Нижегородский государственный университет им. Н.И. Лобачевского

Методологии проектирования и разработки программного обеспечения

При поддержке компании Intel

Сысоев А.В., кафедра математического обеспечения ЭВМ, факультет ВМК

Содержание

- Программные продукты и программные проекты
- Зачем нужны методологии?
- Обзор Rational Unified Process (RUP)
- Методология Microsoft Solutions Framework (MSF)
- Обзор Capability Maturity Model (CMM), Capability Maturity Model Integration (CMMI)
- Гибкие методологии разработки (Agile)
- Обзор eXtreme Programming (XP)
- Обзор Scrum
- Заключение

ПРОГРАММНЫЕ ПРОДУКТЫ И ПРОГРАММНЫЕ ПРОЕКТЫ

Программный продукт

- ***ПО*** = программа + документация + [...].
- Вместо ПО часто говорят «***программный продукт***».
- Программные проекты – разработка программных продуктов.

- **Каковы критерии успеха программного проекта?**

Критерии успеха

- Программный продукт – кому-то нужен, кем-то используется.
- Каковы критерии успеха программного проекта?
  - Проект уложился в сроки.
  - Проект уложился в бюджет.
  - Сделано то, что требовалось.

- **Всегда ли так получается?**

Виды проектов

- Виды проектов:
  - ***Успешные*** (все в порядке).
  - ***Испытавшие значительные трудности*** (не выполнена часть критериев, но проект все-таки принес полезный результат, который применяется).
  - ***Провальные*** (отсутствие полезного результата).

- **Как Вы думаете, каково распределение программных проектов между указанными группами?**

Распределение проектов по группам

Год	Провалились (%)	Трудности (%)	Успешные (%)
1994	31	53	16
1996	40	33	27
1998	28	46	26
2000	23	49	28
2004	18	53	29
2006	19	46	35
2008	24	44	32
2010	21	42	37
2012	18	43	39

* По данным Chaos report (http://www.few.vu.nl/~x/chaos/chaos.pdf, http://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf)

Еще немного статистики

- Успешность малых проектов (**< $1 млн.**):
  - Успешные: 76%.
  - Трудности: 20%.
  - Провальные: 4%.
- Успешность больших проектов (> **$10 млн.**):
  - Успешные: 10%.
  - Трудности: 52%.
  - Провальные: 38%.

* По данным Chaos report (http://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf)

ЗАЧЕМ НУЖНЫ МЕТОДОЛОГИИ?

Сложность IT-проектов

- Чем больше проект, тем (скорее всего) выше ***сложность задач*** и ***сложность управления***.

- Как бороться со сложностью задач? (1)
- Как бороться со сложностью управления? (2)

Ответ: разрабатывать и внедрять специальные **технологии и методологии**

(1) – технологии программирования (изучали в курсах Основы программирования, ООП).

(2) – методологии разработки ПО.

Зачем нужны методологии?

- Индустрия решает все более сложные задачи.
- При этом возникают большие проекты (*коллективная работа, распределенная работа, …*).
- Чтобы справиться со сложностью задач, разрабатываются и совершенствуются технологии программирования и **методологии разработки**.
- Усилия постепенно приводят к результату (рост доли успешных проектов).

Методология разработки ПО это…

- Методология есть принципы и способы организации деятельности проектной группы для создания программного продукта.
- В зависимости от сложности программного продукта и связанного с ней размера проектной группы используют разные методологии.
- Рассмотрим некоторые из них…

ОБЗОР RATIONAL UNIFIED PROCESS

Rational Unified Process – процесс разработки ПО

- Предполагает использование объектного подхода.
- Является четким руководством для членов команды.
- Ориентирован на итеративную разработку.
- Основан на вариантах использования и компонентной архитектуре.
- Использует UML в качестве единого языка для документирования (нотация, модели, диаграммы).
- Является интегрированным продуктом (поддержан специализированным ПО).
- Допускает гибкую настройку и изменения под конкретные нужды.

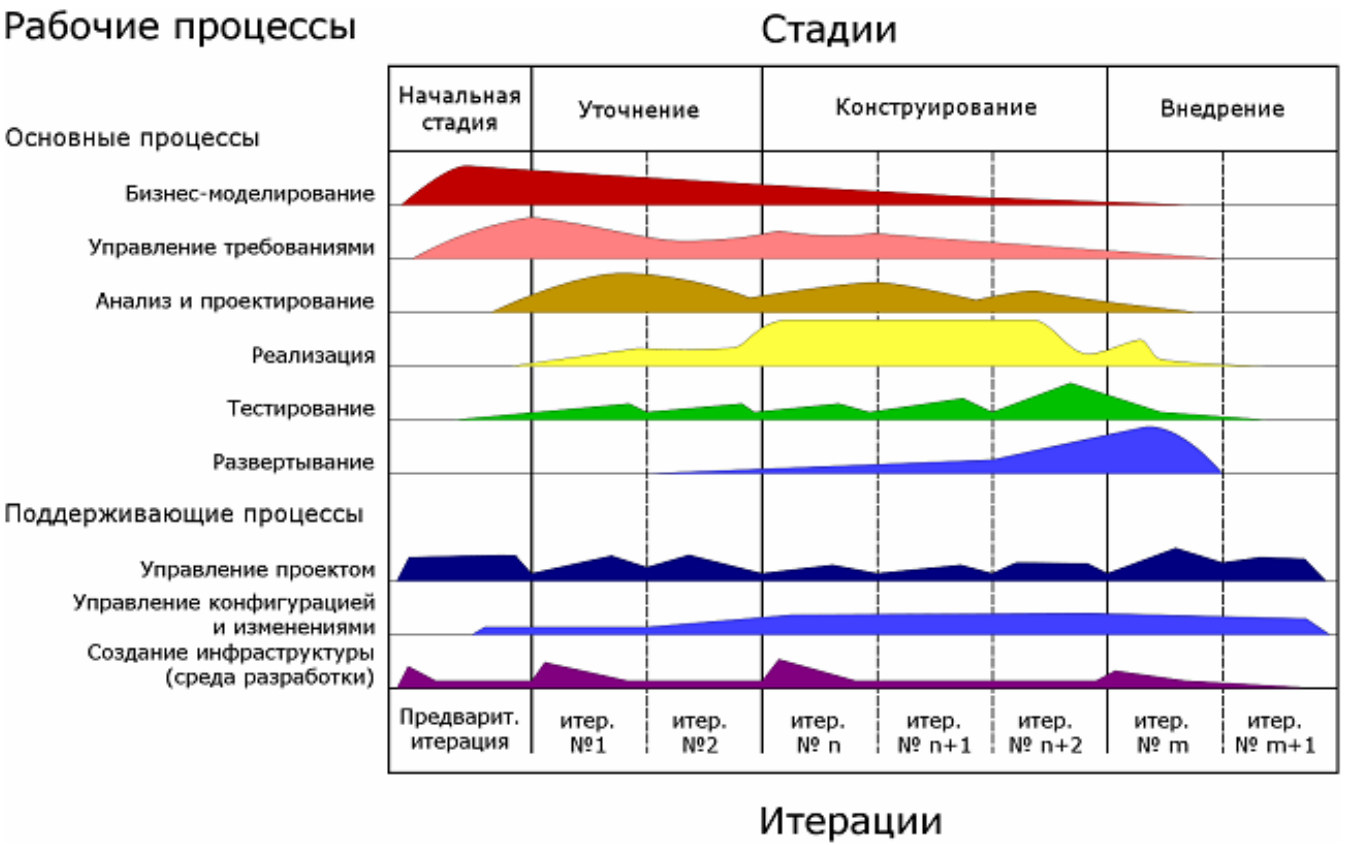
Базовые принципы RUP*

- Непрерывно ведите ***наступление на риски***.
- Обеспечьте ***выполнение требований*** заказчиков.
- Сконцентрируйтесь на ***исполняемой программе***.
- Приспосабливайтесь к ***изменениям с начала проекта***.
- Рано закладывайте ***исполняемую архитектуру***.
- Стройте ***систему из компонентов***.
- Работайте вместе как ***одна команда***.
- Сделайте ***качество*** образом жизни.

* Кролл П., Крачтен Ф. Rational Unified Process – это легко. Руководство по RUP для практиков

Структура RUP. Жизненный цикл, итерации и фазы проекта

- RUP в одном слайде*:



* Источник: www.wikipedia.org

МЕТОДОЛОГИЯ MSF

Основные концепции методологии MSF…

- MSF – методология разработки программного обеспечения от компании Microsoft.
- Опирается на практический опыт компании.
- Описывает управление людьми и управление процессами в ходе разработки программного продукта.
- Предлагает не только концепции и модели, но и сугубо практические приемы и советы.

Основные концепции методологии MSF

- MSF состоит из двух моделей и трех дисциплин:
  - Модель процессов MSF.
  - Модель проектной группы MSF.
  - Дисциплина управления проектами MSF.
  - Дисциплина управления рисками MSF.
  - Дисциплина управления подготовкой MSF.
- MSF предлагает несколько оригинальных идей:
  - Единое видение проекта.
  - Треугольник и матрица компромиссов.
  - “Проектная группа – команда равных”.
  - Управление рисками.

Два направления в MSF 4.0…

- В MSF 4.0 произошло разделение методологии на два направления:
  - MSF for CMMI Process Improvement.
  - MSF for Agile Software Development .

- MSF for CMMI (Capability Maturity Model Integration) Process Improvement – строгий, документированный процесс, рассчитанный на большие команды и длительный процесс разработки, что предполагает больше верификации, больше планирования, процедуры утверждения, отслеживание потраченных ресурсов и т.д.

Два направления в MSF 4.0

- В MSF 4.0 произошло разделение методологии на два направления:
  - MSF for CMMI Process Improvement.
  - MSF for Agile Software Development.

- MSF for Agile Software Development отражает тенденции последнего времени, связанные с появлением методологий, предлагающих максимально облегченный и гибкий подход к процессу разработки.
- Agile-направление в MSF ориентируется на небольшие команды (5-6 человек). Информация о разрабатываемом продукте не просто выясняется в процессе разработки, а может и будет изменяться по ходу.

Элементы MSF 4.0

- Рекомендованные процессы создания IT-проектов.
- Структура итераций.
- Роли членов команды.
- Шаблоны документов (Excel, Word).
- Шаблоны Microsoft Project.
- Отчеты.
- Портал проекта (шаблон сайта SharePoint).

MSF for Agile Software Development ориентирован на использование итеративной и эволюционной модели процесса разработки и основан на сценариях использования.

ОБЗОР CMM, CMMI

Capability Maturity Model

- Capability Maturity Model – модель зрелости процессов разработки программного обеспечения.

- Явилась результатом обзора, подготовленного американским институтом Software Engineering Institute по запросу министерства обороны США для ответа на вопрос: «Как выбирать организацию, которой можно доверить выполнение крупного IT проекта?».

- В отчете изложена модель зрелости организаций, которая определялась через зрелость процесса разработки ПО, применяемого в этой организации.

Уровни зрелости процесса в CMM…

В CMM выделяется пять уровней зрелости процесса, которые устанавливают степень готовности организации выполнить крупный проект.

1. ***Начальный (Initial)***

   Технология полностью импровизированная, в некоторых случаях даже хаотическая. Успех всецело зависит от личных качеств сотрудников и может быть повторен только в случае назначения тех же сотрудников на новый проект.

2. ***Повторяемый (Repeatable)***

   Базовые процессы управления проектом ПО установлены. Есть дисциплина соблюдения, что обеспечивает возможность повторения успеха предыдущих проектов в той же прикладной области.

Уровни зрелости процесса в CMM…

В CMM выделяется пять уровней зрелости процесса, которые устанавливают степень готовности организации выполнить крупный проект.

3. ***Определенный (Defined)***

   Процессы документированы, стандартизованы и интегрированы в единую для всей организации технологию создания ПО. Для каждого проекта используется адаптивный вариант этой технологии. Начиная с этого уровня, организация перестает зависеть от качеств конкретных разработчиков.

4. ***Управляемый (Managed)***

   Собираются и накапливаются метрики (объективные данные) о качестве исполнения процессов и выходной продукции. Управление процессами и выходной продукцией осуществляется по количественным оценкам.

Уровни зрелости процесса в CMM

В CMM выделяется пять уровней зрелости процесса, которые устанавливают степень готовности организации выполнить крупный проект.

5. ***Оптимизируемый (Optimized)***

   Совершенствование технологии создания ПО осуществляется непрерывно на основе количественной обратной связи от процессов и пилотного внедрения инновационных идей. Ведутся работы по уменьшению стоимости разработки программного обеспечения, например, с помощью создания и повторного использования компонент.

Сертификация организаций на уровень CMM

- При сертификации проводится оценка соответствия всех ключевых областей по 10-балльной шкале.
- Для успешной квалификации данной ключевой области необходимо набрать не менее 6 баллов.
- Оценка ключевой области производится по следующим показателям:
  - Заинтересованность руководства в данной области.
  - Насколько широко данная область применяется в организации.
  - Успешность использования данной области на практике.

Capability Maturity Model Integration

- CMMI является развитием CMM.
- CMMI – набор моделей совершенствования процессов в организациях разных размеров и видов деятельности.
- К разработке ПО относится CMMI for Development.
- Существует два представления: непрерывное (continuous) и ступенчатое (staged).
- Ступенчатое представление определяет стандартную последовательность улучшений и может служить базисом для сравнения уровней зрелости (те же, что в CMM) различных процессов и организаций

ГИБКИЕ МЕТОДОЛОГИИ РАЗРАБОТКИ

Гибкие методологии разработки…

- Гибкая методология разработки (Agile Software Development) – набор принципов и правил, в рамках которого осуществляется разработка ПО.
- Методология Agile – семейство процессов/подходов/методологий разработки ПО.
- Ценности и принципы гибких методологий закреплены в документе Agile Manifesto.
- Agile Manifesto разработан в феврале 2001 в штате Юта США. Манифест явился результатом обсуждения представителей гибких методологий разработки: Extreme programming, Scrum, DSDM, Adaptive Software Development, Crystal Clear, Feature-Driven Development, Pragmatic Programming и др.

Гибкие методологии разработки

- Большинство гибких методологий нацелено на минимизацию рисков путем сведения разработки к серии коротких циклов-итераций.
- Типовая длительность итерации – одна-две недели.
- Каждая итерация – программный проект в миниатюре, включая планирование, анализ требований, проектирование, кодирование, тестирование и документирование.
- Гибкий программный проект готов к выпуску в конце каждой итерации.
- По окончании каждой итерации команда выполняет переоценку приоритетов разработки.

Ценности гибких методологий разработки

- 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.

Принципы гибких методологий разработки…

1. Удовлетворение клиента за счет быстрой поставки ПО.
2. Приветствие изменения требований, даже на поздних этапах разработки.
3. Частая поставка рабочего ПО («каждую неделю лучше, чем каждый месяц»).
4. Тесное, ежедневное общение разработчиков с заказчиком на протяжении всего проекта.
5. Проектом занимаются мотивированные личности.
6. Личное общение – лучший метод передачи информации.

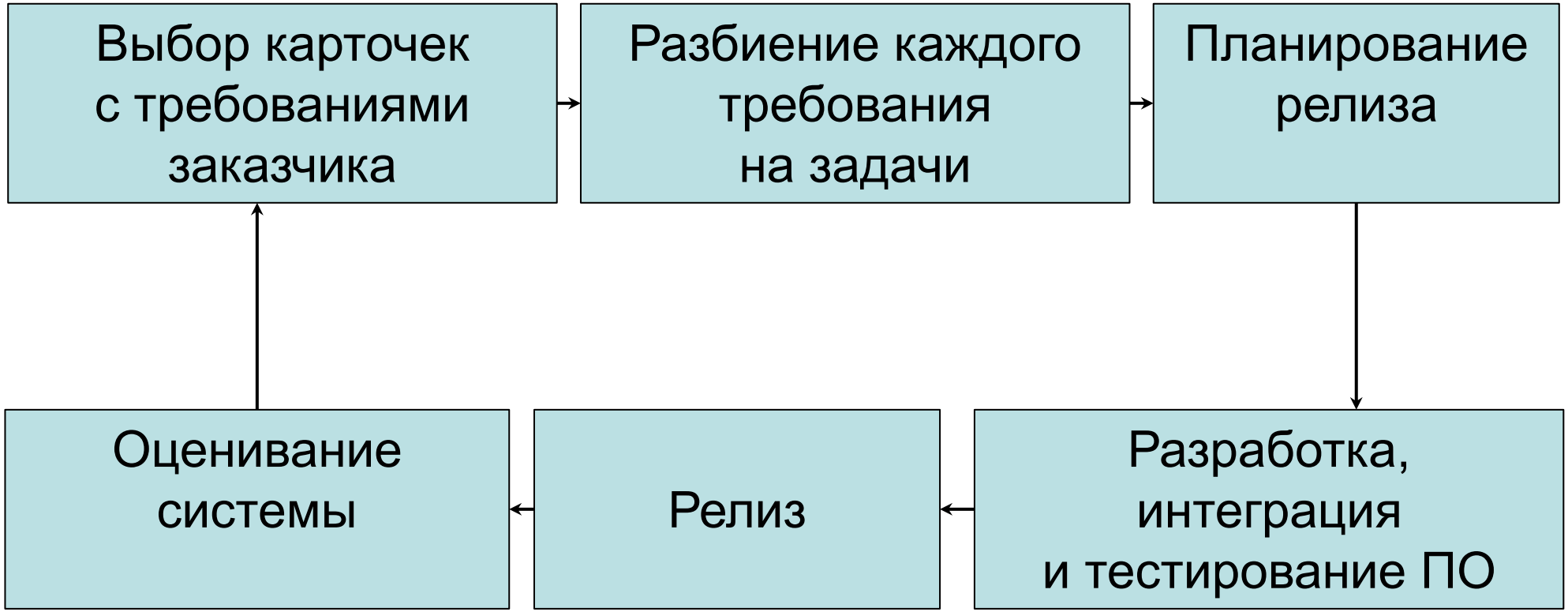
Принципы гибких методологий разработки

7. Работающее ПО – главная мера прогресса разработки.
8. Устойчивая разработка, движущаяся с постоянной скоростью.
9. Постоянное внимание техническому мастерству и удобному дизайну.
10. Простота – искусство НЕ делать лишней работы.
11. Самоорганизация команды.
12. Постоянная адаптация к изменяющимся обстоятельствам.

Экстремальное программирование (eXtreme Programming)

- Экстремальный подход к итеративной разработке:
  - Новые версии собираются несколько раз в день.
  - Прогресс демонстрируется заказчику каждые 2 недели.
  - Тесты запускаются для каждой версии.
  - Версия принимается только при условии, чтобы все тесты выполнились успешно.
- Игра в планирование:
  - Цель – сформировать примерный план работы.
  - Артефакты – набор карточек с пожеланиями заказчика (user stories).
  - Заказчик принимает бизнес-решения, команда разработчиков – технические решения.

Цикл разработки в XP



Sommerville I. Software engineering (9th edition). – Pearson, 2011. – 790 p.

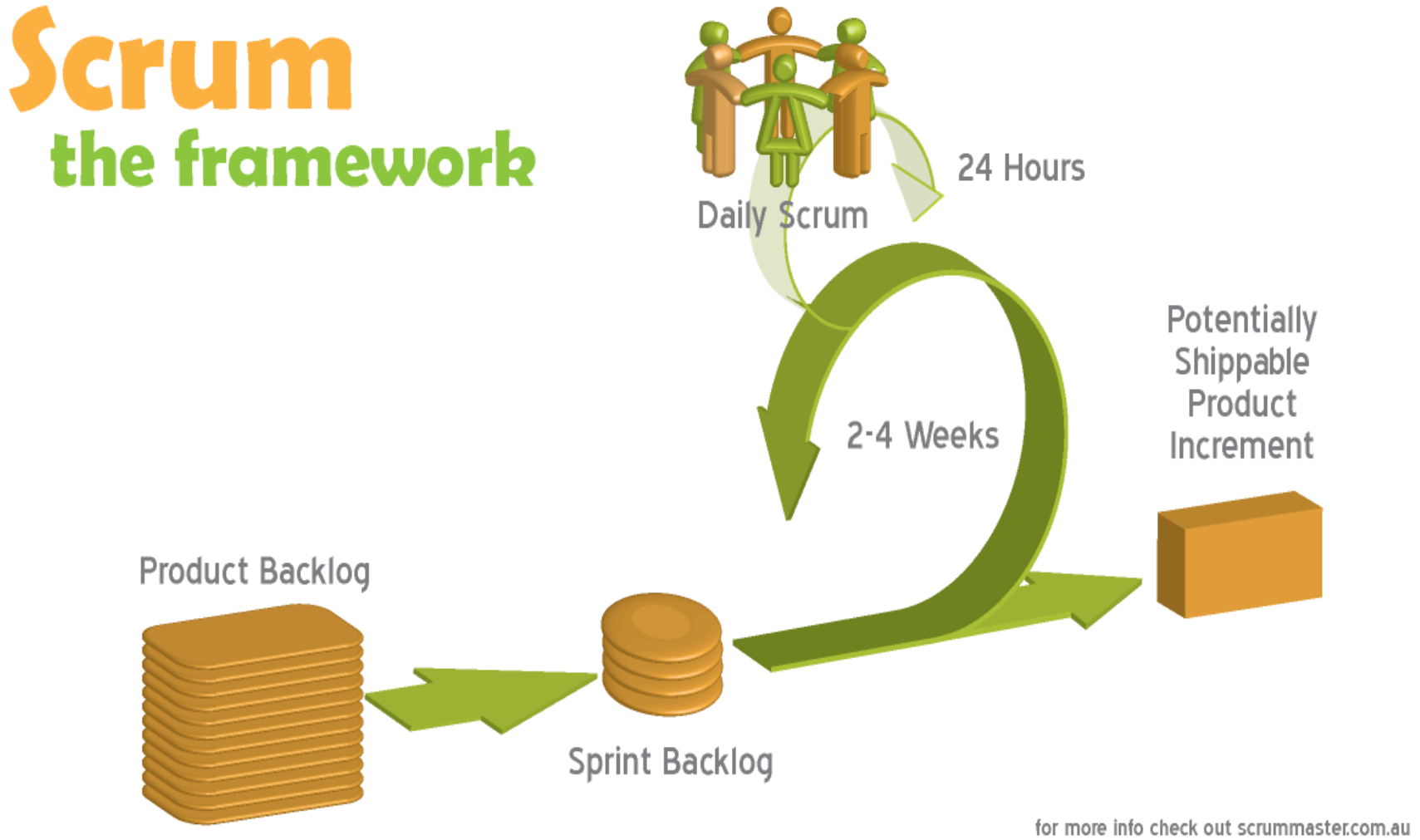
Практики XP

- Инкрементное планирование.
- Небольшие релизы.
- Простой дизайн.
- Разработка «сначала тест, потом реализация» (test first development или test-driven development).
- Рефакторинг.
- Парное программирование.
- Коллективное владение кодом.
- Непрерывная интеграция.
- Устойчивый темп разработки.
- Постоянная доступность представителя заказчика.

Основы Scrum

- Итерация в Scrum называется ***спринтом*** (sprint).
- В ходе спринта создается функциональный прирост программного обеспечения.
- Спринт жестко фиксирован по времени. Длительность одного спринта от 1 до 4 недель (обычно 2 недели).
- На протяжении спринта никто не имеет права менять список требований к работе, внесенных в резерв спринта.
- ***Резерв спринта*** содержит функциональность, выбранную владельцем продукта из резерва проекта.
- ***Резерв проекта*** – список требований к функциональности, упорядоченный по степени важности.

Итеративный процесс разработки в Scrum



Project Management – SCRUM, Resource Management [http://sharewhale.com/wp/tag/minimumopposition].

Ключевые роли в SCRUM-процессе

- ***Владелец продукта*** (Product Owner) представляет интересы конечных пользователей и других заинтересованных сторон.
- ***SCRUM-мастер*** (Scrum Master) следит за соблюдением принципов SCRUM, разрешает противоречия и защищает команду от отвлекающих факторов. Данная роль не предполагает ничего иного, кроме корректного ведения SCRUM-процесса.
- ***Scrum-команда*** (Scrum Team) состоит из специалистов разных профилей: тестировщиков, архитекторов, аналитиков, программистов и т.д. Размер команды в идеале составляет 7±2 человека.

Планирование спринта (Sprint Planning Meeting)

- Из резерва проекта выбираются задачи, обязательства по выполнению которых за спринт принимает на себя команда.
- На основе выбранных задач создается резерв спринта. Каждая задача оценивается в идеальных человеко-часах.
- Решение задачи не должно занимать более 12 часов или одного дня. При необходимости задача разбивается на подзадачи.
- Обсуждается и определяется, каким образом будет реализован этот объем работ.

Ежедневное совещание (Daily Scrum meeting)

- Начинается точно вовремя даже при отсутствии кого-то из команды. Длится не более 15 минут.
- Проводится в одном и том же месте в течение спринта.
- Наблюдать могут все, только ключевые роли имеют право голоса.
- В течение совещания каждый член команды отвечает на 3 вопроса:
  - Что сделано с момента предыдущего ежедневного совещания?
  - Что будет сделано с момента текущего совещания до следующего?
  - Какие проблемы мешают достижению целей спринта?

Обзор итогов спринта (Sprint review meeting)

- Проводится после завершения спринта.
- Команда демонстрирует прирост функциональности продукта всем заинтересованным лицам.
- Привлекается максимальное количество зрителей.
- Ограничен четырьмя часами.

- Нельзя демонстрировать незавершенную функциональность.

Ретроспективное совещание (Retrospective meeting)

- Проводится после завершения спринта.
- Члены команды высказывают свое мнение о прошедшем спринте.
- Отвечают на два основных вопроса:
  - Что было сделано хорошо в прошедшем спринте?
  - Что надо улучшить в следующем?
- Выполняют улучшение процесса разработки.
- Ограничено тремя часами.

Заключение

- Введено понятие методологии разработки программного обеспечения.
- Выполнен обзор следующих методологий:
  - Rational Unified Process (RUP).
  - Методология Microsoft Solutions Framework (MSF).
  - CMM (Capability Maturity Model), CMMI (Capability Maturity Model Integration).
  - Гибкие методологии разработки (Agile):
    - eXtreme Programming (XP).
    - Scrum.

Контрольные вопросы…

- Зачем нужны методологии?
- Что такое методология разработки программного обеспечения?
- Приведите базовые принципы RUP.
- Из каких элементов состоит методология MSF?
- Какие уровни зрелости выделены в методологии CMM?
- На каких идеях основаны гибкие методологии разработки ПО?
- Приведите ценности гибких методологий ПО.
- Приведите принципы гибких методологий ПО.
- Опишите цикл разработки ПО в XP.
- Поясните основные практики методологии XP.

Контрольные вопросы

- Опишите итерацию разработки ПО в Scrum.
- Какие ключевые роли выделяются в Scrum?
- В чем состоит планирование спринта в Scrum?
- Зачем необходимо ежедневное совещание в Scrum? Какие вопросы решаются?
- Зачем необходимо ретроспективное совещание в Scrum? Какие вопросы решаются?

Литература к лекции

1. **Соммервиль И.** Инженерия программного обеспечения. – М.: Издательский дом "Вильямс", 2002. – 624 с.
2. **Буч Г.** Объектно-ориентированный анализ и проектирование с примерами приложений на C++. Второе издание. – Бином, 1998.
3. **Кролл П., Крачтен Ф.** Rational Unified Process – это легко. Руководство по RUP для практиков. – Изд-во “КУДИЦ-Образ”, 2004.
4. **Мееров И.Б., Сысоев А.В., Козинов Е.А.** Технологии программирования. Курс на базе Microsoft Solutions Framework (MSF), 2006. [http://www.software.unn.ru/msf].
5. **Карпенко С.Н.** Введение в программную инженерию. Учебный курс, 2011. [http://www.software.unn.ru/?doc=903].
6. **Schwaber K., Sutherland J.** The Scrum Guide, 2013. [https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf].
7. **Бек К.** Экстремальное программирование. – Питер, 2002.