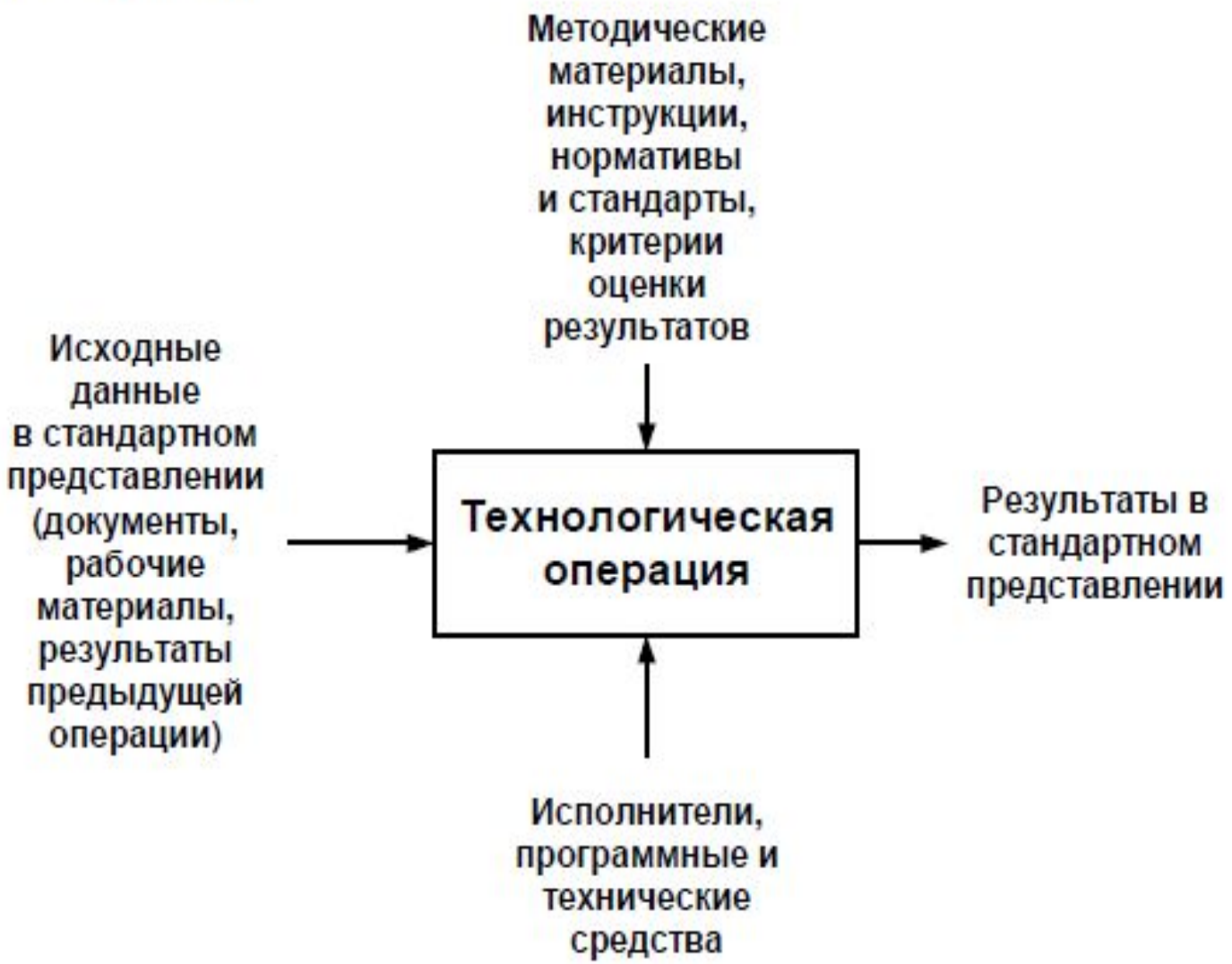
Сопровождение программных средств на основе технологии RUP

- *Технология создания ПО* (ТС ПО) – это упорядоченная совокупность взаимосвязанных технологических процессов в рамках ЖЦ ПО.
- *Технологический процесс* – это совокупность взаимосвязанных технологических операций.

Технологическая операция – это основная единица работы, выполняемая определенной ролью, которая:

- • подразумевает четко определенную ответственность роли;
- • дает четко определенный результат (набор рабочих продуктов), базирующийся на определенном наборе входных рабочих продуктов;
- • представляет собой единицу работы с жестко определенными границами, которые устанавливаются при планировании проекта.

- *Рабочий продукт* – информационная или материальная сущность, которая создается, модифицируется или используется в некоторой технологической операции (модель, документ, код, тест и т.п.).
- *Роль* – определение поведения и обязанностей отдельного лица или группы лиц в среде организации-разработчика ПО, осуществляющих деятельность в рамках некоторого технологического процесса и ответственных за определенные рабочие продукты.
- *Руководство* – практическое руководство по выполнению одной или совокупности технологических операций. Руководства включают методические материалы, инструкции, нормативы, стандарты и критерии оценки качества рабочих продуктов.
- *Инструментальное средство* (CASE-средство) – программное средство, обеспечивающее автоматизированную поддержку деятельности, выполняемой в рамках технологических операций.


Методические материалы, инструкции, нормативы и стандарты, критерии оценки результатов
Исходные данные в стандартном представлении (документы, рабочие материалы, результаты предыдущей операции)
Технологическая операция
Результаты в стандартном представлении
Исполнители, программные и технические средства

Набор процессов ЖЦ

- управление требованиями;
- анализ и проектирование ПО;
- разработка ПО;
- эксплуатация;
- сопровождение;
- документирование;
- управление конфигурацией и изменениями;
- тестирование;
- управление проектом.

Также есть ряд других требований:

- адаптируемость технологии к условиям применения;
- поддержка поставщика;
- простота использования;
- удовлетворительные стоимостные характеристики.

Основные принципы RUP :

- *Раннее определение рисков и выработка ответных мер.* В начале каждой стадии ЖЦ составляется список рисков (примеры: неосвоенная СП, интеграция с унаследованным кодом, орг. проблемы). По каждому риску из списка составляется перечень ответных мер. RUP учитывает изменчивость рисков – на разных этапах проекта списки рисков разные.
- *Выполнение требований заказчиков.* Требования описываются вариантами использования. Варианты использования – это отправная точка создания для модели анализа и проектной модели. Планирование и управление проектом ведется на основе вариантов использования. Варианты использования являются «сырьем» для тестов и пользовательской документации.

- *Концентрация на работающем коде.* Прогресс проекта оценивается по тому, какая часть системы готова и работает. Практика – лучший критерий проверки правильности того или иного решения. Все рабочие продукты проекта за исключением работающего кода являются вспомогательными, поэтому не следует слепо их создавать лишь из-за того, что они указаны в руководствах по RUP.
- *Готовность к изменениям с самого начала проекта и управление ими.* Система слишком сложна, чтобы с самого начала получить верное и окончательное проектное решение. Изменения позволяют улучшать принятые решения. Итеративный процесс позволяет исправлять дефекты, допущенные на ранних итерациях. Стоимость изменений в ходе проекта увеличивается. Управление изменениями позволяет уложиться в бюджет и сроки.

- *Сборка системы из компонентов.* Компонентная архитектура позволяет воспользоваться преимуществами инкапсуляции: повышает модифицируемость системы и возможности повторного использования ее частей. Стоимость разработки системы может быть снижена за счет использования компонентных платформ (J2EE, NET).
- *Визуальное моделирование.* Графические модели более наглядны и удобны, чем тексты на естественных и формальных языках. UML – стандартный язык, так что модели понятны большой аудитории (состоящей не только из людей, но и из CASE-средств). По той же причине имеется больше возможностей для повторного использования моделей.
- *Обеспечение качества в течение всего ЖЦ.* Используется упреждающее тестирование, лозунг которого: «Тесты раньше программ!» Регрессионное тестирование и первоочередная реализация приоритетных вариантов использования обеспечивают надежность ключевой функциональности системы. Качество создается в течение всего жизненного цикла за счет того, что все рабочие продукты критически оцениваются при рецензировании.

Динамический аспект:

- начальная стадия (inception);
- стадия разработки (elaboration);
- стадия конструирования (construction);
- стадия ввода в действие (transition).

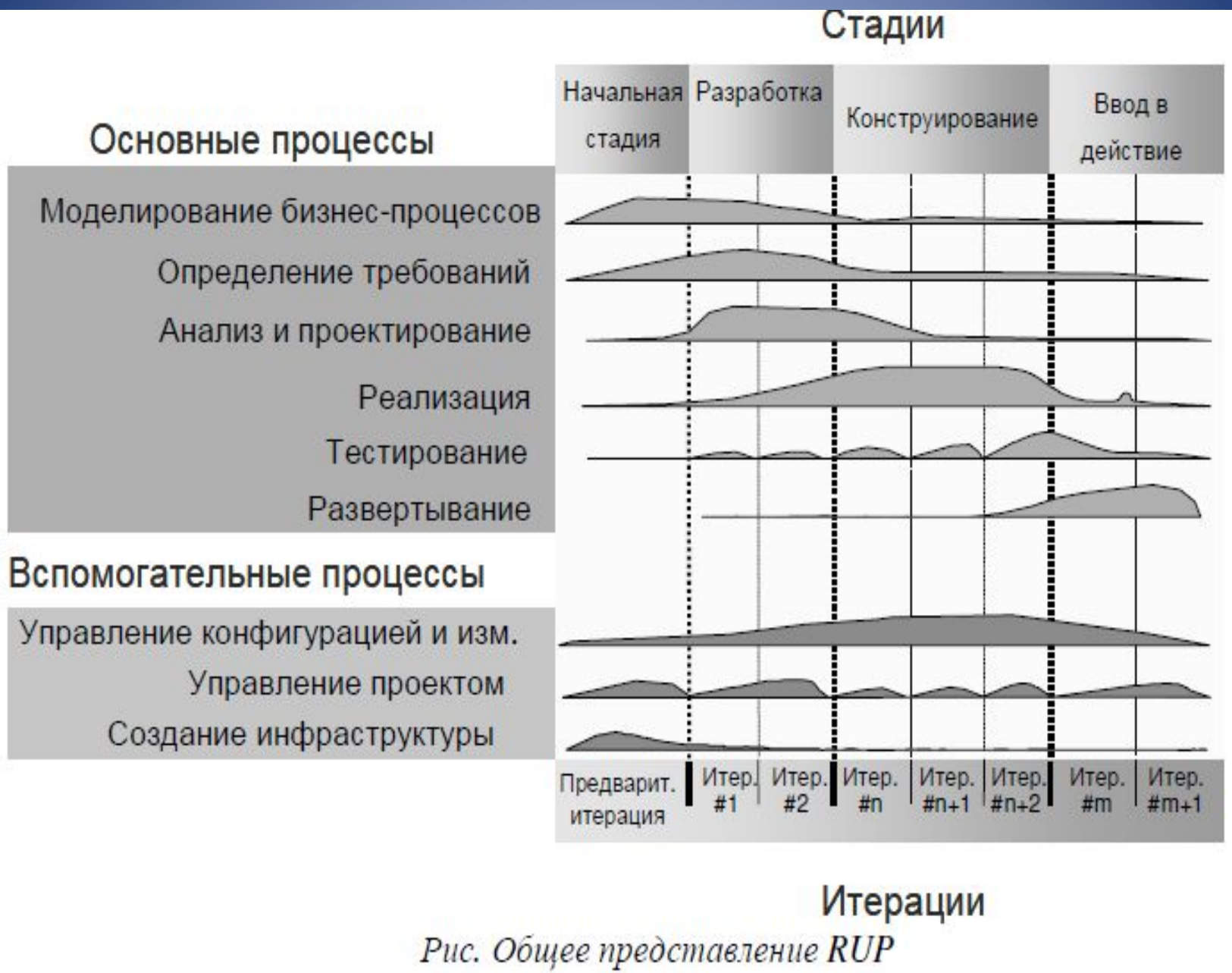


Рис. Общее представление RUP

Результатами стадии разработки являются:

- модель вариантов использования (готовая на 80%);
- перечень дополнительных требований, включая требования нефункционального характера и требования, не связанные с конкретными вариантами использования;
- описание базовой архитектуры будущей системы;
- работающий прототип;
- уточненный бизнес-план;
- план разработки проекта, отражающий итерации и критерии оценки для итераций.

Результатом стадии конструирования является начальная эксплуатационная версия программного продукта, готовая к передаче конечным пользователям. В её составе содержится как минимум следующее:

- ПО, интегрированное на требуемых платформах;
- Руководства пользователя;
- Описание текущей реализации.

Назначением стадии ввода в действие является доводка начальной эксплуатационной версии и передача готового продукта в распоряжение пользователей. Данная стадия включает:

- бета-тестирование, позволяющее убедиться, что новая система соответствует ожиданиям пользователей;
- параллельное функционирование с существующей системой, которая подлежит постепенной замене;
- конвертирование баз данных;
- оптимизацию производительности;
- обучение пользователей и специалистов службы сопровождения.

Статический аспект.

- Статический аспект RUP представлен четырьмя основными элементами:
- • роли;
- • виды работ;
- • рабочие продукты;
- • дисциплины (процессы).

Понятие «роль» (role) определяет поведение и ответственность личности или группы личностей, составляющих проектную команду. Одна личность может играть в проекте много различных ролей.

Вид работы – технологическая операция, выполняемая конкретным исполнителем.

Дисциплина (discipline) соответствует понятию технологического процесса (или процесса ЖЦ) и представляет собой последовательность работ, приводящую к получению значимого результата.

В рамках RUP определены шесть основных дисциплин:

- построение бизнес-моделей;
- определение требований;
- анализ и проектирование;
- реализация;
- тестирование;
- развертывание;

Три вспомогательных:

- управление конфигурацией и изменениями;
- управление проектом;
- создание инфраструктуры.

Задачи построения бизнес-моделей

– понять предметную область или бизнес-контекст, в которых должна будет работать система, и убедиться, что все заинтересованные лица понимают его одинаково, осознать имеющиеся проблемы, оценить их возможные решения и их последствия для бизнеса организации, в которой будет работать система.

В рамках этой дисциплины создаются следующие рабочие продукты:

- видение бизнеса;
- глоссарий деятельности предприятия;
- модель бизнес-процессов (описывающая контекст бизнеса и бизнес-процессы предприятия на диаграмме вариантов использования);
- модель бизнес-анализа (описывающая протекание бизнес-процессов, т. е. их реализацию, на диаграммах взаимодействия, участниками которых являются исполнители и бизнес-сущности, а также на диаграммах деятельности);
- описания бизнес-целей, бизнес-правил и бизнес-событий;
- дополнительная спецификация бизнеса.

В построении бизнес-моделей задействованы следующие роли:

- бизнес-аналитик, возглавляющий и координирующий работу по деловому моделированию, отвечающий за модель бизнес-процессов;
- бизнес-проектировщик, моделирующий отдельные бизнес-процессы под руководством бизнес-аналитика (результатом совместной работы бизнес-проектировщиков является модель бизнес-анализа).

Полученные модели предприятия служат основой для моделей требований и анализа.

Задачи определения требований

– понять, что должна делать система, и убедиться во взаимопонимании по этому поводу между заинтересованными лицами, определить границы системы и основу для планирования проекта и оценок затрат ресурсов в нем.

Требования принято фиксировать в виде модели вариантов использования и различных документов:

описаний вариантов использования, концепции системы, глоссария системы, дополнительной спецификации, отражающей нефункциональные требования.

Роли задействованные при определении требований:

- системный аналитик,
- архитектор,
- писатель вариантов использования,
- рецензент по требованиям,
- разработчик пользовательского интерфейса.

Задачи анализа и проектирования

– выработать архитектуру системы на основе требований, убедиться, что данная архитектура может быть основой работающей системы в контексте ее будущего использования. В результате должна появиться модель проектирования, включающая в себя диаграммы классов системы, диаграммы ее пакетов (подсистем), диаграммы взаимодействия между объектами в ходе реализации вариантов использования, диаграммы состояний для отдельных объектов и диаграммы деятельности, описывающие методы реализации операций некоторых классов, а также модель (диаграмму) развертывания и документ – описание архитектуры.

Основными рабочими продуктами анализа и проектирования являются:

модель анализа, прототипы пользовательского интерфейса, проектная модель, описание архитектуры, уточнённые описания вариантов использования, начальная версия модели развёртывания.

В анализе и проектировании задействованы следующие роли:

- архитектор,
- разработчик,
- разработчик БД,
- разработчик систем реального времени,
- разработчик пользовательского интерфейса,
- технический рецензент.

Дисциплина реализации решает следующие задачи

– определить структуру исходного кода системы, разработать код ее компонентов и протестировать их, интегрировать систему в работающее целое.

Включает в себя:

- Реализацию архитектуры (переход от проектной модели к модели реализации, представленной в виде диаграмм компонент и диаграмм пакетов).
- Выработку плана сборки для каждой итерации.
- Распределение компонентов системы по узлам вычислительной среды.
- Реализацию кода классов и подсистем.
- Покомпонентное тестирование.

Реализацию архитектуры осуществляет архитектор.

Заключается она в трассировке проектных классов, пакетов и подсистем в компоненты и установлении связей (зависимостей) между компонентами.

План сборки описывает функциональность, которая должна быть реализована в билде (сборке) и те компоненты, которые входят в билд. Планы составляет системный интегратор.

За реализацию кода отвечает кодировщик. Покомпонентное тестирование – это раздельное тестирование компонент системы. Осуществляет его кодировщик путем тестирования на основе спецификации («черный ящик») и/или тестирования на основе структуры («белый ящик»).

Основные рабочие продукты процесса реализации: модель реализации, сборка, уточненное описание архитектуры, записи ревью, тесты и их результаты.

Задействованные роли:

- разработчик,
- кодировщик,
- системный интегратор,
- архитектор,
- технический рецензент.

Дисциплина тестирования решает следующие задач

- найти и описать дефекты системы (проявления недостатков ее качества),
- оценить ее качество в целом,
- оценить выполнены или нет гипотезы, лежащие в основе проектирования,
- оценить степень соответствия системы требованиям.

Включает в себя:

- Планирование тестов на каждой итерации.
- Составление тестовых вариантов (test-case) и тестовых сценариев (test scripts).
- Тестирование с целью обнаружения дефектов.

Тестовый вариант включает

- входные данные,
- условия выполнения отдельных шагов,
- корректные ответы системы для всякого шага, на котором ответ системы можно наблюдать.

Тестовый сценарий

способ выполнения одного или нескольких тестовых наборов в рамках тестового варианта. Выполняется вручную или автоматически.

Виды тестирования:

- регрессионное (при котором новые сборки проверяются на тестах для предыдущих сборок, поскольку при интеграции новых компонент может быть нарушено правильное функционирование старых и системы в целом);
- инсталляционное (проверка возможности установки системы на вычислительной среде и правильность работы инсталлятора);
- конфигурационное (проверка работы системы в разных конфигурациях);
- отрицательное (проверка устойчивости системы на заведомо неверных данных, при неверных действиях пользователя – «защита от дурака», при недостаточных ресурсах);
- нагрузочное (проверка нефункциональных свойств, например, производительности, пропускной способности).