Адаптивные технологические подходы:
- Особенности живых подходов.
- Адаптивная разработка ПО (ASD).
- Экстремальное программирование (XP).

Лекция 13

Адаптивные подходы

- ***Адаптивные подходы* являются гибкими подходами, получившими также название *живых подходов*.** Они имеют много общего с эволюционными подходами, особенно с RAD.

Особенности адаптивных подходов:

- Открытое взаимодействие,
- Разработка короткими итерациями,
- Адаптируемость процесса разработки.

Выделяют адаптивные подходы следующих видов:

1. Игровые адаптивные подходы: Адаптивная разработка ПО (ASD), Экстремальное программирование (XP), Скрам (Scrum).
2. Управляемые адаптивные подходы: Управляемая тестами разработка (TDD), Управляемая возможностями разработка (FDD), Управляемая поведением разработка (BDD), Управляемая дизайном разработка (D3).
3. Унифицированные адаптивные подходы: Гибкие варианты UP.
4. Облегчённые адаптивные подходы: Облегчённая разработка ПО (LSD).

Адаптивный подход

- В общем случае адаптивный подход представляет собой **определённый набор принципов и практик, ориентированных на исполнение особенностей подходов.**
- Это позволяет использовать при реализации реальных проектов **сочетания различных подходов, адаптируя процесс разработки к конкретным проектам.**

«Живой манифест»

- В 2001 г. 17 известных сторонников гибких подходов встретились в местечке Сноубёрд (штат Юта, США), для обсуждения вопросов создания ПО более лёгким, быстрым и «человеко-центрированным» способом. Кроме того они предложили общее название для подходов с указанными выше способами разработки: Живая разработка ПО. Результат – «Манифест Живой разработки ПО», известный как «Живой манифест». **Живой манифест включает в себя уведомление с основными положениями и сам документ с принципами живой разработки ПО.**

Основные положения при разработке ПО связаны с правильной оценкой:

1. Люди и их взаимодействие важнее процессов и средств.
2. Работающее ПО важнее исчерпывающей документации.
3. Сотрудничество с заказчиком важнее обсуждения контракта.
4. Реагирование на изменения важнее следования плану.

Положениям и принципам Живого манифеста должны удовлетворять гибкие подходы, которые относятся к живой разработке ПО (т.е. живым подходам).

Адаптивная разработка ПО (АРП, ASD – Adaptive Software Development)

- *Адаптивная разработка ПО* (АРП, ASD – Adaptive Software Development) – живой подход, предложенный Дж. Хайсмитом.
- Идея представления процесса разработки как адаптивной системы была высказана Э. А. Эдмондсом в его статье ещё в 1974 г. Изложение этого подхода приведено в книге «Адаптивная разработка ПО: Подход сотрудничества при управлении сложными системами» (2000 г.) автора подхода Дж. Хайсмита. Она закладывает теоретическую основу адаптивных разработок. Это позволяет использовать АРП совместно с другими гибкими подходами (Crystal, FDD, XP). Подходы АРП и Crystal Family объединены их авторами в единый подход.

Ключевое положение АРП

- Высокая и частая изменчивость окружения приводит к необходимости изменений и в процессе разработки. Поэтому **ключевым положением АРП является естественность постоянной адаптации процесса для выполнения текущей работы.** Теоретической основой подхода служат модели сложных адаптивных систем.

Сложная адаптивная система

Одна из этих моделей основана на трёх ключевых понятиях:

- агент,
- среда,
- проявление.

Сложная адаптивная система представляет собой среду, в которой агенты конкурируют и кооперируют друг с другом за выполнение работы.

Проявление является ключевым свойством системы: результат работы есть итог сотрудничества агентов, а не действий отдельных агентов. Поэтому функционирование системы не может быть предсказано по поведению агентов.

Процесс разработки

Хайсмит рассматривает **процесс разработки как сложную адаптивную систему:**

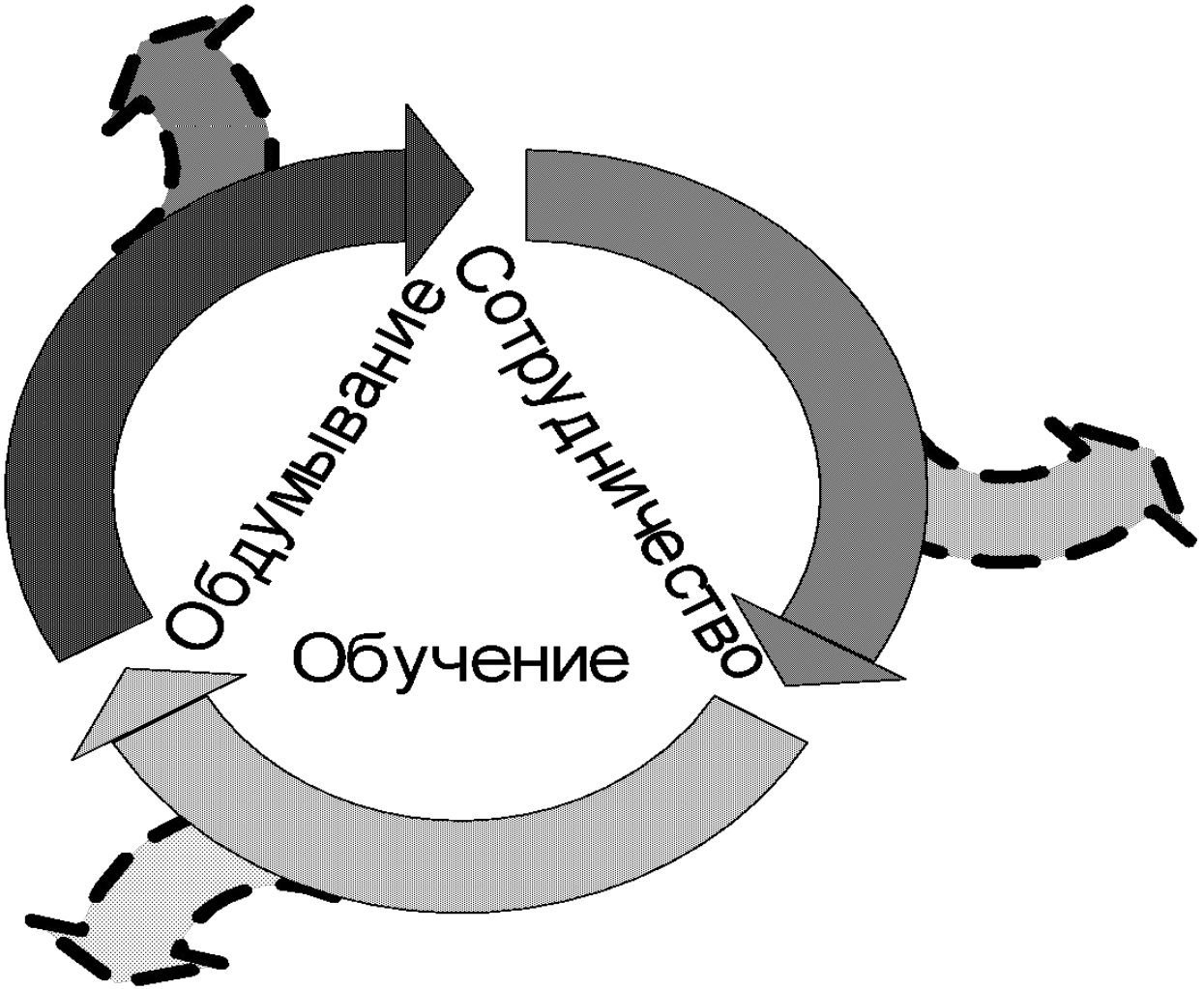
- организация-разработчик – это среда,
- участники проекта – агенты,
- а продукт – проявляемый результат сотрудничества участников.

Такое рассмотрение приводит к иному пониманию разработки и формированию нового подхода.

Динамический ЖЦ

- В АРП предлагается динамический ЖЦ «Обдумывание – Сотрудничество – Обучение». **Цикл (рис.13.1) связан с постоянными изменениями, повторными оценками, попытками предугадать неизвестное на текущий момент будущее проекта и требует тесного взаимодействия между разработчиками, тестировщиками и заказчиками.** При этом цикл не всегда представляет собой круг, так как можно иногда отклоняться в сторону, чтобы изучить неисследованные области.

Рис.13.1. Схема модели ЖЦ для подхода ASD



Цикл обладает следующими свойствами:

1. Целенаправленность;
2. Компонентный подход;
3. Итеративность;
4. Ограниченность по времени;
5. Управляемость рисками;
6. Допущение изменений.

Свойства цикла

- Целенаправленность связана с ясной формулировкой задания, на основе которой определяются цель и содержание проекта.
- Компонентный подход определяет построение ЖЦ исходя не из самих задач, а из результатов – компонентов системы.
- Итеративность необходима при создании и переделке компонентов, чтобы адекватно учитывать изменения требований заказчика по мере развития продукта.
- Ограниченность по времени требует установки фиксированных сроков поставки для каждого итеративного цикла.
- Управляемость рисками позволяет осознать и проанализировать риски.
- Допущение изменений означает приемлемость и приветствование всех возникающих изменений.

Модель ЖЦ для АРП

- Модель ЖЦ для АРП (рис.13.2) основана на приведённой выше схеме цикла.
- В модели схема конкретизируется в виде 3 фаз:

1. Обдумывание;
2. Сотрудничество;
3. Обучение.

Рис.13.2. Модель ЖЦ для подхода ASD

Инициация проекта

Инициация проекта лишь немного отличается от начальной фазы других подходов. Она включает в себя следующие действия:

1. Определение цели и задач проекта.
2. Выявление и краткое описание ограничений и требований к системе.
3. Изучение расстановки сил в проекте (организация проекта и команды).
4. Первоначальная оценка размера и масштабности проекта.
5. Определение ключевых рисков.
6. Установление временных рамок для всего проекта.

Все расчёты являются предварительными и в дальнейшем могут измениться.

Адаптивное планирование циклов

Адаптивное планирование циклов состоит из следующих действий:

1. Определение оптимального числа циклов и временных рамок каждого из них.
2. Определение цели и задач для каждого цикла разработки.
3. Соотнесение компонентов системы с циклами разработки.
4. Планирование циклов с учётом разрабатываемых компонентов.

Распределение компонентов по циклам – непростая задача. Главным критерием здесь является поставка заказчику в конце каждого цикла некоторой видимой, осязаемой, работающей части системы.

К другим критериям относят следующие:

1. Первоочередная разработка компонентов с высокой степенью риска.
2. Учёт естественных зависимостей между компонентами.
3. Балансирование расходов различных используемых ресурсов.

Параллельная разработка компонентов

- Параллельная разработка компонентов включает отдельные параллельно реализуемые действия по разработке каждого запланированного на текущий цикл компонента. **При этом руководителей больше должно беспокоить то, как добиться наиболее эффективного взаимодействия и обеспечить согласованность выполняемых работ,** нежели вопросы проектирования, тестирования и кодирования.

Адаптивность в АРП

- Адаптивность в АРП заключается в следующем: **нужно определить цель и масштаб проекта, показать команде, какие компоненты ей необходимо разработать, а затем отойти в сторону и дать разработчикам самим решать, как они будут это делать.** Чувство ответственности в команде поддерживается при помощи периодической оценки качества. В таком случае качество будет расти не благодаря дотошному контролю, а благодаря формированию соответствующих критериев для выпускаемого продукта и критическому его анализу. **Постоянный анализ и оценка проделанной работы – ключ к обучению.**

В конце каждого цикла нужно знать:

- Качество продукта с точки зрения заказчика;
- Качество продукта с технической точки зрения;
- Работоспособность команды и используемость практик;
- Текущее положение дел в проекте (статус проекта).

Промежуточные контрольные точки в конце каждого цикла призваны обеспечить доступность и обозримость создаваемого продукта для участников проекта. Без этого невозможно увидеть и исправить различные дефекты, которые присутствуют в любом проекте. В конце цикла заказчик получает определённый набор компонентов системы, которые он должен просмотреть и оценить. Это позволяет ему реально увидеть и опробовать разрабатываемую систему. Для интеграции отдельных компонентов системы в течение каждого из циклов служат промежуточные сборки. Они позволяют увидеть и опробовать систему самой команде.

Экстремальное программирование (ЭП, XP – eXtreme Programming)

– живой подход, предложенный Кентом Беком.

Возникновение ЭП тесно связано с выполнением проекта C3 (букв. Всеобъемлющее компенсирование для Chrysler) – разработка системы учёта выплат работникам фирмы Daimler Chrysler. **Суть проекта заключалась в разработке единой системы вместо множества разрозненных приложений.** Именно на этом проекте К. Беком отрабатывались практики подхода. В 1996 г. Бек стал руководить проектом, в 2000 г. фирма прервала проект. Полученная система использовалась только для 10 тысяч человек из 87 тысяч работников. Поэтому успешный на словах проект оказался провальным. Главной причиной этого является не сам подход, а его применение к неподходящему проекту. Для реализации проекта C3 (разработки формализуемой системы) нужно использовать один из строгих подходов. В 1999 г. вышло 1-е издание К. Бека «Экстремальное программирование в объяснении: Избирание изменения», а в 2004 г. – 2-е издание этой книги в соавторстве с Цинтией Андрес, переработанное с учётом опыта применения ЭП.

Ключевая и основополагающая деятельность в ЭП

- В ЭП гораздо больше спорных моментов, чем в каком-либо другом подходе. **Ключевой и основополагающей деятельностью в ЭП считается кодирование,** поэтому подход использует идею непланируемой модели – «кодирование – исправление»,– расширяя её принципами эффективной организации работ. Эти принципы доводятся до крайности их реализации в используемых практиках.
- При этом практики ЭП оказываются слишком тесно взаимосвязанными, что приводит к излишней жёсткости подхода. **Практики считаются фиксированной частью, в то время как ЖЦ системы – адаптивной частью ЭП, что противоречит логике разработки и приводит к проблемам в проектах, использующих этот подход.**

Категории ЭП

ЭП представляется категориями:

- ценности,
- принципы,
- практики.

Ценности выражают общую направленность ЭП и конкретизируются в принципах, соответствие которым служит мерой приемлемости и отбора практик для этого подхода.

В ЭП рассматривается модель ЖЦ идеального проекта для ЭП (рис.13.3).

Согласно этой модели выделено 5 фаз ЖЦ:

1. Исследование; 2. Планирование; 3. Реализация / Итерации; 4. Продуцирование / Обслуживание; 5. Смерть.

Рис.13.3. Схема модели ЖЦ для подхода XP

Фаза 1

На фазе 1 выполняется предварительная подготовка к работе, в рамках которой можно выделить следующие операции.

Команда:

- выбирает инструменты,
- осваивает необходимые знания и навыки,
- анализирует варианты архитектур системы,
- оценивает будущие задачи.

Заказчик:

- готовит истории, сразу же обсуждаемые командой, формируя из них базисный набор историй, который будет реализован в выпуске продукта.

Фаза 2

На фазе 2 выполняются планирование работы по созданию системы и планирование текущего выпуска продукта. Планирование работы предполагает в частности соглашение о сроках поставки первого выпуска. Планирование выпуска основано на отборе приоритетных историй из базисного набора, которые будут реализованы в очередном выпуске.

Планирование работы и выпуска основано на наблюдении за четырьмя ограничениями, названными в ЭП переменными:

- стоимость,
- время,
- качество,
- содержание.

Заказчиком должно фиксироваться не более трёх переменных, а команда выбирает результирующие значения остальных

Фаза 3

- **На фазе 3 выполняются итерации разработки и функциональное (приёмочное) тестирование системы.** Итерации разработки предназначены для реализации историй в рамках плана выпуска и формирования функциональных тестов. В ходе первой итерации формируется общий дизайн, для которого отбираются подходящие истории, для остальных итерации отбор историй выполняется заказчиком. Функциональное тестирование позволяет получить одобренный заказчиком выпуск, включающий истории с учётом плана выпуска.

Фаза 4,5

На фазе 4 выполняются развёртывание системы в реальной среде эксплуатации и её обслуживание. Во время развёртывания и обслуживания системы заказчик может сформировать новый набор историй для их включения в следующий выпуск продукта.

На фазе 5 завершается эксплуатация системы. Это может произойти по двум причинам:

1. У заказчика нет историй для новых выпусков продукта;
2. Система находиться в плохом состоянии из-за большого числа дефектов.

4 деятельности, связанные с программированием

На протяжении этих фаз ЖЦ выполняются 4 деятельности, связанные с программированием:

1. Кодирование;
2. Тестирование;
3. Слушание;
4. Проектирование.

Основой для всех деятельностей является программный код системы.