - **Проектировщик интерфейса пользователя**.
  - **Разработчик.** Проектирование, реализация и отладка отдельных модулей системы.
- В большом проекте может быть несколько производственных групп, ответственных за отдельные подсистемы

Группа тестирования

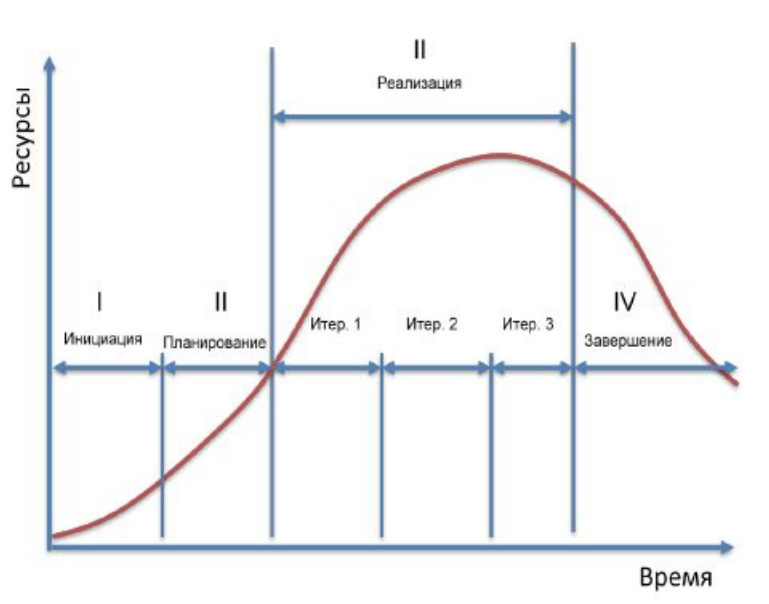
- Состоит из следующих ролей:
  - **Проектировщик тестов.** Разработка тестовых сценариев.
  - **Разработчик автоматизированных тестов**.
  - **Тестировщик.** Тестирование продукта. Анализ и документирование результатов

Группа обеспечения

- Участники этой группы, как правило, не входят в команду проекта. Они выполняют работы в рамках своей процессной деятельности
- К группе обеспечения можно отнести следующие проектные роли:
  - Технический писатель.
  - Дизайнер графического интерфейса.
  - Разработчик учебных курсов, тренер.
  - Продажи и маркетинг.
  - Системный администратор.
  - Специалист по инструментальным средствам

Временное распределение ресурсов

- В операционной деятельности ресурсы расходуются равномерно по времени
- В проектном управлении расходование ресурсов в единицу времени имеет явно выраженное колоколообразное распределение



Начало проекта

- В компании, которая принимает решение о старте того или иного проекта разработки ПО, должна существовать единая система критериев, позволяющая из множества возможных для реализации проектов выбрать наиболее приоритетные для компании
- Приоритет любого проекта должен определяться на основе оценки трех его характеристик:
  - Финансовая ценность.
  - Стратегическая ценность.
  - Уровень рисков.

Шкала оценки финансовой ценности проекта

- Может выглядеть следующим образом:
  - *Высокая.* Ожидаемая окупаемость до 1 года. Ожидаемые доходы от проекта не менее чем в 1.5 раз превышают расходы. Все допущения при проведении этих оценок четко обоснованы.
  - *Выше среднего.* Ожидаемая окупаемость проекта от 1 года до 3 лет. Ожидаемые доходы от проекта не менее чем в 1.3 раза превышают расходы. Большинство допущений при проведении этих оценок имеют под собой определенные основания.

Шкала оценки финансовой ценности проекта

- *Средняя.* Проект позволяет улучшить эффективность производства в компании и потенциально может снизить расходы компании не менее чем на 30%. Проект может иметь информационную ценность или помочь лучше контролировать бизнес.
- *Низкая.* Проект снижает расходы компании не менее чем на 10% и дает некоторые улучшения производительности производства

Шкала оценки стратегической ценности

- Может иметь следующий вид:
  - *Высокая.* Обеспечивает стратегическое преимущество, дает устойчивое увеличение рынка или позволяет выйти на новый рынок. Решает значительные проблемы, общие для большинства важных клиентов. Повторение конкурентами затруднено или потребует от 1 до 2 лет.
  - *Выше среднего.* Создает временные конкурентные преимущества. Выполнение обязательств перед многими важными клиентами. Конкурентное преимущество может быть удержано в течение 1 года.

Шкала оценки стратегической ценности

- *Средняя.* Поддерживается доверие рынка к компании. Повышает мнение клиентов о качестве предоставляемых услуг или способствует выполнению обязательств перед несколькими клиентами. Конкуренты уже имеют или способны повторить новые возможности в пределах года.
- *Низкая.* Стратегическое воздействие отсутствует или незначительно. Влияние на клиентов несущественно. Конкуренты могут легко повторить результаты проекта

Шкала оценки уровня рисков проекта

- Может иметь следующий вид:
  - *Низкий.* Цели проекта и требования хорошо поняты и документированы. Масштаб и рамки проекта заданы четко. Ресурсы требуемой квалификации доступны в полном объеме. Разрабатываемые системы не потребуют новой технологической платформы.
  - *Средний.* Цели проекта определены более-менее четко. Хорошее понимание требований к системе. Масштаб и рамки проекта заданы достаточно хорошо. Ресурсы требуемой квалификации доступны в основном. Системы создаются на новой, но стабильной технологической платформе.

Шкала оценки уровня рисков проекта

- *Выше среднего.* Цели проекта недостаточно четки. Задачи системы или бизнес-приложения поняты недостаточно полно. Понимание масштаба и рамок проекта недостаточно. Ресурсы требуемой квалификации сильно ограничены. Системы создаются на новой технологической платформе.
- *Высокий.* Цели проекта нечетки. Основные функциональные компоненты системы не определены. Масштаб и рамки проекта непонятны. Ресурсы требуемой квалификации практически отсутствуют. Системы создаются на новой технологической платформе. Технологии имеют неподтвержденную стабильность.

Начало проекта

- В начале работы над проектом необходимо:
  - установить цели и проблемную область проекта
  - рассмотреть и обсудить возможные варианты решения
  - выявить технические, организационные и материальные ограничения

Процесс оценки

- Оценка объема предстоящих работ производится в человеко-месяцах
- Исходя из нее определяют необходимые ресурсы:
  - продолжительность работ (в календарных датах)
  - число разработчиков
  - стоимость (в тыс. рублей)
- При оценках используется опыт предыдущих собственных разработок, либо опыт других разработчиков

Анализ рисков

- Риском называется возможная потеря в процессе разработки. Это может быть потеря качества продукта, рост затрат на разработку, отставание от графика и т.д.
- Задачей руководства является анализ возможных рисков и планирование мер по минимизации их влияния на выполнение разработок.

Пример проверочного списка элементов риска

№ п/п	Элемент риска	Вероят-ность (%)	Потери (ч-д)	Влияние риска
1	Дефицит персонала	5-7	10	50-70
2	Нереальные расписание и бюджет	8	8	64
3	Разработка неправильных функций и характеристик	4	7	28
4	Разработка неправильного пользовательского интерфейса	8	3	24
5	Интенсивный поток изменения требований	15	4	60
6	Дефицит поставляемых компонентов	10	4	40
7	Недостатки в задачах, разрабатываемых смежниками	10	3	30
8	Недостаточная производительность продукта	6	3	18

Ранжирование рисков

- Ранжирование заключается в назначении каждому риску приоритета в соответствии со степенью его влияния на проект
- Цель ранжирования – выделение наиболее значимых рисков
- Планирование управления рисками заключается в определении набора организационно-технических мероприятий, имеющих целью уменьшение основных рисков

Планирование процесса разработки

- Основная задача планирования – декомпозиция проекта и представление его в виде дерева, в корне которого находится проект, а на листьях элементарные задачи или работы, которые надо выполнить, чтобы завершить проект в условиях заданных ограничений

Планирование процесса разработки

- *Иерархическая структура работ (ИСР) (Work /Breakdown Structure, WBS)* – это ориентированная на результат иерархическая декомпозиция работ, выполняемых командой проекта для достижения целей проекта и необходимых результатов
- С ее помощью структурируется и определяется все содержание проекта; каждый следующий уровень иерархии отражает более детальное определение элементов проекта

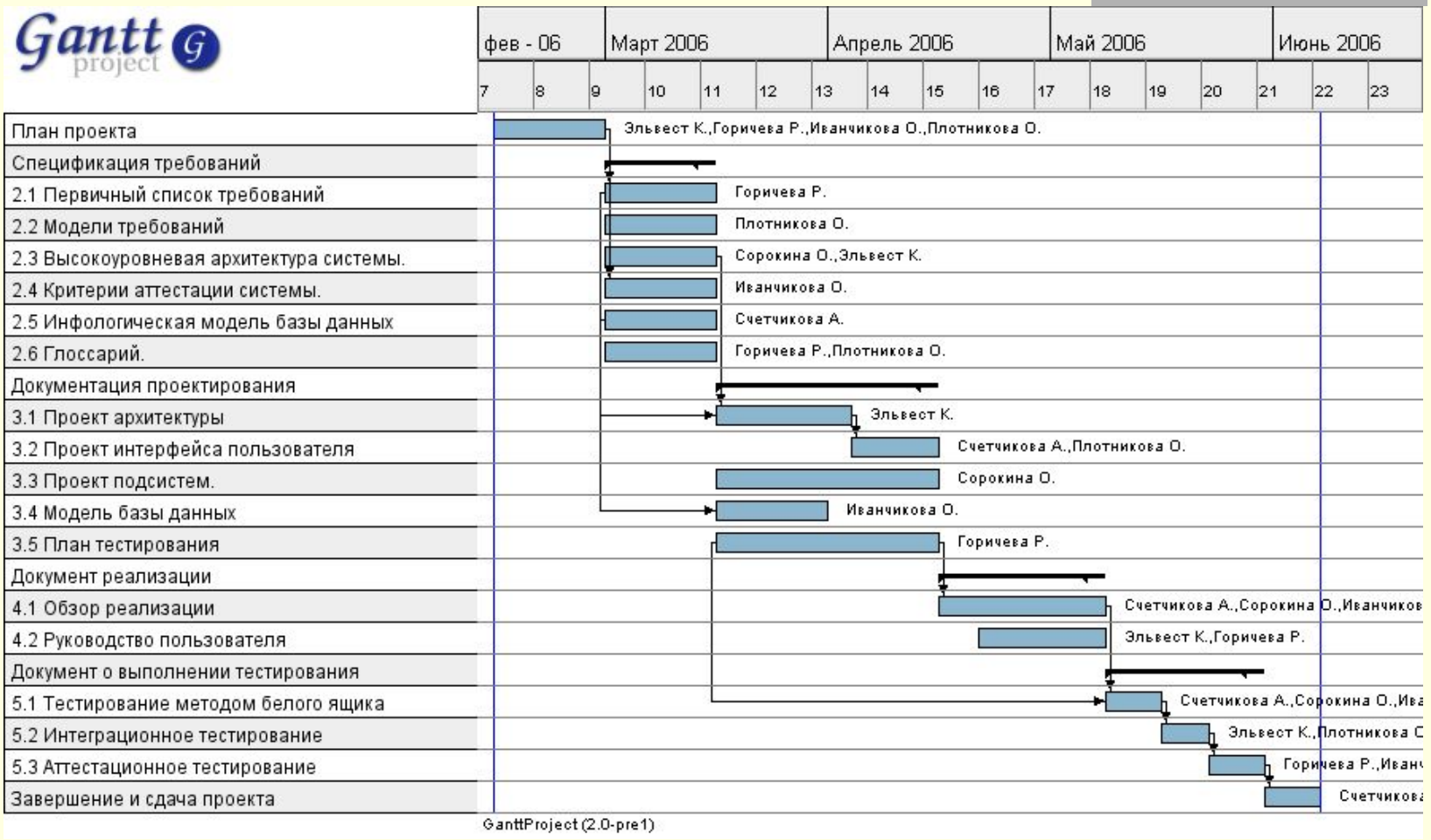
Выделение компонентов

- Выделение компонентов, составляющих программный продукт, это элемент высокоуровневого проектирования, которое мы должны выполнить на фазе планирования проекта
- Компонентами могут быть как прикладные подсистемы, так и инфраструктурные или ядерные, например, подсистема логгирования, безопасности, библиотека визуальных компонентов GUI

Базовое расписание проекта

- Базовое расписание — утвержденный план-график с указанными временными фазами проекта, контрольными точками и элементами иерархической структуры работ.
- Базовое расписание может быть наиболее наглядно представлено *диаграммой Ганта,* содержащей плановые операции или элементы иерархической структуры работ, даты начала и завершения и длительность операций

Диаграмма Ганта



Границы времени выполнения

- Распараллеливание задач требует согласования процессов их выполнения во времени. Для каждой из них должно быть запланировано приемлемое время решения T_{proc}, а также раннее T_{min} и позднее T_{max} время начала решения
- Необходимо выделить задачи, образующие основу проекта, и определяющие временные рамки его выполнения

Распределение времени выполнения

- Рекомендуемое распределение времени выполнения проекта:
  - на анализ и проектирование 40% временных затрат (из них 5% на анализ и планирование)
  - на кодирование – 20%
  - на тестирование и отладку – 40%

Выбор модели разработки

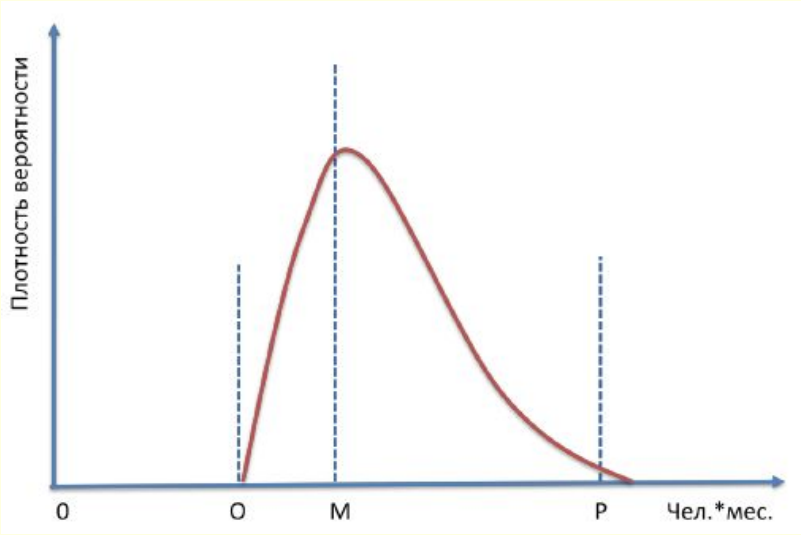
- Реальный процесс разработки никогда жестко не увязывается с какой-либо одной моделью, хотя одна из них может быть ведущей
- Выбор модели определяется:
  - объемом и сложностью проекта;
  - количеством и качеством команды разработчиков
  - квалификацией заказчика, его способностью обеспечить достаточно четкую постановку задачи
- Как правило, процессы разработки сочетают в себе элементы инкрементального и эволюционного подходов

Оценка объема и сложности проекта

- Первое, что необходимо понимать при оценке проекта – это то, что любая оценка всегда является вероятностным утверждением
- Если просто сказать, что трудоемкость данного пакета работ составляет **M** чел.*мес., то это будет плохой оценкой, т.к. одно единственное число ничего не говорит о вероятности того, что на реализацию этого пакета потребуется не более, чем **M** чел.*мес.

Оценка объема и сложности проекта

- Вероятностный характер оценки, означает, что для нее существует некоторое распределение вероятности, которое может быть очень широким (высокая неопределенность) или достаточно узким (низкая неопределенность)



Оценки на основе метрик

- Для оценки различных свойств процесса создания программного продукта, а также и самого продукта, используются комплексные и достаточно сложные методики, основанные на применении количественных характеристик, называемых *метриками*
- Метрики – это функций от так называемых *опорных значений,* получаемых путем непосредственного измерения

Оценки на основе метрик

- Метрики используются для получения предварительной, текущей и итоговой оценок:
  - экономических параметров проекта трудоемкости, длительности, стоимости;
  - оценка рисков по проекту: риска нарушения сроков и невыполнения проекта, риска увеличения трудоемкости на этапах отладки и сопровождения проекта и пр.
- На основе отслеживания метрик проекта можно своевременно предупредить возникновение нежелательных ситуаций и устранить последствия непродуманных проектных решений

Классификация метрик

- Метрики сложности программ принято разделять на три основные группы:
  - метрики размера программ;
  - метрики сложности потока управления программ;
  - метрики сложности потока данных программ

Метрики размера программ

- Метрики этой группы базируются на определении количественных характеристик, связанных с размером программы, и отличаются относительной простотой
- К наиболее известным метрикам данной группы относятся число операторов программы, количество строк исходного текста, набор метрик Холстеда
- Метрики этой группы ориентированы на анализ исходного текста программ и могут использоваться для оценки сложности промежуточных продуктов разработки

Метрики потока управления

- Метрики сложности потока управления базируются на анализе управляющего графа программы
- Управляющий граф программы, может быть построен на основе алгоритмов модулей
- Поэтому метрики данной группы также могут применяться для оценки сложности промежуточных продуктов разработки.
- Представителем данной группы является метрика Маккейба

Метрики потока данных

- Метрики третьей группы базируются на оценке использования, конфигурации и размещения данных в программе
- В первую очередь это касается глобальных переменных
- К данной группе относятся метрики Чепина

Размерно-ориентированные метрики

- Основаны на LOC-оценках, т.е. на количестве строк в текстах программ (Lines Of Code). К числу размерно-ориентированных метрик относятся:
  - производительность
  - качество
  - удельная стоимость
  - документированность

Метрики производительности и качества

$$\text{Производительность} = \frac{\text{Длина [тыс. LOC]}}{\text{Затраты [чел.-мес.]}}$$

$$\text{Качество} = \frac{\text{Ошибки [Единиц]}}{\text{Длина [тыс. LOC]}}$$

Метрики стоимости и документированности

$$\text{Удельная Стоимость} = \frac{\text{Стоимость [Тыс. руб.]}}{\text{Длина [LOC]}}$$

$$\text{Документированность} = \frac{\text{Страниц Документа}}{\text{Длина [тыс. LOC]}}$$

Достоинства и недостатки

- Размерно-ориентированные метрики:
  - основаны на объективных данных
  - просты и легко вычислимы
- Недостатки:
  - зависят от языка программирования
  - трудновыполнимы на начальной стадии проекта
  - не приспособлены к непроцедурным языкам программирования
- Метод LOC является только оценочным методом, который надо принимать к сведению, но не опираться на его результаты в оценках

Метрики сложности кода

- Помимо показателей оценки объема работ по проекту очень важными для получения объективных оценок являются показатели сложности проекта
- Как правило, данные показатели не могут быть вычислены на самых ранних стадиях работы, поскольку требуют, как минимум, выполнения детального проектирования
- Однако эти показатели очень важны для получения прогнозных оценок длительности и стоимости проекта, поскольку непосредственно определяют его трудоемкость

Метрики Холстеда

- При вычислении этих метрик используются следующие опорные значения:
  - nu_1 – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);
  - nu_2 – число уникальных операндов программы (словарь операндов);
  - n_1 – общее число операторов в программе;
  - n_2 – общее число операндов в программе

Метрики Холстеда

- На основании этих опорных значений рассчитываются следующие метрики программы:
  - словарь $NU = nu_1 + nu_2$
  - длина $N = n_1 + n_2$
  - объем $V = N * Log_2 (NU)$
  - сложность $D = (nu_1 / 2) * (n_2 / nu_2)$
  - трудоемкость $E = D * V$

Метрика Мак-Кейба

- Показатель цикломатической сложности является одним из наиболее распространенных показателей оценки сложности программных проектов (Мак-Кейб, 1976 г.)
- Этот показатель вычисляется на основе графа управляющей логики программы (control flow graph)
- Данный граф строится в виде ориентированного графа, в котором вычислительные операторы или выражения представляются в виде узлов, а передача управления между узлами - в виде дуг

Метрика Мак-Кейба

- Упрощенная формула вычисления цикломатической сложности представляется следующим образом:

  C = e - n + 2,

где e - число ребер, а n - число узлов на графе управляющей логики.

- Практически цикломатическая сложность ПО равна числу предикатов плюс единица, что позволяет вычислять ее без построения управляющего графа простым подсчетом предикатов

Достоинства и недостатки метрики

- Достоинства метрики Мак-Кейба:
  - простоту вычисления и повторяемость результата,
  - наглядность и содержательность интерпретации
- Недостатки:
  - нечувствительность к размеру ПО,
  - нечувствительность к изменению структуры ПО,
  - отсутствие корреляции со структурированностью ПО,
  - отсутствие чувствительности к вложенности циклов
- Существует значительное число модификаций показателя цикломатической сложности

Метрика Чепина

- Суть метода состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода
- Все множество переменных, составляющих список ввода-вывода, разбивается на четыре функциональные группы

Функциональные группы

- Множество P - вводимые переменные для расчетов и для обеспечения вывода (переменные, содержащие исходную информацию)
- Множество M - модифицируемые или создаваемые внутри программы переменные.
- Множество C - переменные, участвующие в управлении работой программного модуля (управляющие переменные)
- Множество T - не используемые в программе ("паразитные") переменные

Формула метрики Чепина

- Поскольку каждая переменная может выполнять одновременно несколько функций, необходимо учитывать ее в каждой соответствующей функциональной группе
- Далее вводится значение метрики Чепина:

  $Q = a_1 * P + a_2 * M + a_3 * C + a_4 * T,$

где a_1, a_2, a_3, a_4 - весовые коэффициенты

- Весовые коэффициенты использованы для отражения различного влияния на сложность программы каждой функциональной группы

Весовые коэффициенты

- По мнению автора метрики наибольший вес, равный трем, имеет функциональная группа C, так как она влияет на поток управления программы
- Весовые коэффициенты остальных групп распределяются следующим образом: $a_1=1$; $a_2=2$; $a_4=0.5$.
- Весовой коэффициент группы T не равен нулю, поскольку "паразитные" переменные не увеличивают сложности потока данных программы, но иногда затрудняют ее понимание

Формула Чепина

- С учетом весовых коэффициентов выражение для метрики Чепина примет вид:

  $Q = P + 2 * M + 3 * C + 0.5 * T$

Конец лекции

- В лекции использованы материалы сайта http://citforum.ru/SE/project/arkhipenkov_lectures/