Лекция 11

Подходы разработки ПО

MSF – Microsoft Solutions Framework

- *Каркас решений Microsoft* или *Фреймворк для создания решений от Microsoft* (МСФ, MSF – Microsoft Solutions Framework) – каркасный подход, предлагаемый фирмой Microsoft Corporation. MSF 1.0 был представлен в 1993 г. MSF 4.0 выпущена в 2005 г.

Microsoft Solutions Framework

- Microsoft Solutions Framework является также продуктом, предоставляемым Microsoft.
- В этом качестве он представляет собой базу знаний в виде пакета руководств, разделённого на несколько *белых книг* – документов, каждый из которых охватывает определённую модель или дисциплину. Он входит в набор инструментальных средств Microsoft Visual Studio Team System для поддержки МСФ.

МСФ 4.0 состоит из 5 белых книг:

- Модель руководства МСФ,
- Модель проектной группы МСФ,
- Дисциплина управления проектами МСФ,
- Дисциплина управления рисками МСФ,
- Дисциплина управления подготовкой МСФ.

Принципы МСФ

МСФ основан на наборе
из 9 основополагающих принципов:

1. Работа в рамках единого видения;
2. Проявление живости, ожидание изменений;
3. Сотрудничество с заказчиками;
4. Поощрение свободного общения;
5. Обучение на любом опыте;
6. Вкладывание [денег] в качество;
7. Поставка инкрементного результата;
8. Установление ясной подотчётности;
9. Наделение полномочиями членов команды.

Принципы формируют общую суть моделей и дисциплин МСФ.

9 ключевых концепций МСФ:

1. Фокусировка на конечном результате;
2. Поддержка своей клиентуры;
3. Чувство гордости за мастерство;
4. Просмотр всей картины;
5. Поставка на своих обязательствах;
6. Практика хорошего гражданства;
7. Поощрение команды равных;
8. Непрерывное обучение;
9. Усвоение качеств обслуживания.

В МСФ 4.0 они названы мыслеукладами из-за стремлением Microsoft к созданию и внедрению своей культуры разработки.

Модель руководства МСФ обладает следующими тремя особенностями:

1. Итеративный подход;
2. Подход, основанный на фазах и вехах;
3. Целостный подход к созданию и внедрению решений.

Модель ЖЦ

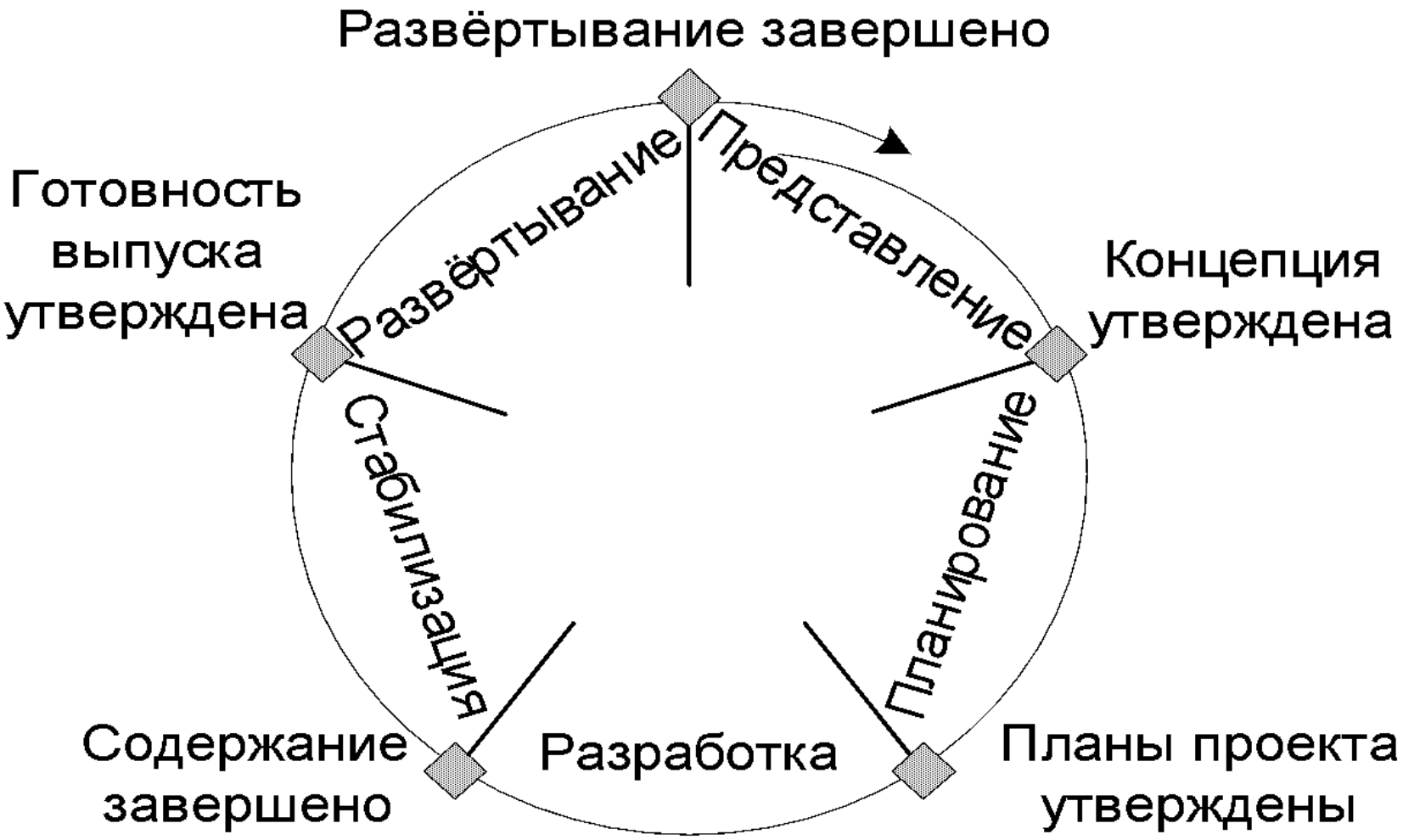
Модель ЖЦ для МСФ отражает один цикл разработки (рис.11.1).

В МСФ выделено всего 5 фаз:

1. Представление;
2. Планирование;
3. Разработка;
4. Стабилизация;
5. Развёртывание.

Все фазы разграничены главными вехами. Для повышенного управления проектом внутри фаз выделяют ряд промежуточных вех, показывающих достижение результата в некоторой деятельности.

Рис.11.1. Модель ЖЦ для подхода MSF



Фаза 1

- На фазе 1 выполняется создание и сплочение команды на основе выработки единого видения. Основными задачами являются создание ядра команды и подготовка документа с описанием концепции проекта, включающего видение и содержание проекта. Главная веха 1 считается достигнутой, если команда и заказчик пришли к соглашению об общих задачах и сроках проекта, включаемой и не включаемой в решение функциональности.

Результатами являются:

- Описание видения и содержания,
- Документ оценки рисков,
- Описание структуры проекта.

Фаза 2

На фазе 2 производится основная работа по составлению планов проекта. Она включает в себя подготовку командой функциональной спецификации, разработку дизайнов, подготовку рабочих планов, оценку проектных затрат и сроков разработки различных составляющих проекта. Главная веха 2 считается достигнутой, если заказчик и команда пришли к соглашению о составе решения и сроках поставок. Утверждённые спецификации, планы и календарные графики образуют *базовый план проекта*.

Результатами являются:

- Функциональная спецификация,
- План управления рисками,
- Сводный план и сводный календарный график проекта.

Фаза 3

- На фазе 3 команда фокусируется на создании компонентов решения. Некоторая часть этой работы может продолжаться на следующей фазе, если такая необходимость выявлена при тестировании. Эта фаза также включает в себя разработку инфраструктуры. Главная веха 3 считается достигнутой, если создание всех компонентов решения завершено и решение готово к тестированию и стабилизации.

Результатами являются:

- Исходный и исполнимый код приложений,
- Скрипты установки и конфигурирования,
- Окончательная функциональная спецификация,
- Материалы поддержки решения,
- Спецификации и сценарии тестов.

Фаза 4

- На фазе 4 производится тестирование разработанного решения. При этом внимание фокусируется на его эксплуатации в реалистичной модели производственной среды. Главная веха 4 считается достигнутой, если команда завершила разрешение всех существенных проблем и производится выпуск или внедрение решения.

Результатами являются:

- «Золотой» выпуск,
- Документация выпуска,
- Материалы поддержки решения,
- Результаты и инструментарий тестирования,
- Исходный и исполнимый код приложений,
- Проектная документация,
- Обзор вехи.

Фаза 5

На фазе 5 команда внедряет технологии и компоненты решения, стабилизирует внедрённое решение, передаёт работу персоналу поддержки и сопровождения и получает со стороны заказчика окончательное одобрение результатов проекта. По завершении внедрения команда производит анализ работы и удовлетворённости заказчика. Главная веха 5 считается достигнутой, если решение начало давать заказчику результат, а команда может свернуть свою деятельность.

Результатами являются:

- Информационные системы эксплуатации и поддержки,
- Процедуры и процессы,
- Базы знаний, отчёты, журналы протоколов,
- Версии проектных документов, массивы нагрузки и код, разработанные во время проекта,
- Отчёт о завершении проекта,
- Окончательные версии всех проектных документов,
- Показатели удовлетворённости заказчика и пользователей,
- Описание последующих шагов

МСФ

Следует сделать следующие замечания по этой модели ЖЦ:

- Длительность фаз не одинакова.
- Деятельность может выходить за рамки одной фазы.
- Наличие / отсутствие некоторых фаз определяется выполняемым проектом.

Таким образом, МСФ предлагает модель ЖЦ, основанную **на распределении работ в команде проекта по фазам,** а не на выделении процессов.

Процесс ICONIX (ICONIX Process)

- *Процесс ICONIX* (ICONIX Process) – каркасный подход, предлагаемый фирмой ICONIX Software Engineering, Inc. Название этого подхода официально не является аббревиатурой, хотя и пишется прописными буквами.
- В 1992 г. Д. Розенберг разработал подход Процесс ICONIX. В него были включены отобранные им приемлемые методы из методик Г. Буча, Дж. Рамбо и А. Якобсона.

Процесс ICONIX

- Общая идея подхода состоит в минимизации времени, требуемого для преобразования требований к системе в работающий код этой системы. Это достигается отбором только основных моделей UML, с помощью которых за 4 этапа выполняется необходимое преобразование. Таким образом, Процесс ICONIX является упрощённым подходом, ориентированным на моделирование при анализе и проектировании. При этом упрощённость не приводит к снижению строгости разработки, а связана с облегчением разработки при применении этого подхода.
- В качестве средств поддержки подхода используется инструментальное средство Enterprise Architect самой фирмы.

Процесс ICONIX

Основные особенности:

1. Упрощённое использование UML;
2. Высокая степень отслеживаемости;
3. Итеративность и инкрементность моделей.

Сутью Процесса ICONIX является понимание того, что построение хороших моделей объектов является простым, если сосредоточиться на нахождении ответа на фундаментально важные вопросы о разрабатываемой системе и отказаться от рассмотрения излишних, ненужных проблем моделирования. Этого можно достигнуть, **если придерживаться направления разработки от требований пользователя и модели ПрО к работающему коду.**

Ключевые принципы ICONIX

Разработка в рамках подхода выражается в виде трёх ключевых принципов:

- Снаружи внутрь,
- Изнутри наружу,
- Сверху вниз.

Принцип «снаружи внутрь» определяет движение внутрь, исходя из требований пользователя, формализуемых в виде сценариев и прецедентов.

Принцип «изнутри наружу» задаёт движение вовне, исходя из основных абстракций ПрО, образующих соответствующую модель.

Принцип «сверху вниз» обозначает движение вниз – от высокоуровневых моделей к детализированному дизайну.

- Модель ЖЦ для Процесса ICONIX отражает построение моделей во всех этапах ЖЦ, связанных с анализом и проектированием (рис.11.2).

Рис.11.2. Модель ЖЦ для Процесса ICONIX

Процесс ICONIX

В подходе выделено 4 этапа:

1. Анализ требований;
2. Предварительное проектирование;
3. Детализированное проектирование;
4. Реализация.

Все этапы разграничены вехами, служащими для обзора работы, выполненной командой на соответствующих этапах.

Этап 1

На этапе 1 выполняются следующие действия:

- Определяются объекты ПрО и выясняются связи обобщения и агрегации между ними.
- На основе этого начинается создание высокоуровневых диаграмм классов – модели (сущностей) ПрО. Если возможно, строится прототип предполагаемой системы (модель пользовательского интерфейса). Или собирается вся необходимая информация об унаследованной системе, которую нужно реорганизовать.
- Определяются прецеденты в виде диаграмм прецедентов.
- Организуется группировка прецедентов в виде диаграммы пакетов.
- Размещаются функциональные требования к системе – в прецеденты и объекты ПрО.

Веха 1

Веха 1 позволяет установить, что:

- диаграммы прецедентов и модель ПрО совместно отражают все функциональные требования;
- заказчик понимает идею системы и понятно отражает её в требованиях;
- команда способна создать дизайн системы по выделенным требованиям.

Таким образом, осуществляется проверка того, что созданы все диаграммы прецедентов и модель ПрО, необходимые для создания системы.

Этап 2

На этапе 2 выполняются следующие действия:

- Формируются описания прецедентов: основной ход событий прецедента («главный поток») и альтернативные ходы событий (редко используемые варианты и ошибочные ситуации).
- Выполняется анализ робастности. Для каждого прецедента: определяется набор объектов, которые участвуют в выбранном сценарии, строится диаграмма робастности с использованием стереотипов из UML Objectory, обновляются диаграммы классов (добавляются новые объекты, а также новые атрибуты и ассоциации).
- Завершается обновление диаграмм классов. Это действие свидетельствует о завершении стадии анализа для проекта.
- Выбирается *техническая архитектура* – технологические инструментарий и платформа (в частности, язык программирования и средство построения распределённых программных компонентов).

Веха 2

Веха 2 позволяет установить, что:

- описание прецедентов и диаграммы робастности соответствуют друг другу и оба представляют правильно и полностью поведение создаваемой системы;
- модель ПрО соответствует диаграммам робастности (в частности, все объекты-сущности на диаграммах робастности представлены и в модели ПрО);
- классы-сущности включают в себя необходимые атрибуты;
- можно проследить потоки данных между именованными экранными формами системы через классы-сущности и, возможно, к лежащему в основе системы хранилищу данных.
- дизайн, разработка которого начинается, выглядит правдоподобным в контексте выбранной технической архитектуры системы.

Таким образом, осуществляется проверка того, что заданы все ключевые абстракции проблемной области, необходимые для реализации поведения системы.

Этап 3

На этапе 3 выполняются следующее действия:

- «Размещается» поведение системы. Для каждого прецедента: определяются сообщения, передаваемые объектами, сами объекты и соответствующие вызываемые методы, строится диаграмма последовательности: слева указывается текст выполняемого сценария, справа – соответствующий дизайн (сама диаграмма), обновляется диаграмма классов (добавляются новые атрибуты и операции).
- Если необходимо, строятся дополнительные диаграммы: диаграмма кооперации для основных взаимодействий между объектами, диаграмма состояний для поведения системы реального времени.
- Завершается построение статической модели добавлением информации о детальном дизайне (например, области видимости значений и паттерны).
- Командой осуществляется проверка дизайна на соответствие все предъявляемым к системе требованиям. **Это действие свидетельствует о завершении стадии проектирования.**

Веха 3

Веха 3 позволяет установить, что:

- детальный дизайн в виде диаграмм последовательности и связанных с ними диаграмм классов соответствует прецедентам;
- детальный дизайн обладает достаточной глубиной (т.е. достаточно подробен), чтобы облегчить относительно небольшой и плавный переход к коду.
- дизайн удовлетворяет заданным критериям качества, позволяющим сделать оценку с различных точек зрения.
- дизайн должен удовлетворять внутренним стандартам, определённым в организации (например, использование паттернов, механизмов доступа к базам данных), что позволяет диаграммам последовательности и детальным диаграммам классов (детальной модели) отразить реальный дизайн системы;
- стабилизированы и утверждены все требования и техническая архитектура; решены все оставшиеся проблемы дизайна.

Таким образом, осуществляется проверка того, что определено соответствие детального дизайна требованиям, представленным в виде прецедентов.

Этап 4

На этапе 4 выполняются следующие действия:

- Если необходимо, строятся диаграммы развёртывания и диаграммы компонентов, которые могут оказаться полезными в стадии реализации.
- Пишется или генерируется программный код.
- Осуществляется модульное и интеграционное тестирование.
- Совершается системное тестирование и тестирование приемлемости для пользователей. **Это действие свидетельствует о завершении стадий реализации и тестирования для проекта.**

Веха 4

Веха 4 позволяет установить, что:

- модульные тесты соответствуют описанию прецедентов и диаграммам последовательности;
- созданный программный код соответствует диаграммам классов и последовательности, рассматриваемых как руководство к написанию кода. **Таким образом, осуществляется проверка того, что определено соответствие программного кода и тестов разработанным статической и динамической моделям – структуре и поведению системы исходя из предъявленных требований.**
- Одним из преимуществ подхода является то, что для каждого этапа указано 10 наиболее распространённых ошибок разработки и их разбор.