Технология разработки программного обеспечения

Современные скорости

Если бы автомобиль прогрессировал так же быстро, как компьютер, “Роллс-Ройс” стоил бы сейчас меньше доллара, а на литре бензина можно было бы проехать тысячу километров.

Б. Гейтс

Доля ИТ в ВВП стран

Страна	Доля ИТ в ВВП,%
США	4,6
Великобритания	4,2
Япония	3,6
Франция	3,3
Германия	3,1
Чехия	2,8
Португалия	2
Индия	2
Испания	1,7
Россия	1,4

Источник: Eurostat, 2007

Доля ИТ в ВВП стран

** Без учета отрасли оптовой торговли ИКТ-товарами.*

Источник: ОЭСР, ИСИЭЗ НИУ ВШЭ, расчеты Института экономики роста.

(Россия и остальные страны – 2017 г. или ближайшие годы)

Серьезные деньги...

Жизненный цикл ПО

1. Анализ
2. Проектирование
3. Разработка
4. Тестирование
5. Развертывание
6. Использование

Жизненный цикл в Rational Unified Process

1. Проектирование
2. Разработка
3. Тестирование
4. Стабилизация
5. Внедрение

Задачи этапа анализа

- Составление списка требований к разрабатываемой системе и среде ее функционирования

Требования к системам реального времени и макросам, автоматизирующим офисные задачи, различны.

Задачи этапа анализа

- Ранжирование списка требований

- ✓ Реализация первоочередных требований
- ✓ Выявление несовместимых требований
- ✓ Требования, которыми можно пожертвовать в кризисной ситуации

Виды требований

1. Функционал системы
2. Качество реализации
3. Сроки реализации
4. Стоимость разработки

Задачи этапа проектирования

- Разбиение системы на отдельные модули
- Определение функциональности модулей

Виды проектирования

Сверху вниз

Плюсы: разработка именно необходимой системы

Минусы: стоимость решения в связи с выполнением полного цикла работ и создания уникальных технологий

Виды проектирования

Снизу вверх

Плюсы: стоимость разработки ниже за счет использования имеющихся решений

Минусы: возможны отклонения от начальной постановки задачи

Задачи этапа проектирования

- Фиксируются методы и методики, используемые для реализации намеченной функциональности
- Фиксируются необходимые для реализации проекта технологии

Задачи этапа проектирования

- Согласование и модификация требований
- Составление плана работ
- Составление финансового плана
- Составление бизнес-плана

Этап разработки

Задача программиста – реализовать систему с заданным качеством и в заданные сроки

Задача менеджера – проводить контроль всех параметров разработки и отслеживать их отклонения

Этап разработки

В начале работ необходимо провести:

- проектирование структуры ПО – структуры классов, структуры данных, алгоритмов функционирования;
- проектирование интерфейса;
- проектирование структуры документации
- проектирование тестов системы ...

Этап разработки

Программирование основной системы не является единственной задачей.

Инструментальные средства, графика, подготовка и ввод данных, написание документации, проведение тестов, ...

Тестирование проекта

Этап посвящен обнаружению и устранению ошибок, однако уровень качества проекта закладывается в процессе разработки

Ошибки в коде

Написанный – 0,5 – 3 ошибки на строку
Скомпилированный – 1 на 10 строк
Отлаженный – 1 на 100 строк
Протестированный – 1 на 1000 строк

Рекорд – 1 на 39000 строк в PostgreSQL

Ошибки в коде

Время на обнаружения

5-20 минут при разработке

15-30 минут при отладке и тестировании

8 часов при комплексном тестировании

10-40 часов при использовании

Виды тестирования

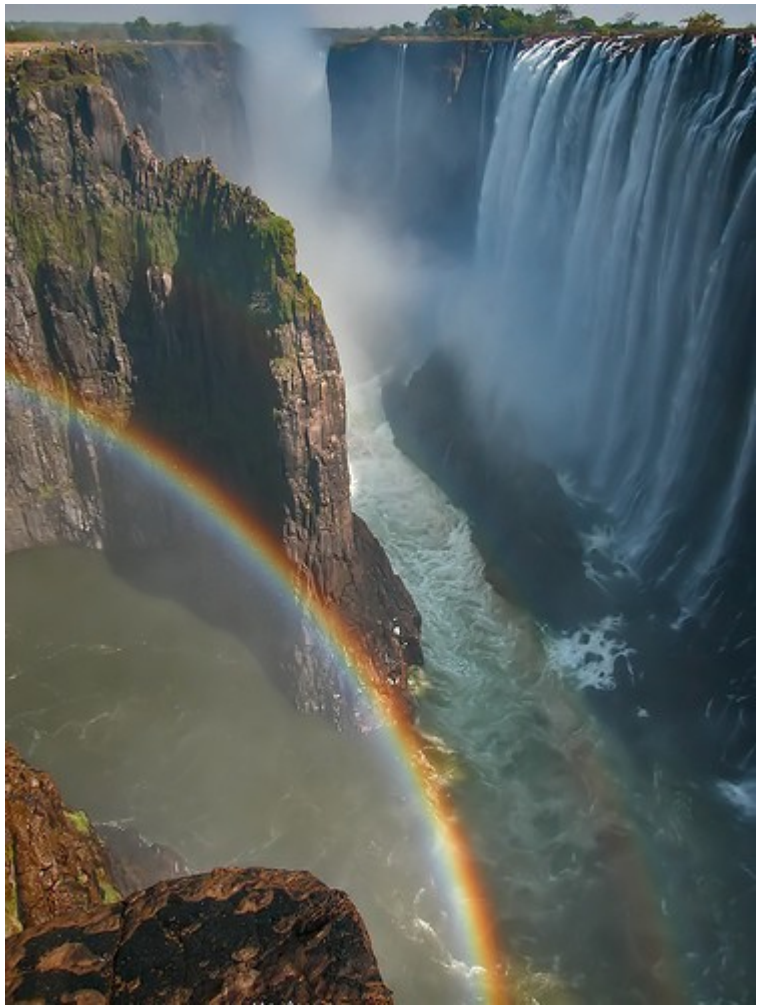
- Начальное – проводится разработчиком
- Функциональное - проверка соответствия ПО заявленному функционалу
- Интеграционное – взаимодействие модулей
- Нагрузочное – функционирование системы при ограниченных ресурсах

Некоторые рекомендации

1. Повторный просмотр кода резко снижает количество ошибок
2. Принцип Парето (80-20): 20% программистов генерирует 80% ошибок
3. Копирование кода - ЗЛО

Каскадная модель

Этапы идут один за одним. Начав один этап, мы не можем вернуться назад



Каскадная модель

Каскадная модель

Плюсы:

- Позволяет предсказать затраты на проект
- Управляемость проекта
- Четкие сроки сдачи

Минусы:

- Отсутствие гибкости
- Получается задуманная, а не необходимая система (если получается)

Каскадная модель

Хорошо подходит

- для небольших проектов, в которых заданы короткие сроки и ясные цели
- для больших проектов, в которых поставленная задача обязана быть достигнута, даже путем перерасхода средств

Модель с возвратами

На практике в ходе разработки происходят изменения, связанные с:

- ошибками анализа, проектирования, разработки
- изменением требований заказчика
- изменениями во внешней среде
- конъюнктуре

Модель с возвратами

Модель с возвратами

Плюсы

- выше шансы получить необходимую систему
- гибкость разработки

Минусы

- не известно, сколько потребуется возвратов, то есть времени, средств,

Итерационные модели

В ряде случаев возвраты неизбежны

- изменяющаяся внешняя среда
- изменяющиеся требования заказчика
- нечеткие требования
- новая предметная область
- невозможность работать по модели с возвратами

Итерационные модели

Итерационные модели

В основу положен цикл

Plan-Do-Check-Act

повторяемый несколько раз

Петля качества

Итерационные модели

Плюсы

- предсказуемость процесса разработки на короткий период
- повышенная гибкость разработки
- минимизация затрат

Минусы

- неопределенность разработки

Гибкие модели

В некоторых случаях удобнее вести разработку без четко очерченного плана, имея перед собой только постоянно уточняемую конечную цель

Гибкие модели

Гибкие модели

Плюсы

- сверхгибкая разработка системы
- максимальная чувствительность к изменению требований
- минимизация затрат

Минусы

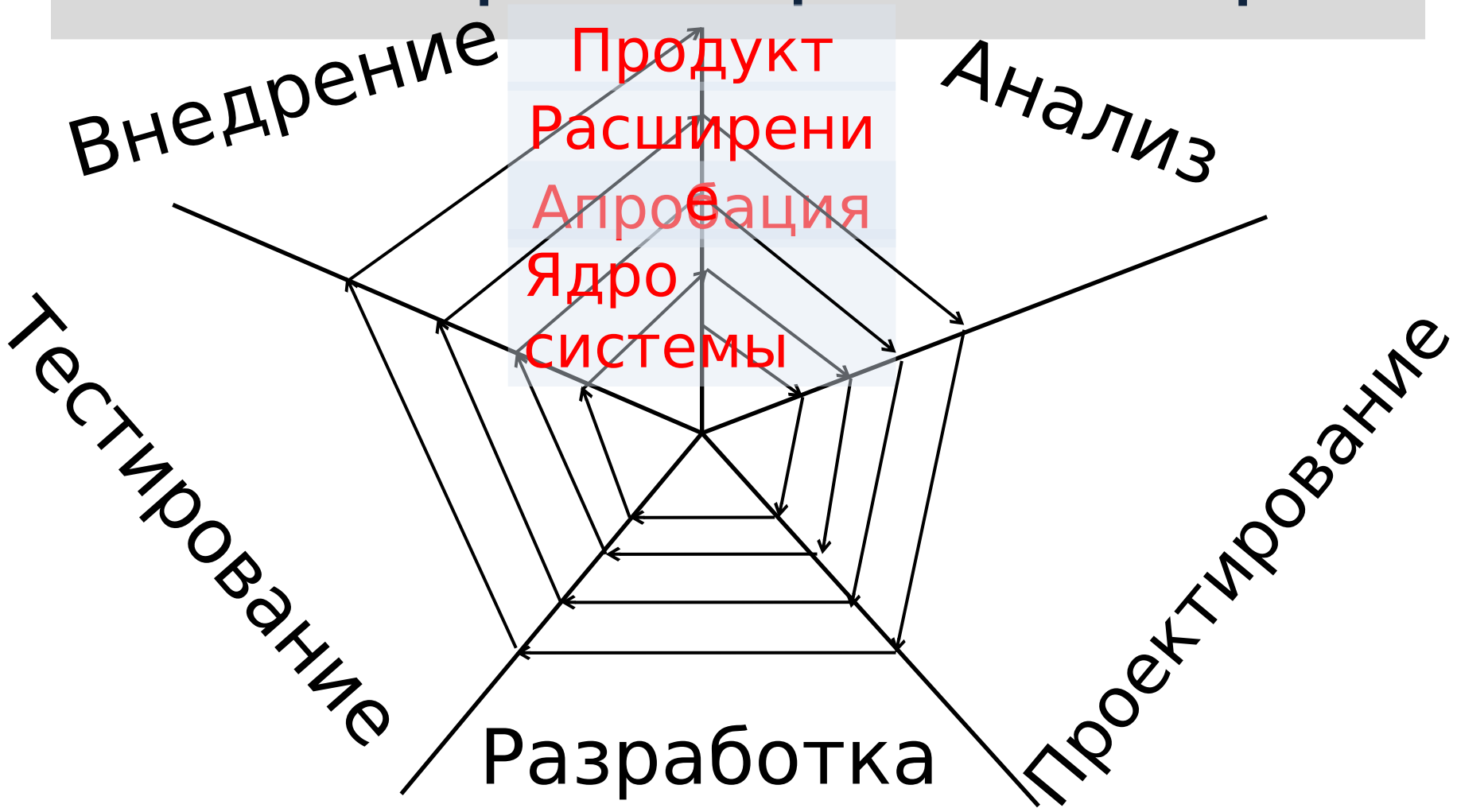
- совершенно не понятно, куда зайдет проект
- подходит лишь для небольших проектов

Метод расширения ядра

Данная модель жизненного цикла ПО предполагает проведение нескольких итераций.

1. Создание ядра системы
2. Уточнение требований, апробация
3. Добавление доп. функций
4. Готовая система

Метод расширения ядра



Возможные этапы

1. Формирование видения проблемы
2. Расстановка приоритетов
3. Согласование временных рамок
4. Определение архитектуры
5. Формирование плана
6. Работы по плану

Метод расширения ядра

Плюсы

- Позволяет создать систему, задание на которую сформулировано нечетко и при отсутствии знаний в данной области
- Экономия средств, если решение об отказе от разработки принимается на начальной итерации

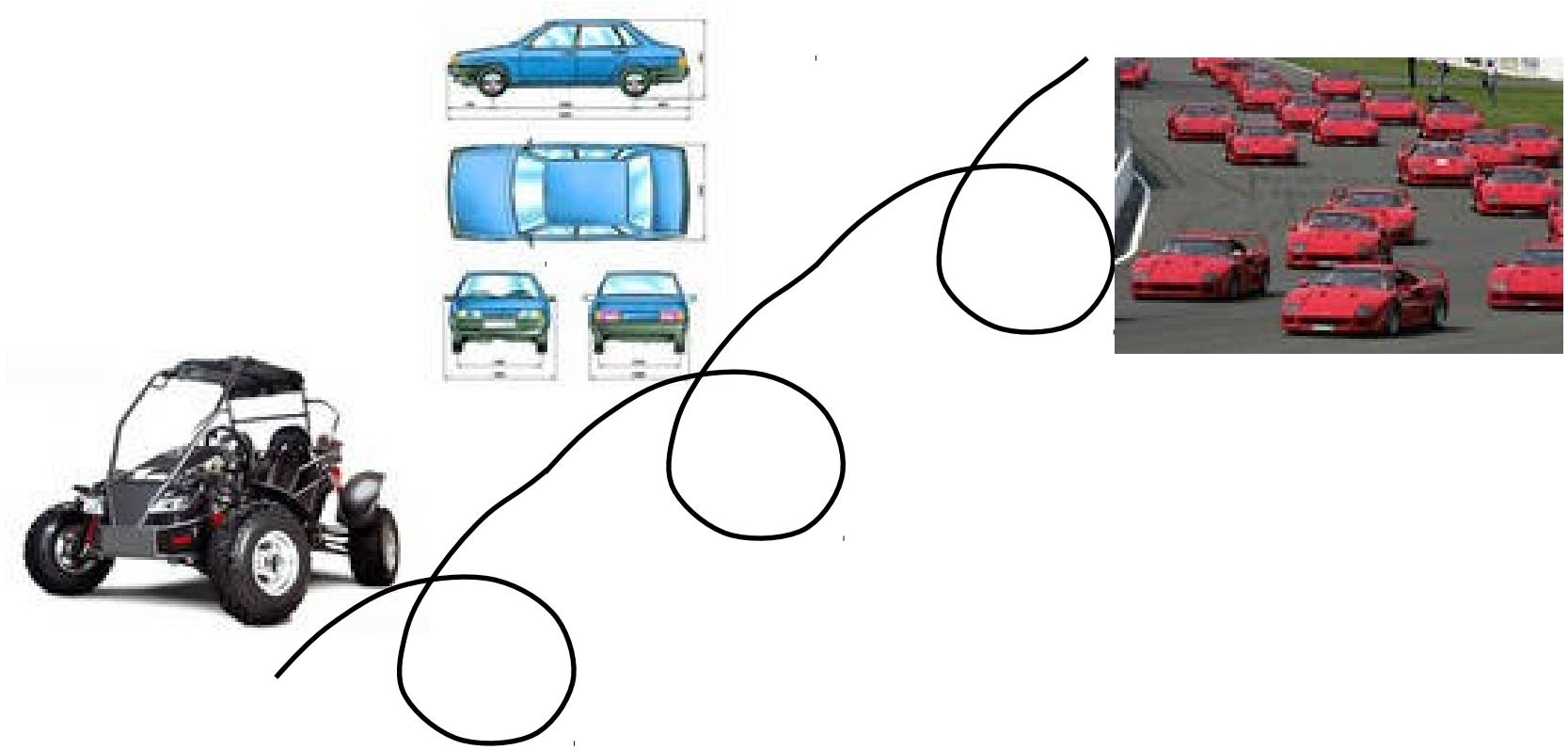
Итерационная методика

Данная методика применима в случаях, когда время на исполнение проекта превышает имеющееся.

Методика предполагает возможность разделить проект на несколько самостоятельных последовательно идущих фрагментов

Аналогия

Надо разработать гоночный автомобиль



– Выбери своего Научного руководителя.
– Как?
– Ты почувствуешь. А затем он должен выбрать тебя.
– А что потом?
– Ты должен установить с ним связь. Действуй быстро, у тебя всего одна попытка.
– Как я узнаю, что он меня выбрал?
– Он захочет тебя убить.

Agile Manifesto

Разрабатывая программное обеспечение и помогая другим делать это, мы стараемся найти наилучшие подходы к разработке. В процессе этой работы мы пришли к тому, чтобы ценить

- личности и их взаимодействия важнее, чем процессы и инструменты,
- работающее программное обеспечение важнее, чем полная документация,
- сотрудничество с заказчиком важнее, чем контрактные обязательства,
- реакция на изменения важнее, чем следование плану.

Понятия справа важны, но мы больше ценим понятия слева.

Agile Manifesto - принципы

1. Удовлетворение клиента за счёт ранней и бесперебойной поставки ценного ПО
2. Приветствие изменения требований, даже в конце разработки. Это может повысить конкурентоспособность полученного продукта
3. Частая поставка рабочего ПО (каждый месяц или неделю или ещё чаще)
4. Тесное, ежедневное общение заказчика с разработчиками на протяжении всего проекта

Agile Manifesto - принципы

5. Проектом занимаются мотивированные личности, которые обеспечены нужными условиями работы, поддержкой и доверием.
6. Рекомендуемый метод передачи информации это личный разговор (лицом к лицу)
7. Работающее ПО — лучший измеритель прогресса
8. Спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределенный срок

Agile Manifesto - принципы

9. Постоянное внимание на улучшение технического мастерства и удобный дизайн
10. Простота — искусство НЕ делать лишней работы
11. Лучшие архитектура, требования и дизайн получаются у самоорганизованной команды
12. Постоянная адаптация к изменяющимся обстоятельствам

Экстремальное программирование

Все этапы проводятся одновременно.

В любой момент времени команда анализирует потребности заказчика, проектирует систему, разрабатывает и тестирует ее.

Основные принципы

Короткий цикл обратной связи

1. Разработка через тестирование
2. Выбор приоритетов и управление рисками
3. Выделенный пользователь (Заказчик всегда рядом)
4. Парное программирование

Основные принципы

Непрерывный процесс разработки

5. Непрерывная интеграция

6.Частые небольшие релизы

7.Переоценка потребностей

Основные принципы

Понимание

8. Простота проектирования
9. Метафора системы/ взаимодействие
10. Коллективное владение кодом
11. Стандарт кодирования

Основные принципы

Социальная защищенность

12.40-часовая рабочая неделя

Условия работы

- Секретарь для ответа на звонки
- Забаррикадировать дверь и выбросить пиратский флаг
- Никаких изменений в поставленную задачу

Экстремальное программирование

Применение отдельных принципов не переводит разработку системы в область экстремального программирования.

Но отдельные принципы могут успешно применяться на практике

Экстремальное программирование

Плюсы:

- Гибкость и скорость разработки
- Повышенное качество разработки

Минусы:

- Подходит для небольшой команды

Проблема карго-культа

Лицензирование проводится зачастую просто ради того, чтобы соответствовать формальным требованиям, а не вывести бизнес-процессы на качественно новый уровень.

Метод функциональных точек

Функциональная точка (ф.т.) –

работа, которую сотрудник может выполнить за 1-2 дня.

100 строк кода на C/C++

8 страниц документации

… введенных элементов данных

… нарисованных изображений

Метод функциональных точек

1 ф.т. – небольшая утилита (1-2 дня)

10 ф.т. – небольшое приложение (до месяца)

100 ф.т. – приложение (6 месяцев в команде, предел для одиночки, 85% вероятность успеха)

Метод функциональных точек

1000 ф.т. – коммерческое приложение (год в команде из 10 человек, 85% вероятность успеха в команде, 45% - для одиночки)

10000 ф.т. – для реализации требуется команда из 100 разработчиков (1,5 – 5 лет, 50% вероятность успеха).

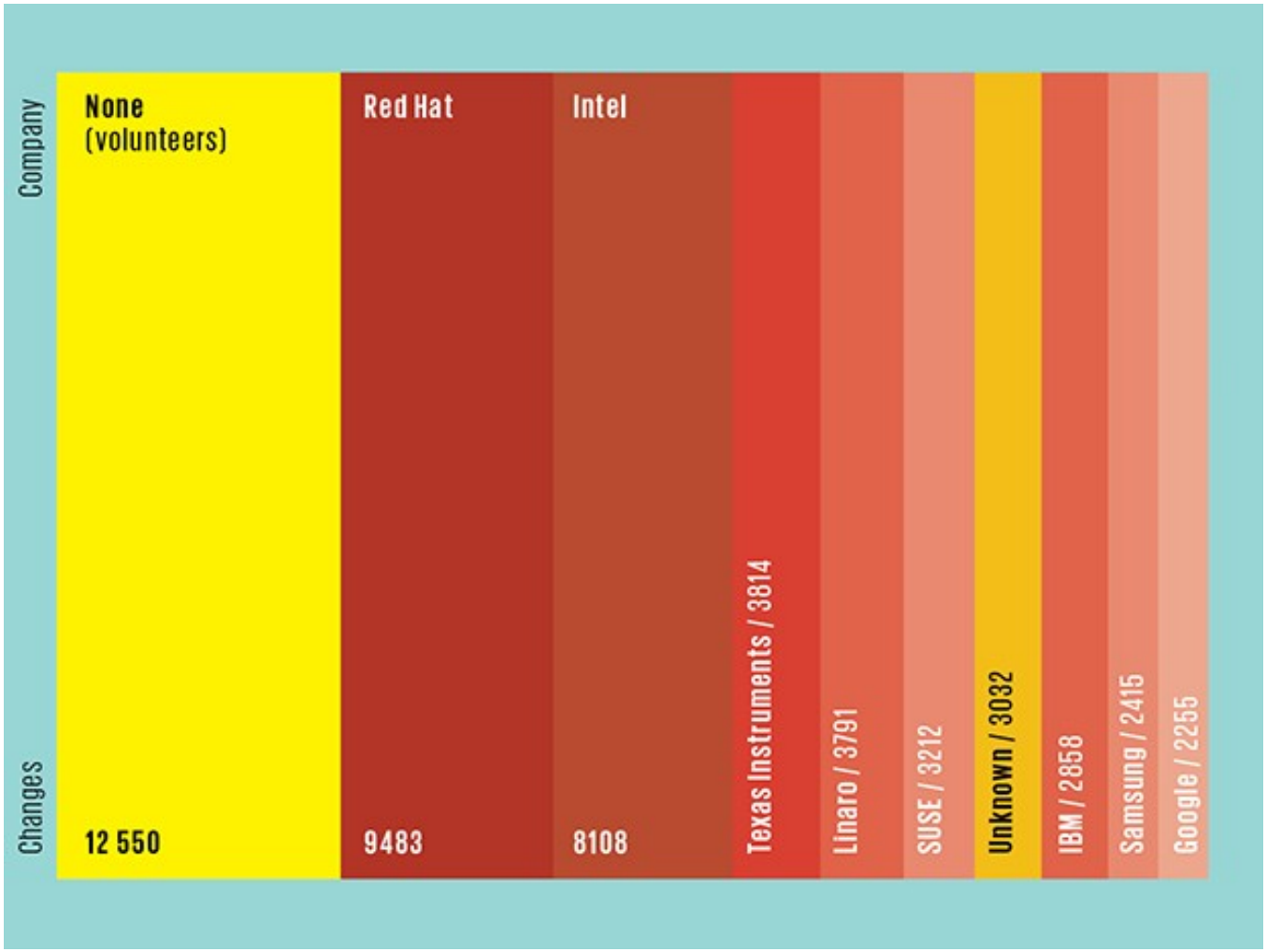
100000 ф.т. – системы уровня Windows (5-8 лет, 35% успеха)

Метод функциональных точек

- Система управления огнем истребителя F16 – 236 тыс. строк кода
- Windows Vista – 55 млн строк кода
- MacOS X – 86 млн строк кода
- Debian Linux 4.0 – 283 млн. строк кода

https://informationisbeautiful.net/visualizations/million-lines-of-code/

Метод функциональных точек



Метод функциональных точек

Американский программист пишет в год порядка 7,7 тыс. строк кода, выполняя при этом в среднем 63 функциональных точек. Европейский программист пишет 16,7 тысяч строчек кода, проходя при этом менее 30 функциональных точек. Основное отличие заключается в качестве сопровождаемости полученного кода и распределении обязанностей.

Персональный процесс

Задачи:

- улучшить навыки оценки и планирования;
- научиться осознавать и выполнять взятые обязательства;
- научиться управлять качеством проекта;
- уменьшить количество ошибок.

Персональный процесс

По PSP специалист должен пройти три стадии из четырех:

- несознательно некомпетентен;
- сознательно некомпетентен;
- сознательно компетентен;
- несознательно компетентен.

Персональный процесс

Формально методика содержит 7 последовательных этапов, через которые проходит программист. Для контроля используется 76 различных форм. Однако положения PSP достаточно просты и могут применяться в неформальном порядке.

Персональный процесс

The key data collected in the PSP tool are time, defect, and size data – the time spent in each phase; when and where defects were injected, found, and fixed; and the size of the product parts. Software developers use many other measures that are derived from these three basic measures to understand and improve their performance. Derived measures include:

estimation accuracy (size/time); prediction intervals (size/time); time in phase distribution; defect injection distribution; defect removal distribution; productivity; reuse percentage; cost performance index; planned value; earned value; predicted earned value; defect density; defect density by phase; defect removal rate by phase; defect removal leverage; review rates; process yield; phase yield; failure cost of quality (COQ); appraisal COQ; appraisal/failure COQ ratio

Персональный процесс

В ходе работы измеряются:

- размер (например, в строчках кода);
- усилия (потраченное время);
- качество (число ошибок);
- планирование (соответствие сроков)

Уровень 0

Начальный уровень.

Здесь программист приучается вести тестирование, документирование всех параметров своей деятельности, учится планировать свою работу.

Уровень 0.1.

Прививаемые навыки:

- Составление последовательности выполняемых работ
- Примерно оценивать время, необходимое на создание отдельных модулей

Уровень 0.1.

Прививаемые навыки:

- Тестирование разрабатываемых модулей
- Документирование всех действий – количество строчек кода на модуль, количество ошибок, затраченное время на разработку и поиск ошибок

Уровень 0.2.

Программист учится оценивать объем работ по задаче в целом

Прививаемые навыки:

- Документирование объема работ по задаче – общее количество строк, модулей, затраченное время, обнаруженные ошибки, страницы документации

Уровень 1.1.

На данном уровне программист учится правильно оценивать сложность будущих работ. При этом учитывается дописывается новый код, создается новый модуль или объект, модифицируется имеющийся код, оценивается сложность проекта.

Уровень 1.2.

На данном уровне программист учится оценивать объем предстоящих работ. На основе накопленного опыта составляется календарный план выполнения работ.

Уровень 1.2.

Этапы составления плана работ:

- Оценка требований к продукту
- Состав модулей программы
- Оценка объема модулей
- Оценка времени на модули
- Составление плана работ

Уровень 1.2.

По ходу выполнения работ отслеживается соответствие плану. При выявлении несоответствий необходимо понять причину их появления.

Уровень 2.

Уровень посвящен управлению качеством программного кода и качеству проектирования.

Уровень 2.1.

Основным инструментом здесь является поиск ошибок «методом пристального взгляда» - тщательное изучение написанного кода еще до этапа компиляции.

Уровень 2.1.

Целью подобных просмотров является обучение поиску ошибок, выявление «любимых» ошибок и борьба с ними.

Некоторые организации проводят собрания, посвященные разбору новых видов ошибок.

Уровень 2.1.

Еще одним методом является повторный просмотр кода, например, раз в неделю. Целью повторного просмотра является анализ написанного кода, когда программист не «зашорен» текущим решением, сидящим у него в голове.

Уровень 2.2.

Основной задачей здесь является улучшение навыков проектирования систем.

Уровень 2.2.

Проводится сравнение начальных планов с реально получившейся траекторией выполнения работ по проекту.

Анализируются причины отклонений, причины неточного составления планов.

Уровень 3.

Посвящен самостоятельному совершенствованию процесса разработки.

Программист находит индивидуальные недостатки и разрабатывает программу борьбы с ними.