[**Лекция 6 Методологии разработки ПО и модели жизненного цикла (часть2)** 2](#_Toc468138609)

[**6.4 Объектно-ориентированные модели (ООМ) жизненного цикла** 2](#_Toc468138610)

[**6.4.1 Принципы объектно-ориентированного проектирования в модели ООМ** 2](#_Toc468138611)

[**6.4.2 Особенности традиционных этапов проектирования в ООМ** 5](#_Toc468138612)

[**6.4.3 Модификация модели фазы-функции при переходе к ООП** 6](#_Toc468138613)

[**6.4.4 Графическое изображение ООМ** 17](#_Toc468138614)

[**6.4.5 Сравнительный анализ моделей жизненного цикла** 22](#_Toc468138615)

[**6.4.6 Модели процесса разработки программного продукта** 22](#_Toc468138616)

## **Лекция 6 Методологии разработки ПО и модели жизненного цикла (часть2)**

### **6.4 Объектно-ориентированные модели (ООМ) жизненного цикла**

#### **6.4.1 Принципы объектно-ориентированного проектирования в модели ООМ**

Коренное (главное, принципиальное) отличие ООМ от всех последовательных методологий заключается в другом отношении к итеративности развития проекта при его реализации.

Для традиционных последовательных подходов итерация[[1]](#footnote-1) заключается в исправлении допущенных ошибок. Такой процесс трудно регламентировать – то есть создать и использовать технологические нормы.

При объектно-ориентированном подходе в рамках ООМ итерации должны не отменять результаты друг друга, а всегда (в идеале) **дополнять и развивать их**.

Сопоставим главные моменты ООМ в сравнении с последовательными методологиями.

**А. Итеративность развития**

***Начиная с фазы анализа и до завершения реализации, процесс объектно-ориентирования проектирования в противоположность последовательному развитию строится как серия итераций, которой возможно предшествует определенный период последовательного изучения предметной области и задач проекта в целом (этапы определения требований и начального планирования).***

**Б. Изменение функциональности**

*В ходе развития проекта требования практически всегда пересматриваются. При этом обычно требуется изменение ранее принятых требований.* ***Отсюда следует, что необходимо реализовывать функциональность так, чтобы пересмотр ее производился с минимальными затратами****.*

**В. Формирование системы понятий проекта.**

*Система понятий проекта развивается в процессе пересмотра требований и изменения функциональности, а также в ходе* *распространения изменений. Для поддержания целостности этой системы необходимы соответствующие меры, использование специальных инструментов. Эта задача решается с помощью ведения глоссария проекта — специальной базы знаний понятий, их взаимосвязей и истории изменения в ходе**итеративного развития проекта.* ***Если глоссарий явно не предусматривается, то система понятий проекта формируется стихийно, что отрицательно сказывается на развитии проекта****. В частности, повышается риск изменения концепций вслед за изменениями в программном коде, что в результате приводит к построению концептуально рассогласованного программного изделия.*

**Г.** **Наращивание функциональности в соответствии со сценариями**

*Наращивание функциональности проектируемого изделия представляется как развитие сценариев****, которые соответствуют описаниям (диаграммам) взаимодействия объектов*** *и отражают отдельные стороны функционирования. Эти описания предписывают развитие на этапе программирования операционной базы проекта: она вырабатывается исходя из сценариев уровня проектирования (конструирования). Полная функциональность состоит из функциональностей всех сценариев. Таким образом, данная стратегия довольно близка классическому методу пошаговой детализации, при использовании которого функциональность наращивается путем уточнения (доопределения) модулей нижнего уровня. Однако в отличие от этого метода итеративное наращивание требует, чтобы в результате каждой итерации изделие получало полностью готовую функциональность, планируемую реализуемым сценарием. Последующие итерации добавляют уже другую функциональность, которая планируется другим сценарием.*

**Д. Ничто не делается однократно**

*Последовательный подход предполагает, что анализ завершен перед конструированием, завершение которого предшествует программированию. Перекрытие этапов ослабляет это предположение, но принципиально ситуацию не меняет.* ***В большинстве объектно-ориенти­ро­­ванных проектов анализ никогда не завершается в течение всего развития проекта, а процесс конструирования сопровождает разработку в ходе всего ее жизненного цикла****.*

**Е. Оперирование на размножающихся фазах**

*Как* ***в начале проектирования****, на* ***последующих итерациях***анализ ***предшествует***конструированию, *за которым следует программирование, тестирование и другие виды работ.*

**Ж. Распределение реализуемых требований по итерациям**

*Совокупность**сценариев, реализуемых на очередной итерации, и набор ранее реализованных**сценариев всегда образуют законченную, хотя и неполную версию системы, предлагаемую пользователям. По разным причинам, в том числе для исключения двусмысленностей в понимании,* ***необходимо представление планируемых для реализации средств в виде моделей, согласующих взгляд на систему со стороны пользователей (а также заказчиков и других заинтересованных лиц) с точкой зрения разработчиков****. Эти модели появляются в ходе этапа анализа, что отражается в их названии:**модели уровня анализа.*

**З. Особый стиль наращивания возможностей системы и ее развития**

*Представление системы как набора* ***связанных различными отношениями классов — основа декомпозиции проекта при******объектно-ориентированном подходе****.*

*Каждая новая итерация расширяет этот набор путем добавления новых классов, вступающих в определенные отношения с ранее построенной системой классов.*

*Выполнить такое расширение корректно без абстрагирования от деталей реализации существующего, а если учитывать перспективу, то и без такого же абстрактного представления добавляемых классов практически невозможно.*

*Иными словами,**требуется построение****моделей уровня конструирования,*** *которые задают реализационное представление проектируемой системы.*

*В приведенном выше перечне этапов жизненного**цикла итерации при**объектно-ориентированном подходе явно выделено* *моделирование уровня анализа, которое сводится к построению модельного представления**сценариев.*

*Но это только один аспект проектного моделирования. Как было только что показано, другой, не менее существенный аспект моделирования, проявляется при конструировании.*

*Наконец, есть еще третий аспект моделирования,* ***связанный с предъявлением каждой версии программного изделия пользователю****,**представление которого о системе, разумеется, не имеет отношения к**моделям уровня конструирования и лишь косвенно связано с**моделями уровня анализа. Таким образом, если следовать гантеровскому стилю описания жизненного**цикла, то правильнее будет выделять не этап моделирования (как это, следуя уже сложившейся традиции, чаще всего и делают), а* ***организационно-техническую (производственную) функцию моделирования****, пронизывающую весь процесс разработки проекта*

#### **6.4.2 Особенности традиционных этапов проектирования в ООМ**

При объектно-ориентированном проектировании в ходе итеративного наращивания обыкновенно выполняются вполне традиционные этапы. Опишем их с учётом особенностей ООМ.

***Определение требований, или планирование итерации*** — фиксируется, что должно быть выполнено на **данной итерации** в виде описания области, для которой планируется разработать функциональность на данной итерации, и что для этого нужно. Обычно этот этап включает отбор сценариев, которые должны быть реализованы на данной итерации.

***Анализ***— исследуются условия выполнения планируемых требований, проверяется полнота отобранных сценариев с точки зрения реализации требуемой функциональности.

***Моделирование пользовательского интерфейса***— коль скоро итерация должна обеспечивать функционально законченную реализацию, требуется определить правила взаимодействий, необходимые для активизации требуемых функций. Модель интерфейса представляет пользовательское представление поведения объектов данной итерации.

***Конструирование***— обычная декомпозиция проекта, проводимая в объектно-ориентированном стиле. Конструирование включает построение или наращивание иерархии системы классов, описание событий и определение реакции на них и т. д. В ходе конструирования определяются объекты, реализуемые и/или доопределяемые на данной итерации, и набор функций (методов объектов), которые обеспечивают решение задачи данной итерации.

***Реализация*(программирование)** — программное воплощение решений, принятых для данной итерации. Необходимым компонентом реализации здесь считается автономная проверка соответствия составляемых модулей их спецификациям (в частности, должно быть обеспечено требуемое поведение объектов).

***Тестирование*** — этап комплексной проверки результатов, полученных на данной итерации.

***Оценка результатов итерации*** — этап включает работу, связанную с рассмотрением полученных результатов в контексте проекта в целом. В частности, должно быть выяснено, какие задачи проекта можно решать с учетом результатов итерации, на какие ранее поставленные вопросы получены ответы, какие новые вопросы возникают в новых условиях.

Фазовое измерение жизненного цикла при объектно-ориентированном проектировании почти не изменяется: появляется лишь один дополнительный этап: «***Моделирование пользовательского интерфейса***», **который в старой схеме можно рассматривать как часть этапов анализа и/или конструирования**. Однако это весьма существенное дополнение, характеризующее подход в целом. Главный мотив явного рассмотрения моделирования в жизненном цикле при объектно-ориентированном развитии проектов связан с особенностями **Ж** и **З** ООМ.

#### **6.4.3 Модификация модели фазы-функции при переходе к ООП**

Представление модели ЖЦ в ООМ представлено на рис. 6.2.1.

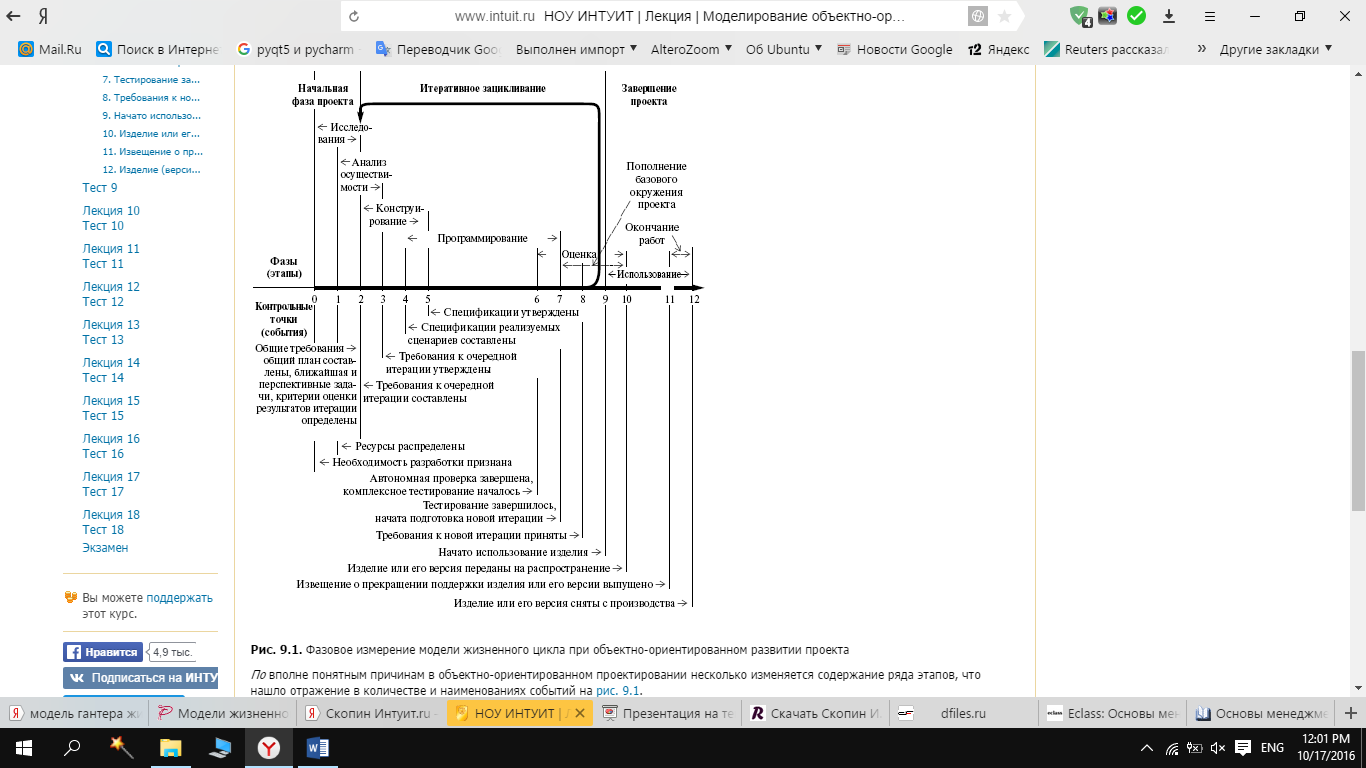


Рис. 6.2.1. Фазовое измерение модели жизненного цикла при объектно-ориентированном развитии проекта

Рассмотрим содержание основных контрольных точек

**Точка 0. Необходимость разработки признана.**

Эта точка означает начало работ над проектом, которые еще не утверждены как официальный проект. Официальным он становится, когда доказана его осуществимость, задачи проекта в целом обозначены, в том числе определены те задачи, которые предстоит решать в первую очередь ( *контрольная точка* 2). В этот период можно говорить лишь о предпроектной деятельности менеджера. Если схема развития проекта допускает распределение менеджерских обязанностей в команде исполнителей и есть команда, которая принимает решение о целесообразности работ над проектом, предпроектная деятельность менеджера становится сферой ответственности команды целиком. Но и при таких схемах эта деятельность может быть персонифицирована, когда предполагаемый менеджер готов взять на себя обязательства, имея в виду свои планы по использованию некоторого сложившегося коллектива, по формированию команды и др. Собственно говоря, один из главных менеджерских вопросов, который нуждается в решении в обсуждаемой контрольной точке, — стратегия и тактика формирования проектной команды.

**Точка 1. Ресурсы распределены.**

Для начала работ над проектом необходимо знать, какими *ресурсами* можно располагать. В данной точке фиксируется, что предполагает планировщик предоставить менеджеру, оцениваются средства, которые разумно ожидать от заказчика и от инвесторов. Эти сведения есть результат соответствующей деятельности менеджера. В частности, он определяет основы *концепции развития проекта*, мотивирующие потребность.

**Точка 2. Требования к очередной итерации сформулированы, общие требования и общий план составлены, ближайшая задача, критерии оценки результатов и перспективные задачи определены.**

Эта *контрольная точка* обозначает начало **стационарного цикла развития проекта**, когда исходя из имеющейся информации планируется ход текущей итерации. Для первой итерации эта информация собирается на **начальной фазе проекта**, которая в данный момент завершается. Исходная информация для последующих итераций извлекается из всей предшествующей истории проекта, она формируется в доступном для работы виде в ходе этапа оценочных работ предыдущей итерации к моменту *расщепления жизненного цикла*.

Отсутствие опыта работы над проектом при выполнении первой итерации проявляется в ее следующей особенности: у менеджера нет оснований для выработки надежных планов и достоверных критериев. Вместо этого он вынужден пользоваться оценками и применять стратегии, настраиваемые *по* ходу дела. Для новой команды в это время закладывается основа ролевой специализации и разделения труда исполнителей. Все это указывает на целесообразность выделения **начального периода проекта**, включающего в себя предпроектную *деятельность*, начальную фазу проекта, работы над первой итерацией и подведение итогов. Для этого периода характерны специальные методики, важной стороной которых является обучение и настройка на стационарный режим. Существенно, что в *начальный период* складывающаяся проектная *команда* должна доказать свою работоспособность и продемонстрировать продуктивность принятых стратегий.

Как для первой, так и для всех последующих итераций в контрольной точке 2 должны быть определены:

* ***общие требования*** — что требуется от проекта в целом в данный момент;
* ***общий план*** — как предполагается достигать поставленных целей;
* **ближайшая задача** (должна быть четко определена) — задача текущей итерации, решаемая в проекте с целью предложения пользователю полученных результатов в качестве продукта, которая задана как набор конкретных реализуемых требований и *сценариев*.

Для отбора того, что предполагается реализовать на итерации, есть несколько **критериев предпочтения**. В реальной ситуации их сравнительная важность зависит как от специфики проекта, так и от того, как далеко на данный момент продвинулись работы над проектом. Вот перечень наиболее типичных критериев:

1. **Актуальность для пользователя**. С точки зрения бизнеса важно, чтобы в первую очередь в рабочем продукте предоставлялась функциональность, которая позволила бы устранить узкие места автоматизируемой деятельности. Этот критерий практически всегда рассматривается как наиболее существенный.
2. **Полнота и функциональная замкнутость предлагаемых средств**. Связан с предыдущим критерием. С его точки зрения, отбирается для реализации такой комплект средств системы, который автоматизирует какой-либо вид пользовательской деятельности в полной мере, не требуя чего-то еще. Важность критерия растет по мере увеличения объемов уже выполненных в проекте работ (обсуждение критерия см. ниже).
3. **Системная значимость**. Этот критерий отражает внутренние для проекта предпочтения. В первую очередь целесообразно реализовывать функциональность, которая полезна для развития: заготовки широкого применения, базовые и инструментальные компоненты для других предоставляемых средств системы. С точки зрения развития проекта решение ближайшей задачи должно обеспечить осуществимость последующего *итеративного наращивания* возможностей системы (об этом разговор еще предстоит). Данный критерий конкурирует с актуальностью для пользователя. Он является более значимым для начальных итераций проекта.
4. **Демонстрационная значимость**. С точки зрения этого критерия первоочередными считаются те средства, которые в состоянии продемонстрировать работоспособность и лучшие, выгодные для показа качества системы. Наибольший удельный вес данный критерий имеет для начального периода развития проекта, причем не только для заказчика, но и для разработчиков, которые при соответствующей трактовке демонстрационности будут в состоянии убедиться в правильности выбранной стратегии и методов. **Критерий может противоречить и пользовательской актуальности, и*****функциональной полноте*, и системной значимости. (!!!!)**
5. **Скорость реализации**. По этому критерию выделяются те виды работ, которые можно реализовать за самое короткое время. Если же время итерации фиксировано, то для выполнения отбирается пул работ, которые в состоянии завершить команда в заданные сроки.

Что касается затратных характеристик и объема выполняемых *работ*, то чаще всего они выступают в качестве ограничений, в которые необходимо уложиться. В ряде случаев выделяемое время также целесообразно рассматривать не как основу критерия, а как ограничение. Выбор того, что предлагается реализовать, осуществляется в соответствии **со схемой треугольника менеджмента**.

Критерий полноты следует прокомментировать особо. Различают три аспекта полноты, которые отражают три составляющие качества программной системы:

* функциональная полнота — соответствие предлагаемых средств набору пользовательских функций, которые могут быть активизированы в рамках автоматизируемой деятельности;
* реализационная полнота — возможность обеспечения функциональной полноты путем комбинации базовых средств системы, ее реализационных механизмов, а также используемых средств окружения;
* интерфейсная полнота — задание языка управления поведением системы, адекватного восприятию системы пользователем.

В обсуждаемом критерии речь идет о *функциональной полноте*. Функциональная замкнутость означает, что *автоматизация* деятельности, осуществляемая релизом данной итерации, достигается таким путем, который не требует от пользователя ничего, что выходит за рамки средств, предлагаемых релизом системы. Эти качества обеспечиваются реализационной полнотой, а на уровне управления поведением системы — интерфейсной полнотой. Таким образом, дополнительные критерии не требуются: и реализационная, и интерфейсная *полнота* определяются как *производные* от функциональной полноты.

Иногда говорят о минимальности того, что должно быть реализовано на итерации для достижения полноты и функциональной замкнутости. Это подчеркивается, например, в подходе *экстремального программирования* [[3](http://www.intuit.ru/studies/courses/38/38/literature#literature.3)]. Однако реализационная минимальность есть качество, которое характеризует конкретную методологическую стратегию ведения проекта, а потому на уровень общих для любых методологий критериев ее выносить не стоит.

Для ближайшей задачи первой итерации критерии упорядочиваются следующим образом: демонстрационная *значимость*, пользовательская актуальность, скорость реализации, системная *значимость* и *функциональная полнота*. Искусство менеджера состоит в том, чтобы в заданных ограничениях соблюсти необходимый баланс предпочтений и обеспечить возможность проверки и корректировки априорных методик, метрик, стратегий и прогнозов.

Для ближайшей задачи должны быть определены:

* планы реализации — разбиение решения на этапы, сроки их выполнения и ресурсные потребности, определяемые для реализации конкретных требований ближайшей задачи;
* ***критерии оценки результатов*** — методики, с помощью которых определяется, что конкретные требования ближайшей задачи реализованы в срок с необходимым качеством;
* ***перспективные задачи***, формулировка которых в проектах жесткой отчетности является обязательным результатом, проверяемым в контрольной точке 2. Их предполагается решить на последующих итерациях (степень проработки требований к *перспективным задачам* и даже их идентификация могут быть различны).

Последнее не требуется, например, в схемах *экстремального программирования*, однако и для подхода быстрого развития полезно выработать гипотезы о перспективах проекта, которые нужно будет иметь в виду на следующих этапах (итерациях).

В контрольной точке 2 определяется фронт *работ* проектировщиков подсистем.

**Точка 3. Требования к очередной итерации утверждены.**

Все сведения о проекте, представленные в контрольной точке 2, к моменту прихода к третьей контрольной точке должны быть согласованы для утверждения. Для больших и сложных проектов данный момент определяется формально, для него заданы отчетные материалы, которые утверждаются. В более простых случаях этого не требуется, тем не менее веха, знаменующая окончание аналитического этапа проекта (итерации), существует независимо от формальной стороны дела. Ее отслеживание просто необходимо менеджеру в качестве момента, когда подводятся первые итоги проекта (итерации).

**Точка 4. Спецификации реализуемых сценариев составлены.**

Начало этапа конструирования связывается с декомпозицией решаемых проектом (итерацией) задач и с построением архитектуры системы.

Коль скоро архитектура определена, пусть даже лишь в общих чертах, появляется фронт работ для разработчиков подсистем. Соответственно, у *руководителей команд* разработчиков появляется сфера ответственности для текущей итерации. Менеджерские обязанности в проекте, которые становятся главными в этой точке, — оформление подготовленных для реализации *сценариев* к утверждению. Как и в предыдущем случае, данная *контрольная точка* вполне может быть явно не выделена, но это не означает, что работа над трансформацией *сценариев* в архитектуру растворяется в проекте.

**Точка 5. Спецификации утверждены.**

Эта *контрольная точка* обозначает окончание этапа конструирования. Архитектура для очередной итерации утверждена и зафиксирована в виде заданий для разработчиков подсистем и их руководителей, от которых требуется создание или модернизация наборов классов проекта, относящихся к их сферам ответственности.

**Точка 6. Автономная проверка завершена и комплексное тестирование началось.**

По мере продвижения этапа программирования к завершению возникает момент, когда требуется комплексная проверка работоспособности системы. Он означает начало этапа оценки, поскольку с этого момента появляется возможность проверить предварительные суждения о проекте (итерации) на практике.

Часто говорят, что для проектов, разрабатываемых по методикам быстрого развития, нет нужды выделять этап конструирования (см., например, [[3](http://www.intuit.ru/studies/courses/38/38/literature" \l "literature.3)]). Как следствие, контрольные точки 3, 4 и 5 сливаются. Все, что надо сделать в проекте в связи с требованиями к очередной итерации, приурочивается к контрольной точке 6. К тому же в этом подходе не разделяются автономная и комплексная проверка результатов, поскольку каждая новая возможность системы с самого начала интегрируется с уже реализованными возможностями. Как следствие, автономная разработка остается лишь на уровне продумывания того, каким образом встроить новую возможность в складывающийся архитектурный каркас.

Если иметь в виду мероприятия, которые предполагается осуществлять при прохождении указанных точек, то методы быстрого развития действительно их игнорируют. Однако соответствующие процессы, о которых только что шла речь, отменить не удается. Разработчики в любом случае вынуждены заботиться и о точном специфицировании того, что требуется реализовать, и об архитектуре системы в целом. По существу, экономия времени достигается, когда слияние контрольных точек не противоречит, например, сложности проекта, перспективам его развития и т.д., и именно за счет того, что ликвидируется сдерживающий регламент развития системы. К сожалению, это далеко не всегда оправданно.

**7. Тестирование завершилось, начата подготовка новой итерации.**

Эта *контрольная точка* отмечает окончание этапа программистских работ, определенных текущей итерацией. Программистские работы продолжаются как деятельность, связанная с *пополнением базового окружения проекта*, которое мы выделяем как относительно самостоятельный этап проекта, вложенный в этап оценки, и обычно завершающийся вместе с ним ( *контрольная точка* 10). Эти работы подразделяются на два вида:

* выделение общих лишь для данного проекта переиспользуемых компонентов (его целесообразно выполнить до *расщепления жизненного цикла* — *контрольная точка* 9);
* выделение общих, не привязанных к проекту переиспользуемых компонентов (целесообразно начинать эти работы, когда программный рабочий продукт итерации рассматривается как **готовое приложение**, — *контрольная точка* 9, и завершать к моменту передачи системы на распространение — *контрольная точка* 10).

*Итеративное развитие проекта* обеспечивается сбором сведений для новой итерации, которая начинается в рассматриваемой контрольной точке.

**8. Требования к новой итерации приняты.**

В ходе обработки сведений для новой итерации возникает момент, когда их можно оформить как принимаемые требования. Набор таких требований пополняется в течение первого периода использования силами разработчиков (этот период обычно называют альфа-тестированием). **По его завершении происходит*****расщепление жизненного цикла* с целью параллельного выполнения передачи системы в эксплуатацию и организации работ новой итерации**.

Расщепление приводит к одновременному существованию двух версий системы: одна из них начинает использоваться (отменяет ли она другие, ранее существовавшие версии — вопрос специфики проекта), а вторая зарождается в виде набора требований. Это время, когда должна быть произведена ревизия априорных гипотез, предположений, корректировка показателей и нормативов проекта.

**9. Начато использование изделия.**

Эта *контрольная точка* обозначает начало так называемого бета-тестирования, т.е. передачу системы в эксплуатацию для внешнего оценивания. После выполнения соответствующих работ, и в частности исправления обнаруженных дефектов, можно переходить к распространению результатов итерации (контрольная *точка* 10).

*Контрольная точка* 9 отмечает начало так называемой **фазы завершения проекта (итерации)**. Она охватывает часть жизненного цикла, которая отражает деятельность разработчиков, связанную рабочими продуктами итерации после получения результатов. Фаза завершения подобна традиционной фазе эксплуатации и сопровождения, однако есть и отличия, обусловленные тем, что объектно-ориентированный проект обычно имеет дело с иерархиями версий системы, отражающими наращивание возможностей. Данная фаза пересекается с этапом оценки.

Традиционные работы фазы завершения включают в себя:

* *поставку или пакетирование изделия* для потребителя ( *контрольная точка* 9);
* сопровождение программного продукта (по причине разнообразия вариантов организации этих работ они редко описываются структурно, т.е. с разбиением на этапы);
* этап окончания работ (см. ниже — контрольные точки 11, 12).

**Точка 10. Изделие или его версия переданы на распространение.**

Этап использования в данной контрольной точке приобретает новое качество: у версии появляется круг пользователей, нуждающихся в обслуживании.

**Точка 11. Извещение о прекращении поддержки изделия (версии) выпущено.**

Начало этапа окончания работ: оповещение о прекращении сопровождения и сворачивание деятельности по поддержке версии или всех версий к определенному сроку.

Предварительное оповещение о наступлении этапа необходимо пользователям, чтобы они смогли либо перестроиться, либо найти аргументы (в часности, ресурсы) в пользу продолжения сопровождения изделия.

**Точка 12. Изделие (версия) снято с производства.**

Завершение этапа окончания работ: прекращение сопровождения и сворачивание деятельности по поддержке версии или всех версий. После получения извещения о прекращении поддержки изделия (версии) некоторые пользователи, возможно, захотят отложить (на определенный или неопределенный срок) наступление данного события. Если они в состоянии финансировать поддержку, сопровождение и, возможно, развитие изделия, то срок окончания работ передвигается. Это обычная процедура, благодаря которой продолжают жить и даже совершенствоваться устаревающие системы. Причины откладывания прекращения поддержки вполне понятны: это стремление не потерять своих пользователей пусть даже ценой существенных дополнительных затрат. Классический пример тому — уникальное долгожительство языка Fortran.

***Этап окончания работ*** мог бы быть представлен во всех традиционных моделях, но в то время, когда эти модели разрабатывались, ему не придавали особого значения. Вместе с тем, когда речь идет о совместной поддержке нескольких версий (а именно такая ситуация типична для объектно-ориентированного проектирования), *окончание работ* игнорировать нельзя.

Одним из существенных моментов адаптивных методологий, который учитывается и при обычном объектно-ориентированном проектировании, является отказ от традиционного постулата о том, что все требования к системе сформулированы заранее. Как указывает М. Фаулер , предсказать направление целесообразного развития программного проекта чаще всего невозможно, а потому следует строить адаптивные процессы разработки, что применительно к моделированию жизненного *цикла* вообще и его фазы завершения в частности означает необходимость учета обработки потока внешних требований на всех этапах. Этому вопросу еще будет уделено внимание, а пока можно считать (как чаще всего и бывает), что требования, поступающие на фазе завершения итерации, рассматриваются как относящиеся к следующим итерациям, т.е. к следующим версиям системы. В таком случае завершение итерации означает сопровождение программного изделия, а затем — окончание работы с данной версией. Пожелания к развитию проекта в этот период учитываются как требования к последующим (возможно, еще не начатым) итерациям. Окончание проекта рассматривается как отказ от сопровождения всех версий системы. Поучительно сопоставить это положение с традиционными подходами к проектированию, когда учет пожеланий к системе в процессе ее эксплуатации чаще всего означает одно: организацию нового проекта (быть может, специального), цель которого — учет новых требований.

Несколько слов о функциональном измерении в модифицированной для *объектно-ориентированного подхода* матрице фазы—функции. Как было показано выше, целесообразно *список* производственных функций расширить за счет моделирования. Соответственно, следует определить в матрице Гантера строку интенсивностей для этой функции. В предположении о сохранении распределения интенсивностей других функций (см. [рис. 8.2](http://www.intuit.ru/studies/courses/38/38/lecture/1132?page=4)) распределение интенсивности для модифицированной *модели жизненного цикла* можно задать так, как это сделано на [рис. 9.2](http://www.intuit.ru/studies/courses/38/38/lecture/1132?page=4#image.9.2). На рисунке показан новый вид модели целиком (указаны номера контрольных точек жизненного *цикла* без пояснений).

Представленные распределения интенсивностей нельзя абсолютизировать. Наивно было бы предполагать *стабильность* интенсивностей технологических функций *по* итерациям. Следовательно, весь цикл развития проекта в матричном, двумерном представлении модифицированной гантеровской модели изобразить не удастся: оно не может показать изменение интенсивностей технологических функций при переходе от одной итерации к другой. *По* этой причине предлагается распределение интенсивностей производственных функций рассматривать как "среднестатистическую" интегральную *по* итерациям тенденцию. Практическая *полезность* рассмотрения функционального измерения — не в конкретном распределении интенсивностей функций в реальных проектах, а в том, что оно заставляет руководство проекта думать о расстановке сил в коллективе разработчиков и вообще о правильном распределении кадровых и других *ресурсов проекта*.

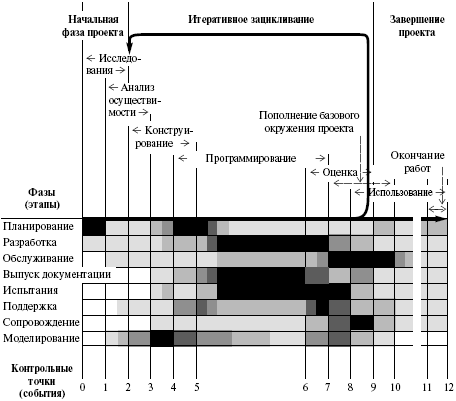


Рис. 6.2.2. Модель фазы—функции, модифицированная для объектно-ориентированного развития

#### **6.4.4 Графическое изображение ООМ**

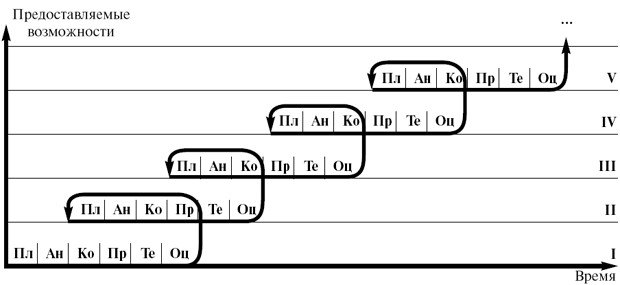


Рис. 6.2.3. Спираль развития

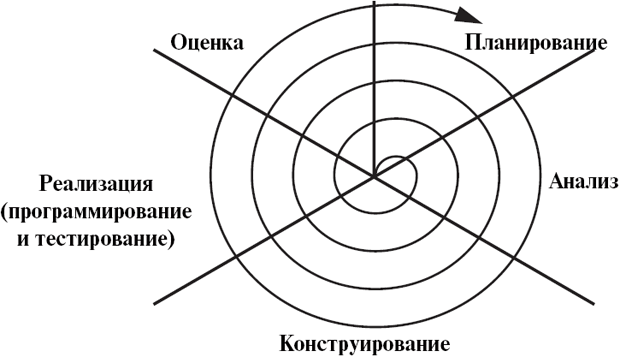


Рис. 6.2.4. Спираль охвата предметной области

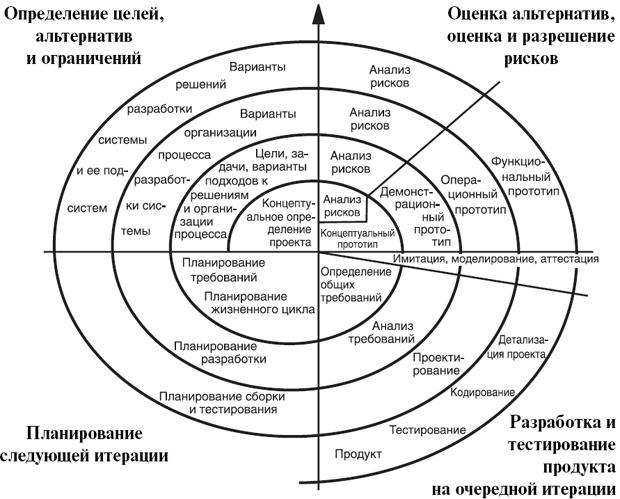


Рис. 6.2.5. Модель раскручивающейся спирали по Боэму

***Спираль Боэма*** раскручивается на плоскости, разделенной на четыре квадранта, каждый из которых отвечает за определенный круг проектных работ. Витки спирали обозначают развитие проектной деятельности от центра (начало проекта) к периферии. Некоторые из витков приводят к построению программного продукта, который предъявляется в качестве текущего результата проекта. Тем самым выделяется проектная итерация и ее релиз в качестве текущего рабочего программного продукта.

В квадрантах раскручивания спирали предусматриваются следующие действия (в описании их порядок соответствует движению поспирали от центра).

* **Определение целей, альтернативных вариантов и ограничений**.

На каждом витке спирали разработчики должны явно определить цели и варианты их достижения в условиях фиксированных ограничений. В этом квадранте начинается проект, т.е. определяется цель проекта и все аспекты условий его выполнения.

Последующие витки в данном квадранте должны сделать явными те аспекты проекта, которые нуждаются в исследовании и в разработке. Например, в какой-то момент признается актуальной реализация некоторой функции. Возможные варианты решения: использование программного кода, разработанного вне данного проекта, адаптация подобного программного кода, разработка своей реализации исключительно как компонента конструируемой системы или как переиспользуемого компонента. В качестве вариантов процесса (здесь речь идет только об одном из его аспектов) уместно рассмотреть реализацию своими силами, привлечение дополнительных сотрудников, субподряд.

Модель не фиксирует количество витков, которые нужно пройти до того момента, который отмечает построение релиза, пригодного для передачи в эксплуатацию. Это зависит от сложности проекта, от уровня предварительной проработки сведений о проблемах пользователей, от других факторов. Считается, что планирование каждого из витков отмечает реальную проблему проекта, которую требуется решить обособленно. Находясь в квадранте определения целей, важно проводить исследования так, чтобы получать полную картину всего спектра возможных решений задачи, поставленной на соответствующем витке спирали.

* **Оценка альтернативных вариантов, идентификация и разрешение рисков**.

В этом квадранте разработчики оценивают информацию, полученную при определении целей, вариантов и ограничений. Метод *оценки*, предлагаемый Боэмом, заключается в анализе *рисков*, на основе которого готовится *прототип*для изучения ситуации. В качестве *прототипов* могут выступать программы, документы, схемы и другие материалы. *Прототип* либо разрабатывается специально (чаще всего), либо заимствуется из окружения проекта (для хорошо проработанных задач всегда можно найти набор решений, пригодный для анализа).

*Оценка* должна затрагивать все стороны проектного решения, относящиеся как к пользовательским нуждам и *рискам* (неоправданные ожидания, потери инвестиций и др.), так и к конкретным методам и приемам. Не следует упускать из виду кадровые обстоятельства проекта: уровень опыта и квалификации команды, необходимость организации обучения и т.п. Основанием для *оценки* служит изучение *прототипов*.

Модель не регламентирует характер прототипирования и изучения *прототипов*. Предполагается, что разработка *прототипа* требует затрат меньше, чем прямая реализация проверяемого решения.

Один из подходов к прототипированию заключается в построении наращивания возможностей уже построенной части системы (на самом внутреннем витке это лишь стартовое видение проекта). Полученный результат может стать новым работоспособным релизом. В этом случае *схема Боэма* оказывается одним из вариантов модели итеративного развития проекта, в котором каждый релиз рассматривается как *прототип* для последующих версий.

*Прототип* очень редко оказывается тем результатом, который можно использовать как непосредственное приращение возможностей. Обычно предполагается сначала черновая реализация, возможно, несколько таких реализаций, а уж потом (в следующем квадранте) строится реализация, отвечающая требованиям предъявления заказчику результатов.

*Прототип* внутреннего витка обычно определяет концептуальное рассмотрение проекта. Этот *рабочий продукт* фиксирует решения, которые позволяют планировать процесс развития проекта.

Обычно *прототип* второго витка направлен на то, чтобы все действующие лица проекта и его окружения убедились в работоспособности принятых соглашений и проектной команды. Поэтому данный *прототип* чаще всего оказывается демонстрационным (мы уже рассматривали особенности демонстрационных рабочих продуктов, когда обсуждали вопросы типичных мероприятий в контрольных точках).

Последующие витки и их *прототипы* относятся к конкретизированным задачам, возникающим при воплощении выработанных концепций.

* **Разработка и тестирование продукта на очередной итерации**.

В этом квадранте выполняются работы, которые имеют прямые аналоги этапам традиционного последовательного развития проектов. Они могут быть разнесены по виткам спирали, если требуется провести исследование для некоторого аспекта проектируемой системы. В этом случае *рабочий продукт*, который можно считать релизом, появляется, когда будут пройдены все витки, вложенные в итерацию (на рис. 6.2.5 показано, что выпуск релиза для данного проекта запланирован на четвертом витке).

Принципиально, что работы в данном квадранте начинаются с верификации и аттестации *прототипных* решений (сектор "Имитация, моделирование, аттестация"). Только после преодоления этого барьера разрешается дальнейшее движение по спирали. Это позволяет быть уверенным, что процесс организован должным образом и удастся избежать хаоса, что принимаемые решения удовлетворяют соответствующие потребности, что нет случайных решений.

Результаты прототипирования на всех витках одной итерации выстраивают продвижение разработки в направлении получения продукта, предъявляемого заказчику. Само предъявление может быть порционным, т.е. разбитым на части, относящиеся к виткам. На последнем витке итерации принимается решение о том, что релиз как продукт разработки готов к передаче в эксплуатацию.

* **Планирование следующей итерации**.

Этот квадрант отвечает за планирование работ проекта, которое зависит от накопленной к текущему моменту информации. Принципиально, что для работ планирования на основе достигнутых результатов привлекаются представители заказчика. *Оценка* результатов всеми заинтересованными инициаторами работ позволяет выстроить правильные критерии, откорректировать позиции, связанные с актуальностью задач, пересмотреть текущее представление о требованиях, получить новые требования.

На вложенных витках итерации определяются моменты, когда можно будет приступать к выпуску релиза. На последнем витке итерации планирование развития проекта выполняется исходя из сведений о выпущенном релизе. В этот момент возможно изменение стратегии реализации, в частности изменение модели жизненного цикла разработки последующих релизов.

*Модель Боэма* достаточно глубоко проработана с точки зрения процессов, возникающих в производстве программной продукции. Предусматривается, в частности, возможность настройки ее на конкретные методологии. Важная особенность модели состоит в том, что она явно указывает на действия, которые требуется выполнять при движении *по* спирали.

С точки зрения представленных выше критериев *инструментальности* модель можно охарактеризовать следующим образом. Основное достоинство модели связывается с ее *расширяемостью*. Обсуждая проект, разработчики могут выстраивать стратегии достаточно глубоко и привязывать к спирали свои этапы, определяемые потребностями стратегии. Важно, что детализированное рассмотрение процесса достаточно ограничивать уровнем одной итерации или даже одного витка. Более поздние витки можно только обозначить, чтобы конкретизировать их, когда появится необходимая *информация*. *По* этой причине для данной модели *масштабируемость* не считается существенным свойством — при продвижении проекта к результатам его фактический *цикл разработки* целенаправленно уточняется. Конкретное планирование действий каждого из витков для данной модели не рассматривается. Оно осуществляется на более низком уровне, нежели *разбиение* жизненного *цикла* на этапы. За счет этого *атрибутивность* для *модели Боэма* становится не очень важным свойством — *доступ* ко всем нужным атрибутам осуществляется при обращении к моделям уровня конкретного планирования.

**Таким образом,*****модель Боэма* вполне можно рассматривать в качестве инструмента поддержки управления проектом на верхнем уровне**.

Подключение пользователей к определению перспектив проекта, *анализ* *рисков* и подготовка к их нейтрализации с помощью *прототипов* — все это позволяет говорить о вполне определенной методике работы**. В качестве недостатка модели с точки зрения управления процессом разработки программ следует указать на то, что она, как и любая раскручивающаяся спираль, плохо отображает временные соотношения между сроками выполнения*****работ* на разных витках. Когда нужно отслеживать такие соотношения, приходится прибегать к дополнительным средствам.**

#### **6.4.5 Сравнительный анализ моделей жизненного цикла**

Очень хорошо в <https://habrahabr.ru/company/edison/blog/269789/> !!!

<http://www.studfiles.ru/preview/4533054/>

#### **6.4.6 Модели процесса разработки программного продукта. Модель CMM**

Интересный вопрос – отличие моделей ЖЦ ПО от моделей процесса разработки ПО!

<http://software-testing.ru/library/around-testing/engineering/267-swebok>

<http://www.studfiles.ru/preview/4533054/>

1. **Итерация** (лат. iteratio — повторяю) — повторение какого-либо действия.  [↑](#footnote-ref-1)