Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Институт повышения квалификации
и переподготовки

Кафедра «Профессиональная переподготовка»

А. Н. Осипенко

МЕНЕДЖМЕНТ В ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЯХ

ПОСОБИЕ
по одноименному курсу
для слушателей специальности 1-40 01 73
«Программное обеспечение информационных систем»
заочной формы обучения

Гомель 2017

УДК 681.3.06(075.8)
ББК 32.973.202я73
О-74

Рекомендовано кафедрой «Профессиональная переподготовка»
Института повышения квалификации
и переподготовки ГГТУ им. П. О. Сухого
(*протокол № 5 от 20.09.2016 г.*)

Рецензент: зав. каф. «Информатика» ГГТУ им. П. О. Сухого
канд. физ.-мат. наук, доц. *Т. В. Тихоненко*

Осипенко, А. Н.

О-74 Менеджмент в информационных технологиях : пособие по одноим. курсу для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем» заоч. формы обучения / А. Н. Осипенко. – Гомель : ГГТУ им. П. О. Сухого, 2017. – 84 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://elib.gstu.by. – Загл. с титул. экрана.

Пособие содержит основные теоретические сведения, необходимые для понимания принципов менеджмента в информационных технологиях, а также для организации эффективной работы менеджера проекта информационной системы.

Издание адресовано слушателям ИПКиП специальности 1-40 01 73 «Программное обеспечение информационных систем».

УДК 681.3.06(075.8)
ББК 32.973.202я73

Оглавление

Предисловие 4
1 Менеджмент в разработке программных изделий 5
1.1 Общие положения о менеджменте в информационных технологиях 5
1.2 Функциональные роли в коллективе разработчиков 8
2 ИТ-сервис – основа деятельности современной ИС службы 18
2.1 Понятие ИТ-сервиса 18
2.2 Функциональные области управления службой ИС 23
3 ITIL/ITSM – концептуальная основа процессов ИС-службы 33
3.1 Общие сведения о библиотеке ITIL 33
3.2 Процессы поддержки ИТ-сервисов 37
2.3 Процессы предоставления ИТ-сервисов 48
4 Повышение эффективности ИТ-инфраструктуры предприятия на примере методологии Microsoft 58
4.1 Библиотека документов MOF 58
4.2 Модель процессов MOF 59
5 Основные методологии разработки ПО 64
5.1 Модель водопада 64
5.2 Итеративная разработка 67
5.3 Методология RUP 68
5.4 Гибкие методологии 73
Заключение 83
Список использованных источников 84

Предисловие

Менеджмент в информационных технологиях является одним из важнейших направлений деятельности в современной информационной сфере. Стремительный рост числа пользователей интернета и открытие большого числа сайтов привело к тому, что создание и поддержка веб-сайтов со все более развитым функционалом стало востребованным направлением IT-технологий. Вместе с тем, в данный момент создание программного обеспечения происходит по далеким от совершенства методикам. В частности, почти полностью отсутствует качественный менеджмент IT-проектов.

Настоящее пособие разработано в соответствие с программой дисциплины «Менеджмент в информационных технологиях» для обучения слушателей ИПК и П заочной формы обучения по специальности 1-40 01 73 «Программное обеспечение информационных систем».

Целью методического пособия является повышение профессионального уровня будущих специалистов по созданию программного обеспечения информационных систем в области менеджмента IT-проектов. Задачей методического пособия является формирование у слушателей современных представлений о разработке IT-проектов, привлечении кадров, распределении времени, работе с заказчиками, формированию требований к проекту, а также решении проблем эффективного использования современных инструментов разработки программного обеспечения на основе накопленного к настоящему времени мирового опыта.

1 Менеджмент в разработке программных изделий

1.1 Общие положения о менеджменте в информационных технологиях

В общем виде проект (англ. project) – это «что-либо», что задумывается или планируется, например, программное изделие.

С точки зрения системного подхода, проект может рассматриваться как процесс перехода из исходного состояния в конечное – результат при участии ряда ограничений и механизмов.

В «Кодексе знаний об управлении проектами» проект – некоторая задача с определенными исходными данными и требуемыми результатами, обуславливающими способ ее решения. Проект включает в себя замысел, средства его реализации и получаемые в процессе реализации результаты [1].

Методы управления проектами позволяют: определить цели проекта и пронести его обоснование; выявить структуру проекта; определить необходимые объемы и источники финансирования; подобрать исполнителей через процедуры торгов и конкурсов; подготовить и заключить контракты; определить сроки выполнения проекта, составить график его реализации, рассчитать необходимые ресурсы; рассчитать смету и бюджет проекта; планировать и учитывать риски; обеспечить контроль за ходом выполнения проекта.

Разработка программного обеспечения в большинстве случаев должна рассматриваться как коллективный труд специалистов, направленный на удовлетворение потребности пользователей в автоматизации их деятельности. Как и любой другой коллективный труд, она требует организации, в частности – *управления*. Это процесс, порою длительный, связывающий производственными и иными отношениями тех, кого в той или иной степени можно рассматривать в качестве производителей программы. Как и любой труд, тесно связанный с неоднозначными потребностями тех, кто будет использовать продукты труда, необходимым элементом *разработки программ* является решение задач изучения пользователей, с одной стороны, а с другой – обеспечения обратной связи с ними, направляющей производство. Это составляющие, из которых формируются главные задачи управления производством

программ. Чаще всего решение таких задач осуществляется руководителем, или, как принято говорить, ***менеджером проекта***.

Понятие «*менеджер проекта*» не обязательно соотносится с конкретной персоной, отвечающей за управление производством программной системы в целом. В небольшом проекте такое единоначалие чаще всего оправданно: оно позволяет концентрировать все нити управления, исключает проблемы внутреннего для проекта согласования противоречий, обеспечивает централизованную ответственность за проект перед теми, кто заинтересован в его выполнении. Однако по мере перехода к более масштабным проектам менеджерские обязанности становится невозможно концентрировать в одних руках. Обычно в таких случаях образуют ***службу менеджера***, состоящую из его помощников, которым он может поручать различные задания, освобождая себя от рутины постоянного контроля. *Делегирование* можно считать основным инструментом разделения труда в проекте (не только в случае *менеджмента*), когда есть ответственность за некоторую функцию (работу и др.), но для ее выполнения нет собственных ресурсов, а потому приходится прибегать к помощи.

Для небольших групп возможна следующая организация работ [2]: обязанности главного менеджера распределяются по команде разработчиков, которая за счет внутренних механизмов решает, как планировать работы, как их распределять и контролировать. Это характерно для так называемого подхода быстрой *разработки (agile development) программных систем*, объединившего в себе несколько методологий программирования, которые отказываются от многих принципов традиционного *менеджмента*. Наиболее ярким примером подхода быстрой разработки является экстремальное программирование. Отметим, что здесь, как и в других случаях, менеджерские задачи не исчезают, как могло бы показаться на первый взгляд. Просто форма и методы их решения приобретают другой характер.

В работе менеджера всегда присутствуют и неразрывно связаны друг с другом два аспекта:

- *управление проектом* как деятельность, продвигающая процесс производства к определенным целям,
- *руководство людьми*, *участниками разработки*.

С организационной точки зрения в *разработке программного обеспечения* можно выделить три варианта целей, определяющие деятельность *менеджера*:

- *производство программ не для продажи*, напрямую не связанное с получением дохода.
- *производство рыночного продукта*, обеспечивающего прибыль за счет распространения (продажи) получаемых результатов.
- *разработка ведется под заказ*, когда все производство программы, от стадии замысла до передачи в эксплуатацию, финансируется внешними по отношению к разработчикам, но весьма заинтересованными агентами, обычно называемыми заказчиками.

Главная и постоянная задача *менеджмента разработки программного обеспечения* – продвижение проекта к обозначенным в начале его развития результатам. Если оставить в стороне приемы и методы, с помощью которых достигается решение этой задачи, то она сводится к распределению и контролю эффективного использования имеющихся ресурсов проекта: времени, финансов, технических средств и производственного потенциала работников. Множественность критериев, необходимость согласования интересов участников проекта и его заказчиков, разнообразие видов деятельности, составляющих развитие проекта, – вот тот организационный и производственный контекст, который вынужден учитывать менеджер.

Характерной особенностью *разработки программного обеспечения* является стремление к *переиспользованию* ранее созданных программных компонентов [3]. Задачи *переиспользования* – это:

- во-первых, определение программных продуктов или каких-либо иных изделий и инструментов, имеющихся в распоряжении разработчиков, использование которых могло бы способствовать прогрессу развития проекта,
- во-вторых, выявление компонентов данного проекта, которые было бы полезно задействовать в других разработках.

Второе дополнительное направление – это задача *распространения построенного программного продукта*, в частности, продвижение сайта. Если с самого начала не рассматривать ее и относить распространение лишь к этапам, следующим за разработкой, есть опасность сделать не то, что нужно потребителю продукции, и

выпустить из рук рычаги, с помощью которых можно влиять на сферу возможного применения создаваемых программных продуктов.

С точки зрения управления ***участники проекта*** – это абстрактные действующие агенты, которые выполняют заданные функции. Здесь под функцией понимается некое действие, при выполнении которого потребляются определенные ресурсы и производится определенный результат. Функциональный взгляд на *участников разработки проекта* делает их взаимозаменяемыми, обезличенными в пределах компетенции, соответствующей выполнимости функции. Он приводит к понятию роли, назначаемой работнику для выполнения соответствующих обязанностей.

Ключевым качеством коллектива, определяющим его успешность, является слаженность [4]. В идеальном коллективе все понимают друг друга с полуслова, есть взаимопонимание и уважение, не происходят или сведены к минимуму внутренние конфликты и противоречия. В реальности менеджеру редко приходится иметь дело с таким коллективом, а значит, необходимы меры, способствующие не только росту индивидуальной квалификации работников, но и дееспособности формируемой команды в целом. Эти меры следует рассматривать в качестве задач менеджера как руководителя коллектива.

Таким образом, при обсуждении задач менеджера в разработке программных изделий мы сталкиваемся с тремя взаимосвязанными *направлениями его деятельности:*

- первое направление – это те *функции*, которые необходимо выполнять для успешного развития проекта. Здесь следует выяснить, какие роли сотрудников требуются для данного проекта;
- второе направление – планирование и контроль хода проекта в соответствии с *жизненным циклом создаваемого программного обеспечения;*
- наконец, третье направление определяется кругом задач формирования коллектива и, в частности, кадровым обеспечением проекта.

1.2 Функциональные роли в коллективе разработчиков

Для целенаправленного выполнения проекта должен быть выполнен ряд работ, различных как по своему назначению, так и по квалификационным требованиям, предъявляемым к *разработчикам*.

Иными словами, в ходе развития проекта командой *разработчиков* выполняются те или иные функции.

Функции, выполняемые *разработчиками*, – понятие неформализованное. В разных проектах оно может обретать свое содержание. Тем не менее, *типовые функции*, которые предполагают практически все программные проекты, можно перечислить. Так, в любом программном проекте есть функции кодирования, т.е. записи программы на алгоритмическом языке по имеющимся спецификациям, анализа требований, т.е. выявления истинной потребности в создаваемой программе, тестирования и отладки.

Таким образом, в рамках любого проекта возникает задача повышения квалификации сотрудников. Для различных схем ведения проектов эта задача решается по-разному. Крайнюю точку зрения на проблему соответствия квалификации работников требованиям проекта отражает идея ***экстремального программирования***, когда используется неформальная организация группы исполнителей проекта без четкого распределения ролей, а значит, и обязательств сотрудников. В этом случае провозглашается принцип, когда каждый в группе делает то, что он умеет делать лучше всего. И хотя все функции, которые должны выполняться, остаются, создается впечатление, что в группе исполнителей проекта исчезают роли. В результате возможны пробелы в разработке, в частности при анализе и декомпозиции проектируемой системы. Чтобы этого избежать, *разработчики* должны понимать, какую абстрактную роль они исполняют в каждый конкретный момент, выполнение каких проектных функций необходимо сейчас, как связаны между собой работы, как должны распределяться ресурсы. Словом, они должны обращать внимание на выполнение распределенных по группе менеджерских функций. И даже в этом случае схему *экстремального программирования* можно рекомендовать лишь для слаженных групп исполнителей с высоким уровнем коллективной ответственности.

В *модели проектной группы концепции Microsoft Solution Framework (MSF)* предлагается образовывать небольшие мобильные коллективы как атомарные производственные единицы с общей ответственностью за выполняемые задания – так называемые проектные группы [5]. Такие группы строятся как многопрофильные команды, члены которых распределяют между собой ответственность и дополняют *области компетентности* друг друга. Группа состоит не более чем из 10 человек. Все они считаются обладающими

сходным уровнем профессиональной подготовки, хотя и в разных областях индивидуальной специализации. Провозглашается равноправие членов группы и коллективная ответственность за выполняемые задания: проектная группа – команда равных. Все это позволяет сохранять внутри группы неформальные отношения.

Вместо понятия роли для группы в целом определяются ***ролевые кластеры***, которые заполняются точно так же, как происходит распределение ролей. В то время как за успех проекта ответственна вся команда, каждый из ее *ролевых кластеров*, определяемых моделью, ассоциирован с одной из проектных целей и работает над ее достижением. В данной модели именно эти цели задают роли *разработчиков*, которые определяются кластерами. В терминологии MSF используется понятие *области компетенции*, или *области функциональной специализации* (functional area), обозначающее ту или иную роль, которую выполняет кластер группы в проекте. Принципиальное отличие распределения исполнителей по *ролевым кластерам* от распределения ролей заключается лишь в том, что ответственность за это несет не *менеджер проекта*, а сама группа. Менеджер проекта выдает задания и контролирует их выполнение лишь в целом для группы, не вмешиваясь в ее работу.

Определено шесть *ролевых кластеров*, которые соответствующим образом структурируют проектные функции *разработчиков* (рисунок 1.1).

Рисунок 1.1 – Ролевые кластеры модели проектной групп MSF

Управление продуктом (product management). Ключевая цель кластера – обеспечивать удовлетворение интересов *заказчика*. Для ее достижения кластер должен содержать следующие области компетенции:

- планирование продукта,

- планирование доходов,
- представление интересов *заказчика*,
- маркетинг.

Управление программой (program management). Задача – обеспечить реализацию решения в рамках ограничений проекта, что может рассматриваться как удовлетворение требований к бюджету проекта и к его результату. *Области компетенции* кластера:

- управление проектом;
- выработка архитектуры решения;
- контроль производственного процесса;
- административные службы.

Разработка (development). Первостепенной задачей кластера является построение решения в соответствии со спецификацией. *Области компетенции* кластера:

- технологическое консультирование;
- проектирование и осуществление реализации;
- разработка приложений;
- разработка инфраструктуры.

Тестирование (test). Задача кластера – одобрение выпуска продукта только после того, как все дефекты выявлены и устранены. *Области компетенции* кластера:

- разработка тестов;
- отчетность о тестах;
- планирование тестов.

Удовлетворение потребителя (user experience). Цель кластера – повышение эффективности использования продукта. *Области компетенции* кластера: – общедоступность (возможности работы для людей с недостатками зрения, слуха и др.);

- интернационализация (эксплуатация в иноязычных средах);
- обеспечение технической поддержки;
- обучение пользователей;
- удобство эксплуатации (эргономика);
- графический дизайн.

Управление выпуском (release management). Задача кластера – беспрепятственное внедрение и сопровождение продукта. *Области компетенции* кластера:

- инфраструктура (infrastructure);
- сопровождение (support);
- бизнес-процессы (operations);

- управление выпуском готового продукта (commercial release management).

Мы придерживаемся ролевой структуры проекта, которая предложена Центром объектно-ориентированной технологии компании IBM. Эта структура включает достаточно полный перечень типичных ролей, согласованный со многими реальными дисциплинами развития программных проектов. В то же время она представляет роли *разработчиков* в организационном контексте, т.е. рассматривает не только *разработчиков*, но и тех, кто, не участвуя в проекте в качестве исполнителей, оказывает влияние на постановку задач проекта, на выделение ресурсов и обеспечение осуществимости развития работ:

- *заказчик* (Customer) – реально существующий (в организации, которой подчинена команда, или вне ее) инициатор разработки или кто-либо иной, уполномоченный принимать результаты (как текущие, так и окончательные) разработки;
- *планировщик ресурсов* (Planner) – выдвигает и координирует требования к проектам в организации, осуществляющей данную разработку, а также развивает и направляет план выполнения проекта с точки зрения организации;
- *менеджер проекта* (Project Manager) – отвечает за развитие проекта в целом, гарантирует, что распределение заданий и ресурсов позволяет выполнить проект, что работы и предъявление результатов идут по графику, что результаты соответствуют требованиям. В рамках этих функций *менеджер проекта* взаимодействует с *заказчиком* и *планировщиком ресурсов;*
- *руководитель команды* (Team Leader) – производит техническое руководство командой в процессе выполнения проекта. Для больших проектов возможно привлечение нескольких руководителей подкоманд, отвечающих за решение частных задач;
- *архитектор* (Architect) – отвечает за проектирование архитектуры системы, согласовывает развитие работ, связанных с проектом;
- *проектировщик подсистемы* (Designer) – отвечает за проектирование подсистемы или категории классов, определяет реализацию и интерфейсы с другими подсистемами;
- *эксперт предметной области* (Domain Expert) – отвечает за изучение сферы приложения, поддерживает направленность проекта на решение задач данной области;

- *разработчик* (Developer) – реализует проектируемые компоненты, владеет и создает специфичные классы и методы, осуществляет кодирование и автономное тестирование, строит продукт. Это широкое понятие, которое может подразделяться на специальные роли (например, *разработчик* классов). В зависимости от сложности проекта команда может включать различное число *разработчиков;*
- *разработчик информационной поддержки* (Information Developer) – создает документацию, сопровождающую продукт, когда выпускается версия. Включаемые в нее инсталляционные материалы, равно как ссылочные и учебные, а также материалы помощи предоставляются на бумажных и машинных носителях. Для сложных проектов возможно распределение этих задач между несколькими *разработчиками информационной поддержки;*
- *специалист по пользовательскому интерфейсу* (Human Factors Engineer) – отвечает за удобство применения системы. Работает с *заказчиком*, чтобы удостовериться, что пользовательский интерфейс удовлетворяет требованиям;
- *тестировщик* (Tester) – проверяет функциональность, качество и эффективность продукта. Строит и исполняет тесты для каждой фазы развития проекта;
- *библиотекарь* (Librarian) – отвечает за создание и ведение общей библиотеки проекта, которая содержит все проектные рабочие продукты, а также за соответствие рабочих продуктов стандартам.

Взаимодействие менеджера с *заказчиком*, *планировщиком* и другими *инициаторами работ* – прямая обязанность *менеджера проекта*. Таким образом, для него закреплен круг функций, которые являются внешними по отношению к работам над проектом.

Выполнение *внешних функций* создает лишь условия для разработки. Как видно из перечня ролей, другой круг функций, возлагаемый на менеджера, связан с взаимодействиями с действующими лицами из команды *разработчиков* проекта. Это *внутренние функции менеджера*.

В зависимости от проекта и условий его выполнения роли участников проекта могут совмещаться. Предельный случай – программист разрабатывает проект для себя (по собственному заказу), сам планирует распределение ресурсов (сроки выполнения работы, использование вычислительной техники и др.), сам

принимает проектные решения (управляет и руководит собою) и сам же занимается разработкой, экспертизой и обслуживанием.

В таблице 1.1 приводятся краткие характеристики *совмещения ролей*, которые можно рассматривать как рекомендации для менеджмента

Таблица 1.1 – Совместимость ролей

Роли	**Характеристика совмещения ролей**
Менеджер и архитектор	Желательно
Менеджер и руководитель команды	Противоречиво
Руководитель команды и архитектор	Возможно
Руководитель команды и проектировщик подсистемы	Нежелательно
Менеджер и разработчик	Не допускается
Для различных разработчиков	с ограничениями
Создание документации (все сотрудники)	Успешно распределяется
Специалист по интерфейсу и менеджер	Разумно
Эксперт предметной области и менеджер	Зачастую разумно
Специалист по интерфейсу и эксперт предметной области	Редко бывает эффективно
Эксперт предметной области и разработчик	Бывает полезно
Специалист по интерфейсу и разработчик	Часто полезно
Библиотекарь и один из разработчиков	Допустимо
Тестировщики и другие члены команды	Перекрестно
Эксперт предметной области, тестировщик	Оправданно

. Не стоит жестко фиксировать распределение ролей и в одном проекте. Более того, распределение ролей можно рассматривать в качестве одного из инструментов, с помощью которого менеджер может руководить коллективом.

В большинстве случаев *заказчик* и *планировщик ресурсов* являются действительно внешними по отношению к проекту действующими лицами, а потому *совмещение этих ролей* с другими – нечто экзотическое. Тем не менее, роль *заказчика* как члена коллектива *разработчиков*, аккумулирующего точки зрения всех *инициаторов работ*, весьма полезна. В частности, подход *экстремального программирования* считает это обязательным, чтобы развитие проекта всегда гарантированно было направлено в сторону, нужную пользователям.

Менеджер проекта по своему назначению является выделенным в команде. Он берет на себя взаимодействие с *заказчиком* и *планировщиком ресурсов*, с одной стороны, а с другой – распределяет работы среди членов команды. Последнее означает, что он должен обладать полной информацией о декомпозиции проекта. Как следствие, совмещение его роли с ролью *архитектора* проекта является весьма желательным, а потому, довольно частым. Единственным условием для такого совмещения является требование четко знать, когда и чьи функции выполняет данное действующее лицо.

Вполне возможно продуктивное *совмещение ролей руководителя команды* и *архитектора* – это дает ощутимые результаты, когда команда достаточно крепко связана со своим руководителем, например, по предшествующим работам, но при условии не слишком высокой сложности задач декомпозиции для данного проекта.

Совмещение ролей *руководителя команды* и менеджера допустимо, но лишь тогда, когда осознается и учитывается противоречивость целевых установок этих ролей: *руководитель команды* действует в условиях, которые формируются менеджером.

Нежелательно *совмещение ролей руководителя команды* и *проектировщика* какой-либо подсистемы. И это обусловлено противоречивостью их ролевых интересов.

Если используется *перекрестное совмещение ролей руководителя команды* и *проектировщика подсистемы*, то возникает еще одно противоречие их ролевых интересов: у руководителя могут быть предпочтения в пользу «своего» компонента и, как следствие, возможен дисбаланс проекта в целом в пользу выделенных его составляющих, который придется компенсировать дополнительными усилиями менеджера.

По тем же причинам не допускается *совмещение ролей* менеджера и *разработчика*. Здесь запрет более жесткий, поскольку контрольные функции менеджера несовместимы с исполнительскими задачами *разработчика*. Даже в тех случаях, когда у менеджера остается свободное время после выполнения своих прямых обязанностей, ему не следует «помогать» *разработчикам*. Лучше занять себя другим делом, в частности выступить в роли *разработчика* другого проекта.

В то же время *совмещение ролей различных разработчиков* – обычное дело для больших проектов. Оно является частью распределения работ по исполнителям. Решают эту задачу совместно *руководитель команды* и менеджер, когда основные архитектурные положения утверждены. Способы решения зависят от того, как формируется команда. Если разработка поручается готовой команде, то возрастает роль руководителя, а менеджер лишь утверждает его предложение. Когда команда создается для проекта, растет удельный вес менеджерских функций в подборе кадров для работ. Но в любом случае при *совмещении ролей различных разработчиков* нужно учитывать уровень квалификации работников, их предпочтения и другие индивидуальные особенности. Необходимо знать, что, выделяя одному действующему лицу несколько работ, следует стремиться к однородности заданий и указывать их сравнительные приоритеты.

Допустимы и другие *совмещения ролей*. Так, довольно часто создание документации распределяется между всеми исполнителями проекта, а на *руководителя команды* и менеджера возлагается задача интеграции документов. Для обозримых проектов функции библиотекаря можно поручить одному из *разработчиков*, соответственно скорректировав его индивидуальное задание. Специалистом по пользовательскому интерфейсу вполне может быть, например, менеджер, поскольку именно он осуществляет контакты с *заказчиком* (в данной ситуации *заказчик* либо сам является пользователем, либо выступает как представитель пользователя программного изделия).

По поводу *тестировщиков* необходимы некоторые разъяснения. Следует различать два вида тестирования: тестирование модулей, автономное по своей сути, и тестирование предоставляемых функций, которое может быть и комплексным (проверка совместного выполнения функций), и автономным, когда проверяется

работоспособность отдельного аспекта системы. В первом случае задачи *тестировщика* невозможно выполнить без знания не только назначения, но и структуры проверяемого модуля. Лучше всего в этом разбирается *разработчик* модуля, а значит, именно ему этот модуль и отлаживать. Автономное тестирование второго вида, по существу, есть другая сторона проверки работоспособности модуля, реализующего определенную функцию. Во многих случаях даже нет нужды различать его и тестирование модуля. Поэтому и здесь разумно говорить о деятельности *разработчика*. В обоих случаях самотестирование для *разработчика* есть не дополнительная роль в проекте, а часть его профессиональных обязанностей в рамках автономной отладки.

Другой метод, нисколько не противоречащий априорному тестированию, – перекрестное тестирование, когда *разработчики* независимых компонентов проекта тестируют функциональность друг друга. Здесь можно говорить о *перекрестном совмещении ролей разработчиков* и *тестировщиков*, поскольку для выполнения перекрестного тестирования нужно различное представление об испытываемом модуле.

Вопросы для самоконтроля:

1. Назовите три варианта целей, определяющих деятельность менеджера.
2. Назовите основные задачи менеджера по разработке программных проектов.
3. Назовите основные направления деятельности менеджера по разработке программных проектов.
4. В чем заключается идея экстремального программирования?
5. Назовите ролевые кластеры, структурирующие проектные функции разработчиков.
6. Назовите участников ролевой структуры проекта, предложенной компанией IBM.
7. Какие роли в коллективе разработчиков целесообразно совмещать?
8. Какие роли в коллективе разработчиков совмещать крайне нежелательно?

2 ИТ-сервис – основа деятельности современной ИС службы

В данном разделе рассматриваются основные понятия ИТ-менеджмента, ИТ-сервиса, характеристики ИТ-сервиса, основы процессной модели управления ИС-службой в ее взаимосвязи с ИТ-сервисами, с одной стороны, и функциональной моделью с другой.

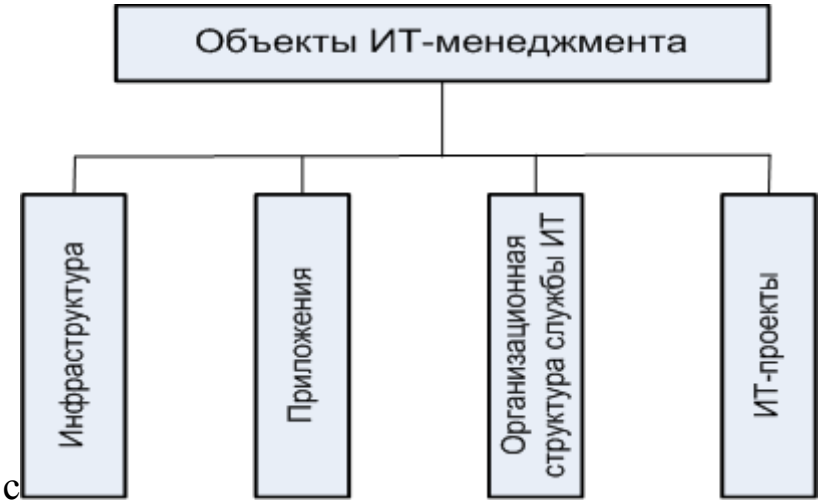
2.1 Понятие ИТ-сервиса

Системы управления информационными технологиями (ИТ) предприятий и организаций являются достаточно сложными, поскольку требуется учет интересов множества участников, вовлеченных в создание и использование ИТ-ресурсов (спонсоров создания информационной системы, конечных пользователей и разработчиков).

Понятие «информационные технологии» является общеупотребительным, в то же время отсутствует общепризнанное определение этого понятия. Мы будем придерживаться определения, данного в энциклопедии Wikipedia: Информационные технологии (ИТ), или информационные и коммуникационные технологии (ИКТ), – это технологии, применяемые для обработки информации. В частности, они используют компьютеры и программное обеспечение для преобразования, хранения, защиты, передачи и извлечения информации в любом месте и в любое время. С учетом этого определения ИТ-менеджмент охватывает управление всеми компьютерными и коммуникационными ресурсами предприятия [6]. Его основная задача состоит в создании и поддержании в работоспособном состоянии приложений и инфраструктуры, на которой они исполняются [7]. Подобный менеджмент можно разделить на три уровня: операционный, тактический и стратегический. На стратегическом уровне обеспечивается установление соответствия между информационными функциями системы и ее контентом, что сводится к атрибуции задач на поле информационной политики, определению содержания информационных функций и ИТ-поддержке. На операционном и тактическом уровнях ИТ-менеджмента должны обеспечиваться заданные уровни работоспособности и надежности эксплуатации

приложений информационной системы (ИС) на продолжении всего жизненного цикла системы.

Создание системы управления ИТ, как и любой другой системы управления, предполагает определение управляемых объектов и управляющих воздействий (рисунок 2.1).



с

Рисунок 2.1 – Объекты информационного менеджмента

Объектами ИТ-менеджмента являются:

- инфраструктура;
- приложения;
- организационная структура службы ИТ;
- ИТ-проекты.

Инфраструктура ИТ включает техническое и системное программное обеспечение. Техническое обеспечение ИТ состоит из серверов, персональных компьютеров, систем хранения данных, сети и коммуникационных приложений. Программное обеспечение характеризуется операционными системами, инструментальными средами разработки, программами поддержки ИТ-менеджмента и средствами обеспечения информационной безопасности.

Приложения обеспечивают поддержку бизнес-процессов предприятия и работоспособность отдельных автоматизированных рабочих мест.

Организационная структура службы ИТ определяет состав подразделений, распределение между ними функций и задач. Служба ИТ должна обеспечивать разработку, ввод в действие и эксплуатацию информационной системы посредством координированных действий,

которые обеспечивают непрерывность функционирования существующей системы в соответствии с согласованными правилами и процедурами на протяжении жизненного цикла ИТ.

ИТ-проекты представляют собой проекты внедрения новых информационных систем, а также модернизацию существующих. При этом модернизация (изменения, дополнения) рассматривается как результат действий, выполненных по запросу и относящихся к функциональным или нефункциональным требованиям, которые не были специфицированы изначально, при разработке и внедрении системы.

В настоящее время бизнес характеризуется высокой динамикой (слияния, поглощения, смена стратегических целей). Это обуславливает тот факт, что информационные системы предприятий находятся в условиях постоянных изменений, вызванных следующими факторами:

- перемены как внутри предприятий, так и в окружающей среде;
- развитие технологий, появление принципиально новых технических решений;
- появление новых информационных технологий;
- социальные изменения.

Кроме того, современное состояние бизнеса в отношении информационных технологий характеризуется достаточно жестким контролем инвестиций, выделяемых на ИТ, и возросшими требованиями к ИТ со стороны бизнеса. С учетом этого, на первый план выходят требования к информационным системам, которые определяют систему информационного менеджмента, способную видоизменять ИТ предприятия или организации синхронно с изменением бизнеса [8]. В соответствии с этими требованиями основная роль ИТ на предприятии определяется как информационное обслуживание её подразделений с целью повышения эффективности бизнеса. Информационное обслуживание бизнеса состоит в предоставлении информационных сервисов (ИТ-сервисов) заданного качества подразделениям предприятия.

ИТ-сервис в корпоративной среде – это ИТ-услуга, которую ИТ-подразделение (департамент, отдел, служба) или внешний провайдер предоставляет бизнес-подразделениям предприятия для поддержки их бизнес-процессов.

Примерами корпоративных ИТ-сервисов могут быть электронная почта, сетевая инфраструктура, системы хранения данных, бизнес-

приложения (начисление заработной платы, формирование счетов), бизнес-функции (списание/начисление денежных средств на счете клиента).

Набор ИТ-сервисов, необходимых организации, индивидуален и в значительной степени зависит от отрасли, размеров организации, уровня автоматизации, квалификации персонала, стратегии развития и т. п. Корпоративные ИТ-сервисы можно разбить на три большие группы:

- поддержка ИТ-инфраструктуры;
- поддержка бизнес-приложений;
- поддержка пользователей.

В общем случае ИТ-сервис характеризуется рядом параметров [9]:

- функциональность;
- время обслуживания;
- доступность;
- надежность;
- производительность;
- конфиденциальность;
- масштаб;
- затраты.

Функциональность определяет решаемую задачу (информатизацию бизнес-операции, бизнес-функции, бизнес-процесса) и предметную область её использования.

Время обслуживания определяет период времени, в течение которого ИТ-подразделение поддерживает данный сервис, то есть несет ответственность за его непрерывное функционирование. Время обслуживания измеряется долей суток и долей календарной недели, в течение которых ИТ-подразделение поддерживает ИТ-сервис. Например, время обслуживания 24x7 означает, что ИТ-сервис поддерживается 24 часа в сутки 7 дней в неделю, 8x5 - 5 дней в неделю по рабочим дням по 8 часов в день, то есть в течение рабочего дня.

Доступность определяет долю согласованного времени обслуживания, которая измеряется в процентах, и характеризует, в течение какого времени ИТ-сервис доступен.

Надежность определяется средним временем наработки на отказ ИТ-сервиса, то есть средним периодом времени между двумя сбоями в предоставлении ИТ-сервиса. Например, если в условиях

предыдущего примера (время обслуживания 8x5, доступность 95%) в неделю в среднем происходит два сбоя ИТ-сервиса, среднее время наработки на отказ составляет 19 часов.

Производительность характеризует способность информационной системы соответствовать требованиям своевременности. Для различных ИТ-сервисов показателями производительности могут быть время реакции (время выполнения бизнес-транзакции) или пропускная способность системы. Например, при задании времени реакции системы пользователь может потребовать чтобы время проводки по счету клиента было не более 5 сек., а при задании производительности – количество транзакций по счету клиента было не менее 20 в течении одного часа то есть 20 транзакций/ч. Для задания производительности ИТ-сервиса следует использовать бизнес-операции (бизнес-функции), существенные для конечного пользователя, – ввод документов, подготовку отчетов и т.д.

Конфиденциальность определяет вероятность несанкционированного доступа к данным и/или их несанкционированное изменение. Количественные измерения данного показателя обычно не проводятся. Вместо этого ИС, обеспечивающие ИТ-сервис, классифицируются по степени конфиденциальности. Принадлежность ИС к тому или иному классу подтверждается независимой сертификацией. Конфиденциальность ИТ-сервиса в целом определяется классом безопасности наиболее слабой из обеспечивающих сервис ИС, а также корректируется с учетом качества инструкций для конечных пользователей и их обучения.

Масштаб характеризует объем и сложность работ по поддержке ИТ-сервиса. Единого измерителя масштаба не существует, к его показателям относятся число рабочих мест, количество удаленных сайтов, сложность используемых приложений и т.п.

Затраты – стоимость всей совокупности ресурсов, вовлеченных в сопровождение ИТ-сервиса, а также потерь от простоев ИТ-сервиса. В ресурсы включаются стоимость оборудования, ПО, используемых ресурсов СКС и каналов связи, внешних услуг, заработная плата сотрудников организации (включая связанные с ней расходы) и т.д.

Параметры сервиса определяются не только свойствами ИС, которые его обеспечивают. Существенное значение имеет качество работы самой службы ИС, а также уровень регламентации деятельности службы ИС и конечных пользователей ИТ-сервисов.

Важным фактором эффективности деятельности службы ИС является инструментальная поддержка автоматизации процессов управления информационными технологиями предприятия, которая в значительной степени может способствовать снижению затрат на управление и мониторинг ИС с целью предоставления ИТ-сервисов требуемого качества.

2.2 Функциональные области управления службой ИС

Информационная система предприятия предназначена для информационной поддержки бизнес-процессов.

В наши дни основой успешного бизнеса является бесперебойное функционирование информационных систем, обеспечивающих конкурентоспособность и прибыльность компании. Основная задача службы ИС – обеспечение бизнес-процессов информационным обслуживанием заданного качества с использованием соответствующих информационных технологий. Поддержка информационных процессов осуществляется посредством ИТ-сервисов с заданными характеристиками.

Служба ИС предприятия, как правило, организует свою работу по четырем функциональным направлениям [7]:

- планирование и организация;
- разработка, приобретение и внедрение;
- предоставление и сопровождение ИТ-сервиса;
- мониторинг.

В рамках направления «Планирование и организация» решаются задачи разработки стратегии в области ИТ, координации развития ИТ организации, планирования ресурсов службы ИС (бюджет, человеческие ресурсы, внешние услуги и др.), управления рисками, управления качеством.

Основная задача направления «Разработка, приобретение и внедрение» – внедрение новых ИС.

Функциональное направление «Предоставление и сопровождение сервиса ИТ» обеспечивает формализацию требований подразделений-заказчиков к ИТ-сервисам, согласование требований к сервисам с соответствующими ресурсами службы ИС и предоставление конечным пользователям сервисов ИТ, соответствующих согласованным требованиям.

Основная задача направления «Мониторинг» – аудит процессов службы ИС.

Организационная структура службы ИС зависит от многих факторов:

• масштаб службы ИС – более крупные службы ИС обычно имеют более сложную и разветвленную организацией структуру;

• отраслевую принадлежность, с которой связано наличие или, напротив, отсутствие определенных структурных подразделений;

• распределение организации по территории – наличие территориально удаленных подразделений и филиалов существенно меняет организационную структуру службы ИС.

Этот перечень отнюдь не исчерпывающий, в него входят и другие факторы, например состав используемых в организации ИС.

Для малых предприятий организационная структура службы ИС, описанная в [7], представлена на рисунке 2.2.

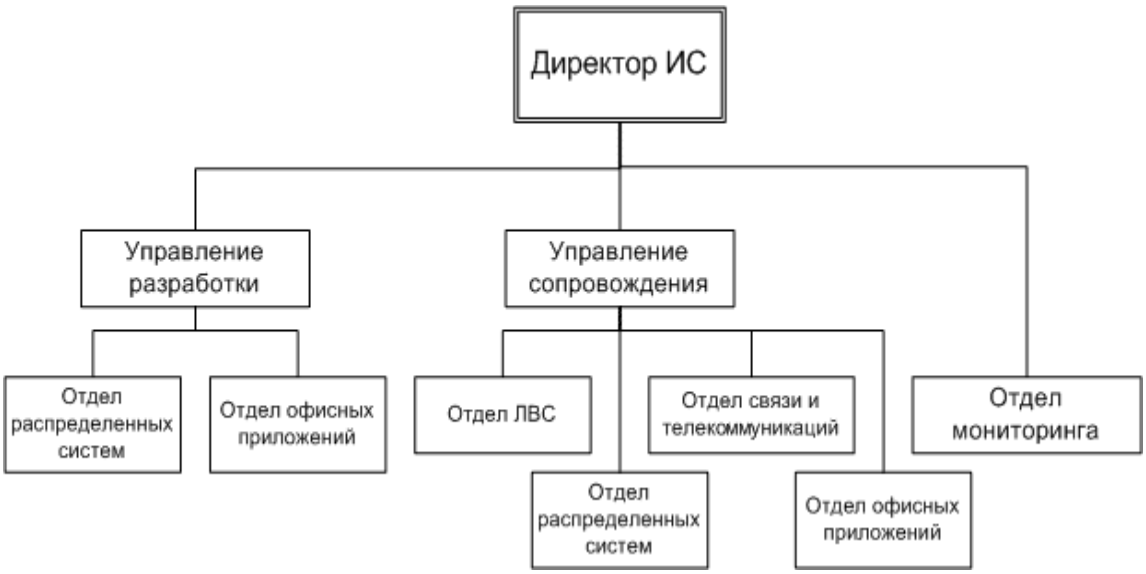


Рисунок 2.2 – Пример плоской структуры службы ИС

Функции планирования в ней выполняются руководителем службы ИС. Именно по этой причине такая структура пригодна только для службы ИС небольшого размера – в более крупных службах ИС объем работ по планированию требует обособления отдельных функций планирования.

Непосредственно подчиняются директору ИС управление разработкой, выполняющее функции разработки, приобретения и внедрения информационных систем, и управление сопровождением,

выполняющее функции предоставления и сопровождения ИТ-сервисов. Организационное разделение разработки и эксплуатации имеет принципиальное значение. Успешная эксплуатация ИС в течение сколько-нибудь длительного времени возможна лишь тогда, когда она не требует постоянного вмешательства разработчика. Это обеспечивается соблюдением существующих методологий разработки и тестирования ИС, а также надлежащей пользовательской и эксплуатационной документацией.

Тестирование ИС и документации на нее на соответствие требованиям устойчивой эксплуатации обеспечивается в ходе передачи системы в эксплуатацию. Этот процесс и определяет важность разделения двух функциональных направлений. Передача ИС от одного управления службы ИС другому, равноправному первому, обеспечивает всестороннее тестирование созданной ИС и документации на нее. Напротив, внутри одного управления передача в эксплуатацию осуществляется обычно формально, с учетом возможности последующих доработок. Таким образом, во втором случае качество эксплуатируемой ИС обычно оказывается ниже.

В рамках процесса разработки одна и та же группа – проектная команда, подчиненная одному руководителю, – должна последовательно выполнить все функции процесса разработки применительно к определенной ИС. Следовательно, распределение функций разработки по различным подразделениям не имеет смысла. Напротив, имеет смысл выделить различные проектные группы для различных видов ИС, требующих от сотрудников различных знаний и навыков.

В результате в нашем примере выделены два отдела разработки – отдел офисных систем и отдел распределенных систем. Офисные системы представляют собой разработки в среде пакета MS Office, распределенные системы – многопользовательские системы, специализированные для выполнения отдельных задач. В малых организациях типичным примером таких задач и соответственно ИС являются бухгалтерские системы. Отдел офисных систем решает задачи «малой автоматизации» задач пользователей в среде MS Office. Отдел распределенных систем занимается внедрением бухгалтерской системы, а после того как внедрение завершено, расширением ее функциональности – внедрением дополнительных модулей, написанием отчетов и других программ в среде данной распределенной системы. Наконец, в штате управления разработкой

необходим хотя бы один менеджер проектов. В простейшем случае им может быть руководитель управления разработкой, однако совмещение этих двух позиций может стать узким местом проектов этого управления. Таким образом, директор ИС должен отслеживать ситуацию с управлением проектами и при необходимости расширить управление разработкой за счет одного или нескольких менеджеров проектов.

В управлении сопровождением выделяют группы специалистов сходной квалификационной базы. Отделами, состоящими из сотрудников сходной квалификации, проще управлять, поскольку однородность упрощает найм персонала, диспетчирование работ, бюджетирование и др. Типичный набор отделов в управлении сопровождением в плоской структуре включает отдел ЛВС (локальной вычислительной сети), отдел распределенных систем, отдел связи и телекоммуникаций, отдел офисных приложений. Первый отдел осуществляет поддержку локальной сети, включая сервер и его ОС, второй – поддержку распределенных систем, например бухгалтерской, третий – связь, телефонизацию и доступ в Интернет, четвертый – поддержку оборудования рабочих мест – компьютеров, принтеров и т.д., а также офисных приложений.

Функции мониторинга в плоской структуре выполняет отдел мониторинга (Service Desk), непосредственно подчиненный директору ИС. В этот отдел поступают сообщения пользователей об инцидентах, он же сообщает об инциденте соответствующим отделам службы сопровождения и контролирует ход работ по разрешению инцидента. Наконец, в этом отделе накапливается большой объем статистики инцидентов и времени их разрешения. Функции мониторинга более высокого уровня – контроль планов работ, графиков проектов, бюджета службы ИС в целом и отдельных ее подразделений – выполняет директор ИС.

Увеличение размера организации и объема работ службы ИС ведет к усложнению её организационной структуры. В этом случае могут применяться развернутые и дивизиональные структуры службы ИС.

Функциональная модель управления и основанная на ней организационная структура службы ИС длительное время представляли собой основной и единственный подход к управлению в этой области. Однако со временем выявился ряд ограничений

функционального подхода, снижавших эффективность управления службой ИС.

Функции службы ИС должны обеспечивать создание конечного продукта – ИТ-сервисов, поддерживающих выполнение определенных бизнес-процессов.

Функциональность ИТ-сервиса затрагивает большое количество функций службы ИС. На этапе планирования ИТ-сервиса функциональность согласовывается со стратегией, стандартами и планами в рамках стратегических функций службы ИС: контролируется соответствие создаваемого сервиса ИТ-стратегии предприятия, принятым стандартам и нормам службы ИТ, а также наличие средств в бюджете предприятия. На этапе разработки и внедрения функциональность ИТ-сервиса обеспечивается всеми функциями направления разработки и внедрения. Наконец, на этапе эксплуатации ИТ-сервиса функциональность обеспечивается управлением данными, оборудованием и системным программным обеспечением и поддержкой конечных пользователей. Соответствующие функции отдела сопровождения и эксплуатации обеспечивают учет связанных с сопровождением ИТ-сервиса расходов, а функции отдела мониторинга – соблюдение условий соглашений между заказчиком и службой ИС, с одной стороны, и службой ИС и внешними поставщиками – с другой.

Время обслуживания, доступность, надежность и производительность сервиса определяется в ходе согласования требований к ИТ-сервису с заказчиком и далее контролируется функциями мониторинга. Обеспечиваются эти параметры функциями поддержки конечных пользователей (устранение возникших сбоев) и управления данными, оборудованием и системным ПО (предотвращение возникновения сбоев и/или снижение их количества). Данные по производительности операций, существенных для конечного пользователя, могут быть получены на основании статистики использования прикладных систем.

Конфиденциальность ИТ-сервиса на этапе планирования формулируется в рамках функции определения политики безопасности отдельных сервисов. На этапе создания ИТ-сервиса в рамках функций разработки, приобретения и внедрения сервиса реализуется необходимая инфраструктура безопасности – разделение полномочий на доступ к операциям и документам, присвоение прав пользователям, шифрование данных и т.д. Наконец, на этапе

эксплуатации сервиса осуществляются обучение пользователей и контроль выполнения требований безопасности на рабочих местах конечных пользователей.

Масштаб сервиса определяется на этапе планирования сервиса в рамках функции планирования сервиса ИТ. Если некие сервисы ИТ реализуются совместно в рамках общего проекта, эти сервисы должны планироваться совместно. Обеспечение доступа к ИТ-сервису на всех серверах и рабочих местах реализуется в рамках функций приобретения, разработки и внедрения. Изменения масштаба сервиса контролируются в рамках функций планирования и организации.

Цена ИТ-сервиса определяется в процессе планирования сервиса. На этапе разработки и внедрения ИТ-сервиса контролируется выполнение бюджета соответствующего проекта и уточняется сумма первоначальных затрат на приобретение и/или разработку и внедрение. На этапе эксплуатации контролируется величина текущих затрат на сервис и их соответствие бюджету организации.

Таким образом, между функциями службы ИС и параметрами ИТ-сервиса нет прямого и однозначного соответствия. Качество ИТ-сервиса в целом и каждый *параметр* сервиса ИТ, в частности, определяются несколькими функциями ИТ. Одна и та же функция службы также может относиться к нескольким сервисам ИТ или даже ко всем сервисам ИТ, существующим в организации. Это обстоятельство создает для управления службой ИС, организованной по чисто функциональному принципу, целый ряд проблем.

Во-первых, обеспечение конечного результата – качества ИТ-сервиса – требует координации различных функций службы ИС. В ряде случаев эту координацию может осуществить вышестоящий руководитель. Однако многие задачи по такой координации требуют полномочий высокого уровня, вплоть до уровня директора ИТ. В результате руководители высокого уровня оказываются перегруженными большим потоком задач, не имеющих отношения к их постоянной деятельности и непосредственным обязанностям.

Во-вторых, управление подразумевает ответственность, и коль скоро параметры сервиса определяют качество последнего, следует назначить лиц, ответственных за эти параметры. При этом сфера ответственности не должна превышать полномочий ответственного лица. Из проведенного анализа прямо следует, что в целом содержание, доступность, надежность, производительность и

конфиденциальность ИТ-сервиса находятся исключительно в сфере полномочий директора ИТ. Такой объем обязанностей директора ИТ возможен в плоской структуре службы ИС, но абсолютно нереалистичен для развернутой или дивизиональной структуры. В результате лицо, ответственное за качество сервиса, при функциональной организации службы ИС отсутствует.

В-третьих, проблемой является «точка контакта» – телефон и/или адрес электронной почты, по которому следует обращаться в случае необходимости. Наличие такой «точки контакта» особенно удобно в случае возникновения у пользователя потребности в новом или измененном ИТ-сервисе, а также при необходимости сообщить о сбое. При этом «точка контакта» может быть использована не только для регистрации запроса пользователя, но и для обработки его – назначения запроса специалисту, контроля хода выполнения работ, информации пользователя. Однако в функциональной организации эту дополнительную обработку организовать затруднительно. Специалисты, обрабатывающие запрос пользователя, не находятся в подчинении службы мониторинга (*Service* Desk) и не ответственны перед этой службой.

Таким образом, функциональная организация обеспечивает лишь текущую деятельность службы ИС, а не решение всех необходимых управленческих задач. С точки зрения обеспечения конечного результата – ИТ-сервиса необходимого качества – основными проблемами являются:

- координация функций;
- трудности обеспечения ответственности;
- трудности обеспечения единой «точки контакта».

Эти трудности успешно преодолеваются при процессном подходе к управлению службой ИС.

Процесс подразумевает наличие цели, критерия результата, ресурсов и определенной последовательности работ (то есть шагов процесса). Применительно к процессам службы ИТ целью является предоставление заказчику ИТ-сервиса приемлемого уровня качества. Эта общая задача может быть разделена на две более частных:

- определение и согласование параметров ИТ-сервиса;
- обеспечение соответствия фактических параметров ИТ-сервиса достигнутым соглашениям.

• Каждая из этих целей, в свою очередь, распадается на несколько целей следующего порядка, каждой из которых соответствует свой процесс.

Управление процессами предполагает следующие шаги:

- определение цели процесса и показателей достижения этой цели (количественных или качественных);
- назначение ответственного за процесс, задачей которого является достижение цели процесса;
- регламентация процесса в целом и составляющих его работ;
- при необходимости – автоматизация процесса посредством инструментальных средств, разработанных в самой организации либо закупленных извне.

Проблемы ответственности за результат процесса и координации разрешается в явном виде посредством назначения ответственного лица – менеджера процесса. Проблема единой «точки контакта» также вполне разрешима в рамках регламента процесса, обязательного для всех сотрудников службы ИС независимо от их функционального подчинения.

Управление процессами изменяет лишь управленческие функции службы ИС, не затрагивая функции собственно разработки и сопровождения ИТ-сервисов. Изменения состоят в систематическом целенаправленном решении задач координации функций в ходе выполнения процессов службы ИС. Для этого достаточно формализовать соответствующий процесс, то есть назначить менеджера процесса, определить роли участников процесса и установить правила его выполнения, то есть последовательность выполнения операций процесса, обязанности в рамках ролей, правила эскалации и т.д.

Как следствие переход к процессной модели управления обычно не требует ни дополнительного персонала, ни изменений в организационной структуре. Участники процесса выполняют свои должностные обязанности в рамках существующей организационной структуры; часть этих обязанностей, относящаяся к данному процессу, формализована в виде ролей процесса. Если все процессы службы ИС формализованы, то совокупность ролей совпадает с должностными обязанностями сотрудника (рисунок 2.3).

В такой системе менеджер процесса является начальником без подчиненных: он координирует деятельность не подчиненных ему сотрудников, относящихся к различным подразделениям

существующей организационной структуры. Сам менеджер процесса тоже имеет должность в рамках существующей организационной структуры.

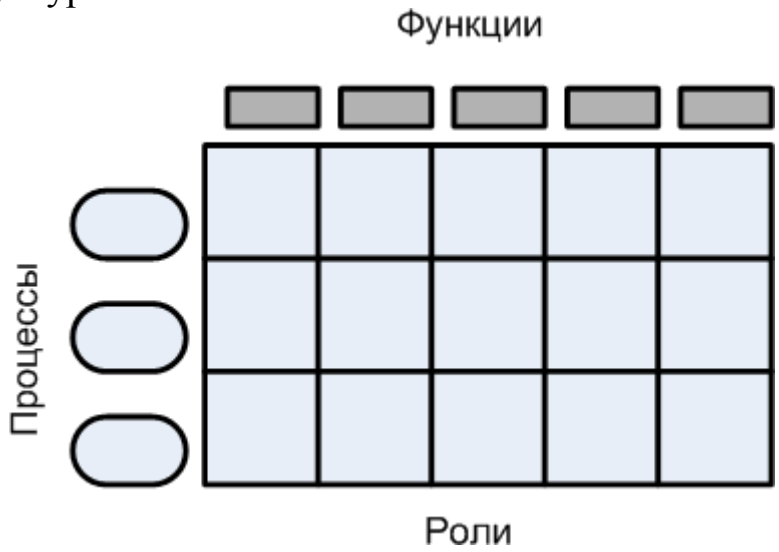


Рисунок 2.3 – Процессы, функции, роли в процессной модели управления

Использование процессов в рамках существующей функциональной структуры весьма удобно. В ходе работы по этой схеме процессная модель и функциональная структура организации взаимодействуют и усиливают преимущества друг друга.

Совместное использование обеих моделей также упрощает внедрение процессной модели. Процессная модель влияет не на полномочия функциональных менеджеров, а на формы осуществления этих полномочий. Процессные менеджеры принимают на себя задачу координации функций, которая в чисто функциональной модели решается на излишне высоком уровне.

Переход к процессной модели можно осуществить двумя путями:

- первый состоит в формализации опыта данной организации;
- второй предполагает использование передового опыта управления службой ИС, который реализован в типовых моделях бизнес-процессов этой службы.

На сегодняшний день общей методологической основой таких моделей является подход *ITIL*/*ITSM*, основанный на сборе и систематизации передовой практики управления службой ИС в течение последних 20 лет.

Использование типовых моделей бизнес-процессов службы ИС имеет целый ряд преимуществ.

Во-первых, типовая модель представляет в концентрированном виде опыт управления службой ИС в тысячах и даже десятках тысяч компаний. Соответственно, отказ от использования этого массива знаний по меньшей мере нецелесообразен.

Во-вторых, переход к процессной модели управления для всех задач службы ИС одновременно, в рамках одного проекта маловероятен. В этом случае процессная модель дает менеджеру образ будущего, который становится ориентиром в ходе отдельных шагов внедрения.

В-третьих, типовая модель процессов службы ИС всегда опирается на некую систему понятий, на некий язык. Использование этого языка значительно облегчает достижение взаимопонимания участников процесса.

В-четвертых, типовая модель процессов поддержана разработчиками программного обеспечения автоматизации управления службой ИС и инфраструктурой ИТ. В результате программное обеспечение реализует именно эти процессы. Реализация собственных процессов потребует разработки собственного ПО.

Наконец, стандартная модель процессов обычно внедряется во многих организациях. В результате образуется сообщество пользователей, которое является ценным источником информации *по* внедрению модели.

Вопросы для самоконтроля

1. Поясните понятие ИТ-менеджмента.
2. Перечислите основные объекты ИТ-менеджмента.
3. Что определяет инфраструктура ИТ-предприятия?
4. Чем обусловлены постоянные изменения в ИС предприятий?
5. Поясните понятие «ИТ-сервис».
6. Приведите примеры корпоративных ИТ-сервисов.
7. Перечислите основные характеристики ИТ-сервисов.
8. Как задается характеристика «время обслуживания» для ИТ-сервиса?
9. Как задается характеристика «производительность» для ИТ-сервиса?
10. Почему в организационной структуре службы ИС целесообразно выделять подразделения разработки и сопровождения ИС?

3 ITIL/ITSM – концептуальная основа процессов ИС-службы

3.1 Общие сведения о библиотеке ITIL

В настоящее время ИТ-служба предприятия становится полноправным участником бизнеса, выступая в роли поставщика определенных услуг для бизнес-подразделений, а отношения между ними формализуются как отношения «поставщик услуг – потребитель услуг». Бизнес-подразделение формулирует свои требования к необходимому спектру услуг и их качеству, руководство предприятия определяет объем финансирования для выполнения этих требований, а *подразделения* ИТ-службы поддерживают и развивают информационную инфраструктуру предприятия таким образом, чтобы она была в состоянии обеспечить запрошенную услугу с заданным качеством.

Отражением трансформации роли и места ИТ-службы в структуре предприятий является концепция и модель управления качеством информационных услуг (*Information Technology Service Management – ITSM*, управление ИТ-услугами) [9]. Бизнес-процессы сегодня неразделимы с программными приложениями, техническими ресурсами и деятельностью персонала ИТ-служб, поэтому качество работы последних становится важнейшим фактором, определяющим эффективность деятельности предприятия в целом.

Модель *ITSM* является открытой для изменения со стороны пользователей и описывает совокупность процессов службы ИС. Это позволяет настраивать процессы *ITSM* для конкретного применения. Существует большое количество инструментальных средств, реализующих модели процессов *ITSM*, разработанных компаниями-консультантами и производителями программного обеспечения управления инфраструктурой ИТ. Модель *ITSM* не дает ИТ-менеджеру службы ИС однозначных рекомендаций как конкретно строить систему управления информационной инфраструктурой предприятия. В то же время концепция *ITSM* содержит модель *типовых процессов* службы ИС, понятийный аппарат, на основе которых целесообразно строить модели процессов для ИТ-службы.

Модель *ITSM*, разработанная в рамках проекта *ITIL* (IT *Infrastructure Library* – библиотека инфраструктуры информационных технологий, произносится как « *айтил* «), описывает *процессный*

подход к предоставлению и поддержке ИТ-услуг [7], [9]. Данная модель получила наибольшую известность в силу того, что предоставление и *поддержка* ИТ-услуг является первичной задачей ИТ-службы предприятия.

В отличие от более традиционного *функционального подхода* к организации ИТ-службы, *ITSM* рекомендует сосредоточиться на клиенте и его потребностях, на ИТ-услугах, предоставляемых пользователю информационными технологиями, а не на них самих. При этом процессная организация предоставления услуг и наличие заранее оговоренных уровней параметров эффективности позволяет ИТ-службе предоставлять качественные ИТ-услуги, измерять и улучшать их качество.

По проекту *ITIL* была разработана библиотека, описывающая лучшие из применяемых на практике способов организации работы подразделений или компаний, занимающихся предоставлением услуг в области информационных технологий [9]. Множество частных и государственных компаний в разных странах мира, включая и Россию, добились значительных успехов в повышении качества ИТ-сервисов, следуя изложенным в *ITIL* рекомендациям и принципам. В настоящее время *ITIL* становится стандартом де-факто для ИТ.

Библиотека *ITIL* создавалась *по* заказу британского правительства. В настоящее время она издается британским правительственным агентством Office of *Government Commerce* и не является собственностью ни одной коммерческой организации. В семи томах библиотеки описан весь набор процессов, необходимых для того, чтобы обеспечить постоянное высокое качество ИТ-сервисов и повысить степень удовлетворенности пользователей. Следует отметить, что все эти процессы нацелены не просто на обеспечение бесперебойной работы *компонент* ИТ-инфраструктуры. В гораздо большей степени они нацелены на выполнение требований пользователя и заказчика.

Особенностью проекта является свобода использования его результатов:

- ограничений на использование нет;
- материалы модели могут быть использованы полностью или частично;
- модель может быть использована в точном соответствии с текстом книг *ITIL* либо адаптирована пользователем.

При этом модель сегодня является наиболее широко распространенным в мире подходом к управлению ИТ-сервисами. Она применима к организациям любого размера и любой отраслевой принадлежности.

Текущая версия библиотеки *ITIL* включает 7 книг *по* основным разделам управления ИТ-сервисами [7]:

- *Service Delivery* (предоставление услуг) – содержит описание типов ИТ-услуг, предоставляемых предприятием;
- *Service Support* (поддержка услуг) – представляет собой описание процессов, позволяющих обеспечить пользователям доступ к ИТ-услугам, необходимым для выполнения бизнес-задач;
- Information & *Computing Technology* Infrastructure Management (управлениеИТ-инфраструктурой). В книге представлено общее описание методики организации работы ИТ-службы по управлению ИТ-инфраструктурой компании;
- *Application Management* (управление приложениями) указывает, как обеспечить соответствие программных приложений изменениям в потребностях бизнеса, а также рассматривает общий жизненный цикл приложений, включающий разработку, внедрение и сопровождение;
- The Business *Perspective* (бизнес-перспектива) – рассматривается, как работа ИТ-инфраструктуры может влиять на бизнес компании в целом;
- Planning to Implement Service Management (планирование внедрения управления услугами) – посвящена проблемам и задачам планирования, реализации и развития *ITSM*, необходимым для реализации поставленных целей;
- *Security Management* (управление безопасностью) – посвящена проблемам безопасности. В ней рассматриваются проблемы разграничения доступа к информации и ИТ-сервисам, особенности оценки, управления и противодействия рискам, инциденты, связанные с нарушением безопасности и способы реагирования на них.

В третьей, разрабатываемой версии библиотеки *ITIL* (проект *ITIL Refresh*), представлено пять книг, названия которых отражают жизненный цикл ИТ-услуг:

- «Стратегии обслуживания» (Service *Strategies*);
- «Проектирование услуг» (Service Design);

• Внедрение услуг» (Service Introduction); «Оказание услуг» (Service Operation);

• «Непрерывное совершенствование услуг» (Continuous Service Improvement).

В Европе существуют два центра сертификации специалистов по модели ITIL/ITSM – EXIN (Нидерланды – Голландский Экзаменационный Институт) и ISEB (The Information Systems Examination Board – подразделение Британского Компьютерного Общества – British Computer Society). Внедрением процессов ITIL/ITSM и обучением занимается целый ряд компаний-консультантов. В России это Hewlett-Packard Consulting, «Ай-Теко», IT-Expert.

Модель ITIL/ITSM поддерживается более чем десятком программных продуктов и пакетов. Лидерами разработки программных инструментов управления ИТ-инфраструктурой являются: Hewlett-Packard, Computer Associated, IBM, BMC Software и Microsoft. Среди российских компаний, поставщиков программных систем автоматизации управления ИТ-услугами следует отметить компании СофтИнтегро и Итилиум.

Важным элементом инфраструктуры ITIL/ITSM являются так называемые ITSM-форумы. Эти форумы представляют собой сообщества пользователей модели, консультантов, внедряющих модель, и производителей инструментального программного обеспечения. Сообщество, как правило, имеет сайт в сети Интернет (например, ITSM ПОРТАЛ.RU), а также проводит конференции и другие мероприятия, обеспечивающие реальное общение участников. Так российское партнерство «Форум по ИТ Сервис-менеджменту» получило международную аккредитацию ITSMF и стало полноправным членом всемирного сообщества. ITSMF International представляет собой независимое сообщество профессионалов в области управления ИТ-услугами. Оно было создано в Великобритании в 1991 году и занимается пропагандой идей ITSM, разработкой стандартов в этой области и поддержкой обмена опытом в десятках стран мира. На сегодняшний день национальные отделения ITSMF действуют уже в 41 стране мира. ITSMF Russia было образовано в 2005 году и на сегодняшний день объединяет около 200 представителей из более, чем 45 российских компаний.

Внедрение методики управления *ITSM* – поэтапный процесс. Как показывает практика, решение первоочередных задач связано с

рекомендациями, приведенными в первых книгах «*Поддержка* сервисов» и «Предоставление сервисов». Процессы группы предоставления сервисов считаются оперативными процессами, поскольку включают в себя повседневные функции ИТ-службы. Процессы группы поддержки сервисов относятся к тактическим, которые предназначены для обеспечения предоставления сервисов заданного качества.

3.2 Процессы поддержки ИТ-сервисов

Блок процессов поддержки ИТ-сервисов включает следующие процессы [10]:

1. управление инцидентами;
2. управление проблемами;
3. управление конфигурациями;
4. управление изменениями;
5. управление релизами.

Процесс управления инцидентами предназначен для обеспечения быстрого восстановления ИТ-сервиса. При этом **инцидентом** считается любое событие, не являющееся частью нормального функционирования ИТ-сервиса. К инцидентам относятся, например, невозможность загрузить операционную систему, сбой электропитания, сбой жесткого диска на рабочей станции пользователя, появление компьютерного вируса в локальной сети офиса, отсутствие тонера или бумаги для печатающего устройства и т.д. Показателями качества реализации процесса являются:

- временная продолжительность инцидентов;
- число зарегистрированных инцидентов.

При реализации процесса должны выполняться следующие функции:

- прием запросов пользователей;
- регистрация инцидентов;
- категоризация инцидентов;
- приоритизация инцидентов;
- изоляция инцидентов;
- эскалация инцидентов;
- отслеживание развития инцидента;
- разрешение инцидентов;
- уведомление клиентов;

- закрытие инцидентов.

Необходимым элементом обеспечения эффективного функционирования процесса является создание службы поддержки пользователей (*Help Desk*), единой точки обращения *по* поводу различных ситуаций в ИТ-инфраструктуре, обработки и разрешении пользовательских запросов. Следует отметить, что роль службы поддержки пользователей в последнее время возрастает, что отражается в её модифицированном названии – *Service* Desk. Это говорит о том, что современные службы поддержки переориентируются с реактивного принципа работы, на проактивный, позволяющий анализировать ситуацию и предотвращать инциденты еще до их возникновения.

Для управления качеством процесса необходимо определить систему управления инцидентами, разработать управленческие отчеты и обеспечивать непрерывное *улучшение* процесса.

На рисунке 3.1 приведена *диаграмма активности* для процесса Управление инцидентами. *Пользователь* ИТ-сервиса обнаруживает нарушение режима предоставления сервиса и обращается в *Service* Desk ИТ-службы. Сотрудник *подразделения Service* Desk фиксирует в регистрационном журнале инцидент, классифицирует его, определяет приоритет и при возможности осуществляет начальную поддержку. Например, при невозможности для пользователя корректно завершить транзакцию предлагается перезагрузить операционную систему и повторно провести транзакцию. Если начальной поддержки пользователю достаточно и не требуется специализированная *поддержка*, то осуществляется закрытие инцидента. Если необходимо специализированное обслуживание, то *информация по* инциденту передается в подразделение сопровождения ИТ-сервисов. В этом подразделении на основе базы знаний выясняется возможность устранения инцидента оперативным персоналом, то есть, нет необходимости эскалации инцидента на более высокий уровень обслуживания. В этом случае оперативный персонал реализует ранее документированную процедуру восстановления ИТ-сервиса.

Если для устранения инцидента отсутствует решение в базе знаний, то осуществляется эскалация на следующий уровень обслуживания, где специалисты высокого класса проводят изучение и диагностику инцидента, разрабатывают методы его устранения, восстановления заданной работоспособности ИТ-сервиса и пополняют базу знаний *по* инцидентам. После закрытия инцидента

для пользователя предоставляется возможность доступа к ИТ-сервису с требуемыми показателями качества. Момент закрытия инцидента фиксируется в журнале службы *Service*Desk.

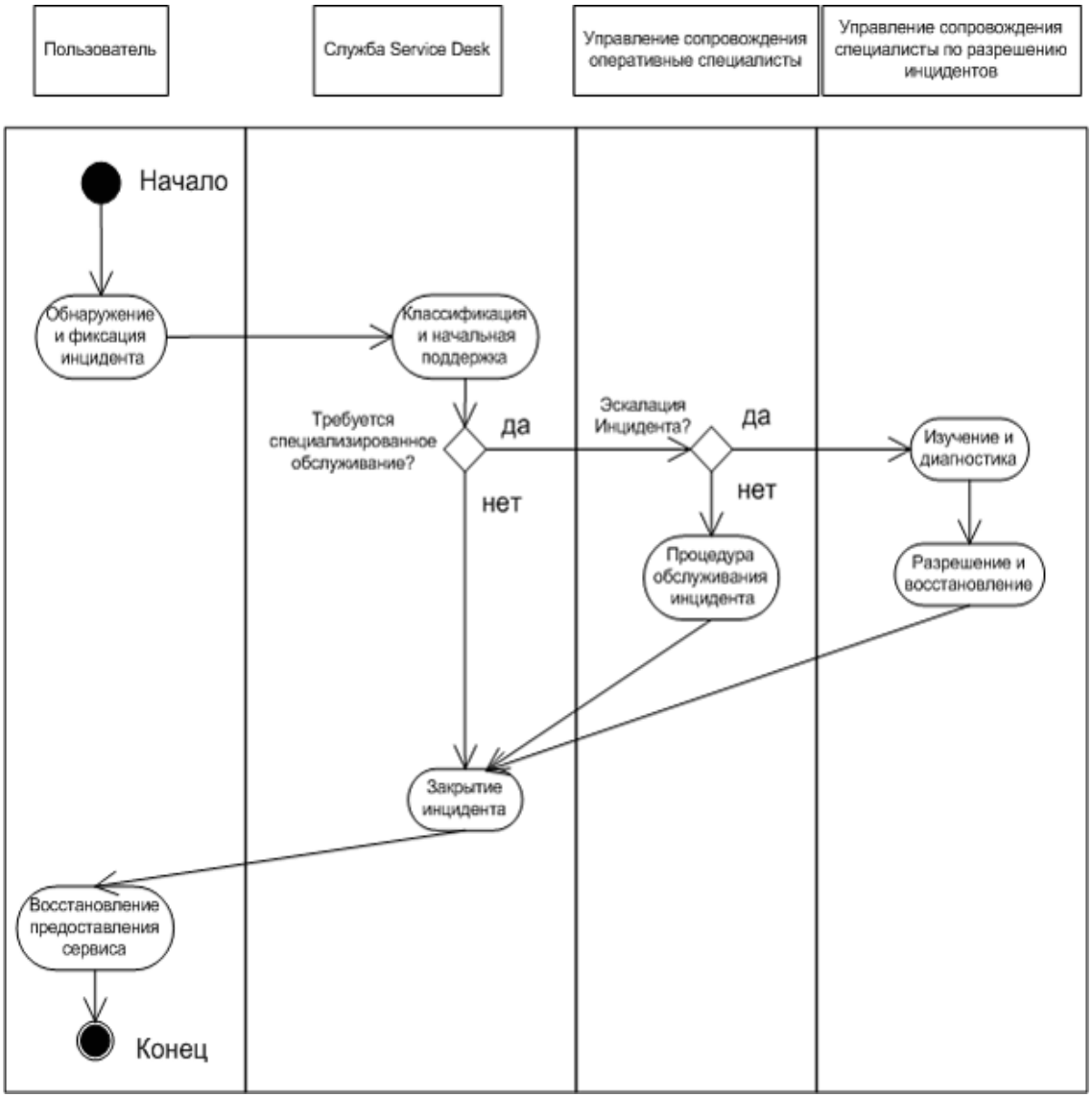


Рисунок 3.1 – Диаграмма активности процесса управления инцидентами

Процесс управления проблемами предназначен для минимизации негативного влияния инцидентов на бизнес и уменьшения количества инцидентов, за счет предотвращения возможных причин инцидентов. В данном контексте под *проблемой* понимают инцидент или группу инцидентов, имеющих общую неизвестную причину.

При реализации процесса должны выполняться следующие функции:

- *анализ тенденций* инцидентов;
- регистрация проблем;
- идентификация корневых причин инцидентов;
- отслеживание изменений проблем;
- выявление известных ошибок;
- управление известными ошибками;
- решение проблем;
- закрытие проблем.

Для управления качеством процесса необходима организация системы управления проблемами/известными ошибками, организация превентивных процедур поддержки, организация способов верификации известных ошибок, организация интерфейса поддержки поставщиком, разработка отчетов для управления, постоянное усовершенствование процесса.

На рисунке 3.2 приведена *диаграмма активности* для процесса Управление проблемами.

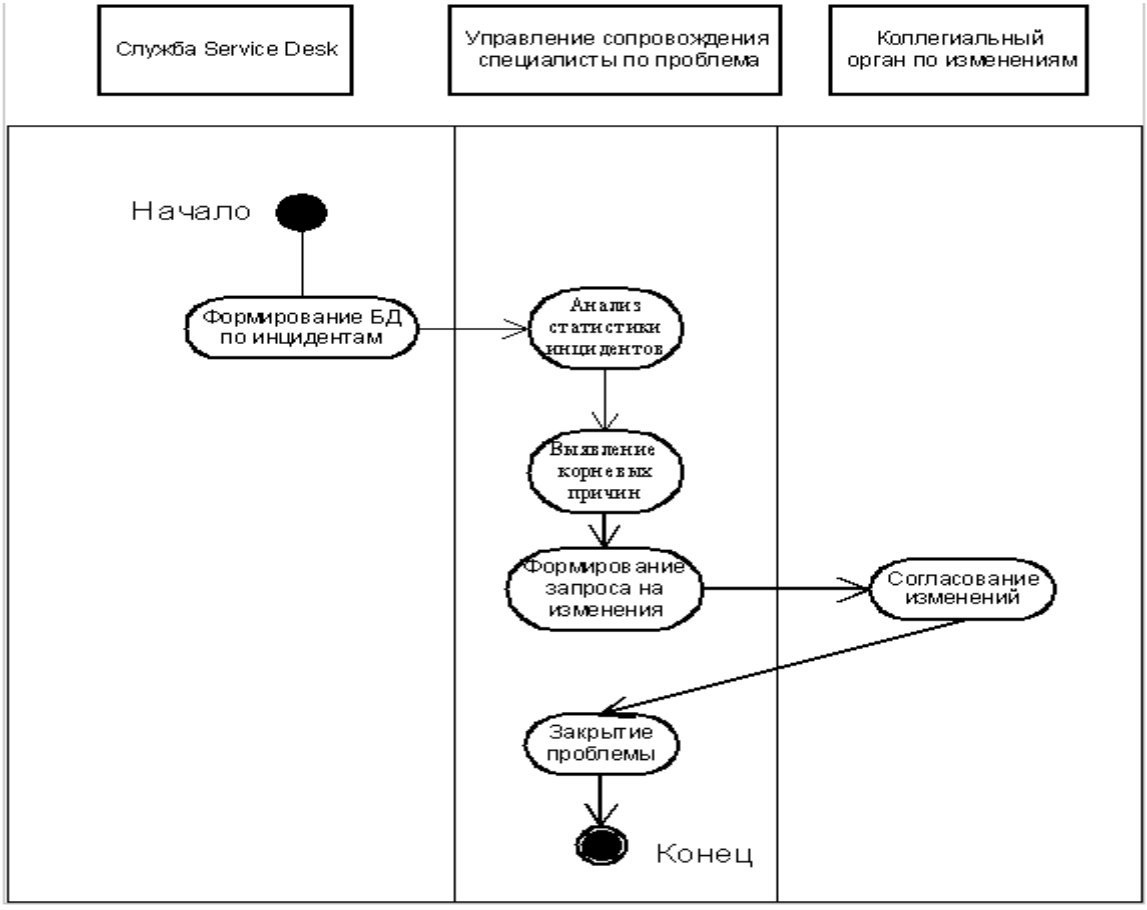


Рисунок 3.2 – Диаграмма активности процесса управления проблемами

Процесс управления конфигурациями предназначен для оказания помощи в управлении экономическими характеристиками ИТ-сервисов (комбинация требований клиентов, качества и затрат) за счет поддержания логической модели инфраструктуры ИТ и ИТ-сервисов, а также предоставление информации о них другим бизнес-процессам. Это реализуется путем идентификации, мониторинга, контроллинга и обеспечения информации о конфигурационных единицах (CI – *Configuration Item*) и их версиях. Конфигурационные единицы описывают системные компоненты с их конфигурационными атрибутами.

Процесс Управление конфигурациями отвечает за поддержание информации о взаимоотношениях между CI и за стандартизацию CI,*мониторинг* информации о статусе CI, их местоположении и всех изменениях CI. *Информация* о CI хранится в базе данных конфигурационных единиц (*Configuration Management Data Base* – CMDB). *База данных* управления конфигурациями представляет собой *репозиторий* метаданных, описывающий элементы конфигурации, их взаимосвязи и атрибуты. Элементы конфигурации представляют информационные компоненты, являющиеся объектами или субъектами процесса управления конфигурациями:

- материальными сущностями (серверная стойка, компьютер, маршрутизатор, модем, сегмент линии связи);
- системными или прикладными программными продуктами и компонентами;
- реализациями баз данных;
- файлами;
- потоками данных;
- нормативными или техническими документами;
- логическими или виртуальными сущностями.

Выбор классов и типов объектов конфигурации, их атрибутов, формируемых в CMDB, определяется разработчиком, в соответствии с требованиями *предметной области*. Атрибуты CI, как правило, отражают их специфические свойства и могут включать:

- идентификаторы;
- марки и названия моделей;
- серийные номера;
- сетевые адреса;
- технические характеристики;
- операционные характеристики.

Взаимосвязи CI представляют отношения, которые существуют или могут возникнуть между двумя и более CI. Как правило, *язык спецификации* модели CMDB – *XML*. На рисунке 3.3 приведен пример модели классификации конфигурации [7].

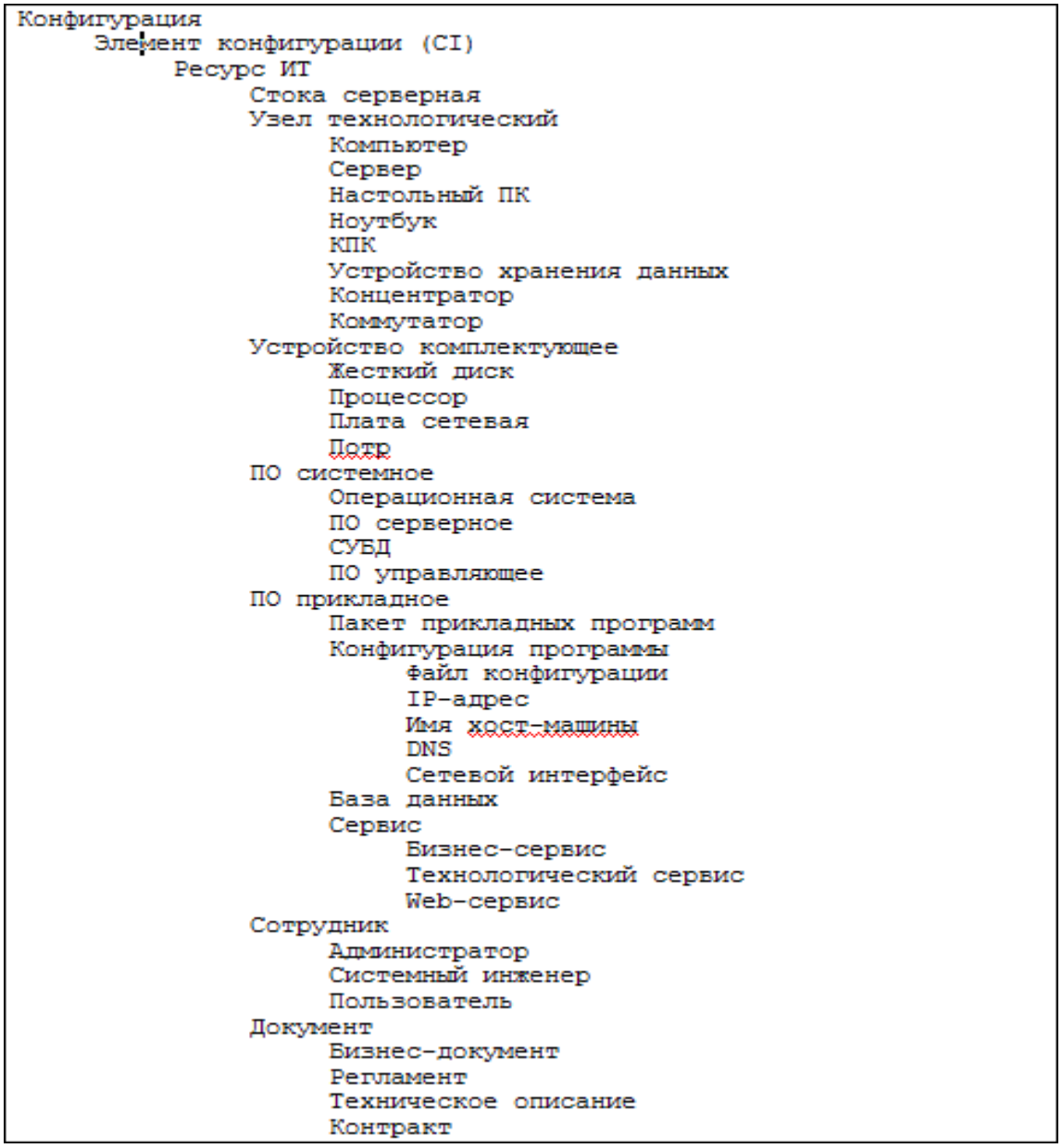


Рисунок 3.3 – Классификация элементов конфигурации

При реализации процесса управления конфигурациями должны выполняться следующие функции:

планирование – определение стратегии, правил и целей для реализации процесса, определение инструментария и ресурсов,

определение интерфейсов с другими процессами, проектами, поставщиками;

идентификация – разработка модели данных для записи в базу конфигураций всех компонент инфраструктуры ИТ, отношений между ними, а также информации о владельцах этих компонент, их статусе и соответствующей документации.

При спецификации процесса важными понятиями являются:

- сфера охвата;
- глубина детализации;
- контроль;
- мониторинг статуса;
- верификация.

Сфера охвата (*Scope*) определяет, какая часть инфраструктуры будет находиться под контролем процесса. Например, можно охватывать только сервера и маршрутизаторы. Правильный выбор Сферы охвата очень важен на начальном этапе внедрения процесса Управление конфигурациями.

Глубина детализации (*Level* of Detail) – важный аспект, определяющий в дальнейшем отношения между CI. Отношения, как правило рассматриваются физические и логические.

Физические отношения:

- родители – дети;
- соединенная.

Логические отношения, в частности, копия; когда одна единица использует другую. Например, программа использует сервер.

Контроль процесса означает, что процесс контролирует все изменения, кем бы они не производились.

Мониторинг статуса предполагает отслеживание реального статуса CI, содержащихся в базе: В *процессе жизненного цикла* информационной системы статус CI может меняться от «заказано» до «исключено из конфигурации»

Верификация предполагает проверку того, насколько *информация* в базе конфигураций соответствует реальности.

При реализации процесса необходимо формировать отчеты руководству и другим процессам для осуществления их эффективного выполнения.

Процесс управления изменениями предназначен для обеспечения уверенности ИТ-менеджера в том, что все изменения необходимы, запланированы и согласованы. Данный процесс предполагает

регистрацию всех существенных изменений в среде ИС предприятия, разрешает изменения, разрабатывает *график работ по* изменениям и организует взаимодействие ресурсов, всесторонне оценивает воздействие изменения на среду ИС и связанные с ним риски. *Диаграмма активности* процесса управления изменениями приведена на рисунке 3.4.

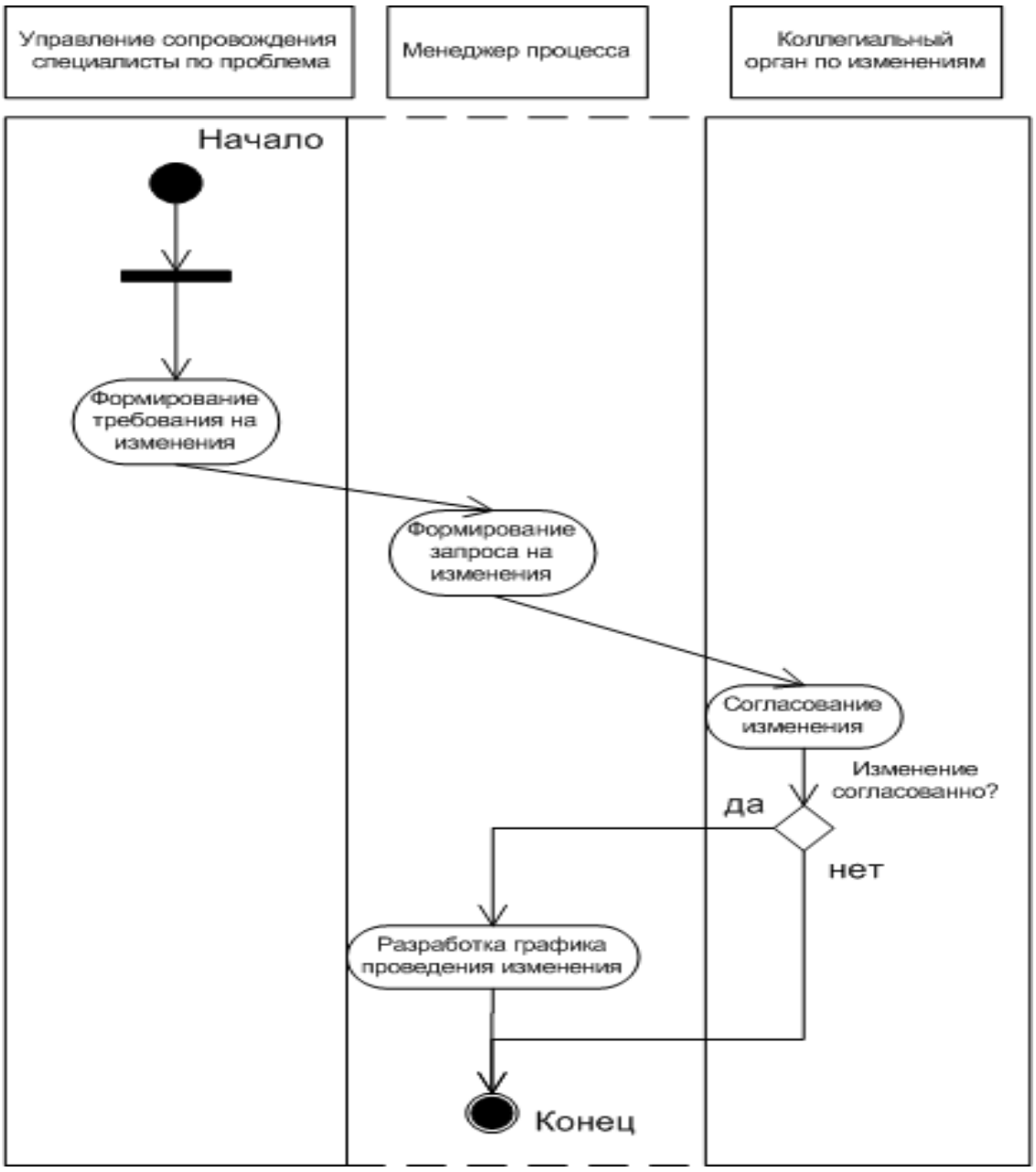


Рисунок 3.4 – Диаграмма активности процесса управления изменениями

Основная задача данного процесса – проведение только обоснованных изменений в ИТ-инфраструктуре и отсев непродуманных или потенциально рискованных изменений. Для этого каждое изменение конфигурации ИС организации в обязательном порядке оформляется запросом на изменение. *Запрос* на изменение проходит стандартную процедуру одобрения. В зависимости от масштаба изменения решение принимается на уровне менеджера процесса, комитета *по* оценке изменений в рамках службы ИС, правления организации.

Конечный результат процесса – набор изменений, согласованных между собой и с существующей конфигурацией информационной системы и не нарушающих функционирования уже существующих сервисов. Все изменения в обязательном порядке регистрируются процессом управления конфигурацией.

Процесс управления изменениями выполняет следующие функции:

- обрабатывает запросы на изменения;
- оценивает последствия изменений;
- утверждает изменения;
- разрабатывает график проведения изменений, включая восстановление при сбое;
- устанавливает процедуру обработки запроса на изменение;
- устанавливает категории и приоритеты изменений;
- управляет проектами изменений;
- организует работу комитета по оценке изменений;
- осуществляет постоянное улучшение процесса.

Важную роль в процессе управления изменениями играет коллегиальный орган *по* согласованию изменений. Этот орган включает в себя ИТ-директора (председателя), представителей бизнес-подразделений (представителей от финансовой службы и основных направлений бизнеса) и сотрудников ИС-службы, отвечающих, *по* мере необходимости, за следующие роли: планирование сервисов, управление изменениями, управление уровнем сервиса, управление проблемами. Задача коллегиального органа – планирование возможных результатов и рисков при внесении изменений в ИТ-инфраструктуру. Изменение отвергается как в случае незначительных результатов, так и в случае значительных рисков. В остальных случаях изменение может быть принято.

На основании положительного решения *по* изменениям разрабатывается *график* будущих изменений – детальный календарный *график* одобренных изменений, согласованный с заказчиками изменений, а также рядом других процессов *ITSM*.

Таким образом, процессы управления изменениями и конфигурациями обеспечивают *целостность* и согласованность информационной системы предприятия. В процессе управления изменениями эта задача решается посредством процесса одобрения изменений, предусматривающего всесторонний *контроль* за изменениями со стороны сотрудников ИС-службы, а при значительных изменениях – и руководства предприятия в целом. *Процесс управления* конфигурациями регистрирует все изменения в ИТ-инфраструктуре организации и обеспечивает все остальные процессы данными об установленных позициях оборудования и программного обеспечения, включая данные о произведенных настройках.

Процесс управления релизами предназначен для обеспечения согласованности изменений, вносимых в ИТ-инфраструктуру предприятия. Под **релизом** понимается набор новых и/или измененных позиций конфигурации, которые тестируются и внедряются совместно.

Процесс управления релизами предполагает консолидацию, структурирование и оптимизацию всех изменений или обновлений, а также снижение риска при переводе сервиса на новый качественный уровень.

Процесс управления релизами состоит из трёх этапов:

- разработка;
- тестирование;
- распространение и внедрение.

Этап разработки не является обязательным для всех предприятий. Но для некоторых компаний, данный этап может являться одним из основополагающих, к ним могут относиться, например, компании *по* разработке программных средств.

Второй этап, этап тестирования, является важным для всех предприятий без исключения. На данном этапе необходимо определить критерии, *по* которым будет проводиться тестирование для каждого релиза, что позволяет определить степень готовности релиза к распространению и внедрению.

Если *процесс Управления* релизами подготавливает реализацию принятых изменений, то необходимо определить, какой процесс ответственен за их непосредственное внедрение. Руководствуясь материалами *ITIL*, можно сделать заключение, что в некоторых случаях, например, внедрение срочных или не значительных изменений, *процесс Управления* релизами осуществляет сам, на этапе внедрения. А в некоторых случаях, возможен вариант формирования целых проектов под управлением процесса управления проектами для внедрения комплексных и глобальных изменений, затрагивающих значительные ресурсы. В любом случае, это решается непосредственно в процессе внедрения самого процесса Управления релизами в каждой конкретной ситуации.

Процесс управления релизами выполняет следующие функции:

- планирование релиза;
- проектирование, разработка, тестирование и конфигурирование релиза;
- подписание релиза в развертывание;
- подготовка релиза и обучение пользователей;
- аудит оборудования и ПО до начала внедрения изменений и по завершении такового;
- размещение эталонных копий ПО в *DSL*;
- установка нового оборудования и ПО;
- постоянное улучшение процесса.

Для оценки качества деятельности процесса важно тщательно выбирать метрики.

По масштабу релизы подразделяются на три вида:

- большой релиз ПО и/или обновление оборудования – обычно содержит значительный объем новой функциональности, которая делает ранее сделанные исправления проблем частично или полностью избыточными. Также большой релиз обычно отменяет предшествующие малые релизы;
- малый релиз ПО и/или обновление оборудования – обычно содержит незначительные улучшения, часть из которых могли быть выполнены ранее как чрезвычайные релизы. Соответственно, эти изменения отменяются малым релизом;
- чрезвычайный релиз ПО и/или обновление оборудования – обычно содержит исправления некоторого числа известных ошибок.

По способу реализации релизы подразделяются также на три вида:

- при полном релизе все компоненты релиза разрабатываются, тестируются, распространяются и внедряются вместе. В результате увеличивается трудоемкость релиза, зато повышается вероятность того, что возможные проблемы будут обнаружены и устранены на этапе разработки и тестирования и не попадут в среду промышленной эксплуатации;
- дельта-релиз, или частичный релиз, включает в себя только новые или измененные позиции конфигурации. Например, если речь идет о программном релизе, дельта-релиз включает в себя только те модули, которые были созданы или изменены с момента прошлого релиза;
- пакетный релиз включает в себя несколько различных полных или частичных релизов, которые распространяются и внедряются совместно для снижения общего числа релизов, что облегчает работу пользователей. Сами релизы могут разрабатываться и тестироваться отдельно и быть объединенными в пакет лишь на заключительных этапах.

Особой сферой ответственности процесса управления релизами является библиотека эталонного *ПО* (Definitive *Software Library – DSL*). Все позиции *DSL* отражаются как записи CMDB. Эта библиотека – физическое хранилище протестированных и подготовленных к распространению копий разработанного и покупного *ПО*, лицензий на последнее, а также пользовательской и эксплуатационной документации. *Информация* о копиях *ПО*, хранящихся в *DSL*, ведется в базе данных позиций конфигурации. Наличие такой библиотеки играет важную роль в процессе управления релизами, особенно на этапе распространения и установки *ПО*.

2.3 Процессы предоставления ИТ-сервисов

Блок предоставления ИТ-сервисов в соответствии с *ITIL* включает следующие процессы:

- процесс управления уровнем сервиса;
- процесс управления мощностью;
- процесс управления доступностью;
- процесс управления непрерывностью;
- процесс управления финансами;
- процесс управления безопасностью.

Процесс управления уровнем сервиса (*Service Level Management – SLM*) определяет, согласовывает и контролирует параметры ИТ-сервиса, определенные с точки зрения бизнеса, а не с точки зрения ИТ. Ключевая роль менеджера процесса – осуществление баланса между требованиями бизнеса и возможностями ИТ.

На основе каталога ИТ-сервисов данный процесс разрабатывает, согласовывает и документирует *соглашение об уровне сервиса* (*SLA – Service Level* Agreement) между менеджментом ИС-службы и бизнес-пользователями.

Основная задача процесса управления уровнем сервиса – согласование специфицированных требований к составу и параметрам ИТ-сервисов, с одной стороны, и объема ресурсов, предоставляемых ИТ-службе, – с другой. В рамках этой работы также уточняются приоритеты сервисов и ресурсов. Результатом такого согласования является формальный документ – *SLA*. *Соглашение об уровне сервиса* необходимо периодически пересматривать поскольку *информационные системы* предприятия подвержены изменениям, появляются необходимость в новых сервисах, модификации или отказе от уже существующих.

Данный процесс осуществляет следующие функции:

- оценивает требования пользователей к ИТ-сервисам, распределяет их по существующим сервисам и определяет потребности в специализированных сервисах;
- согласует и документирует *SLA*;
- организует контроль результативности каталога сервисов в целом и уровня отдельных сервисов;
- определяет приоритетность сервисов;
- осуществляет управление версиями *SLA*;
- готовит планы повышения качества сервиса, направленные на повышение качества существующих сервисов, или включения в *SLA* новых сервисов;
- обеспечивает соответствие соглашения об уровне внутренней поддержки службы ИС (*Operation Level* Agreement – OLA) и субординированных контрактов ИС-службы с поставщиками оборудования, ПО и услуг;
- осуществляет постоянное улучшение процесса.

Диаграмма активности процесса управления уровнем сервиса приведена на рисунке 3.5.

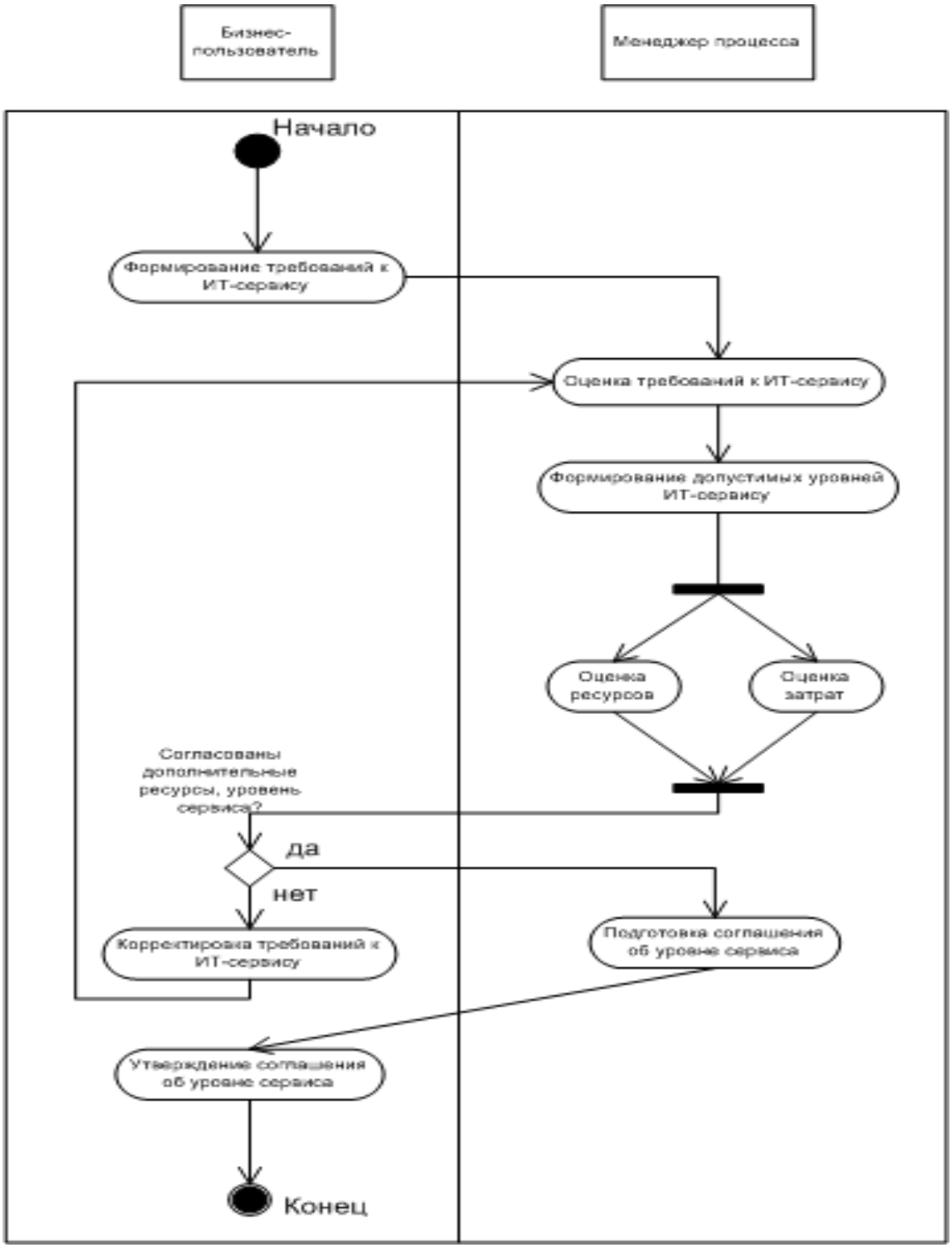


Рисунок 3.5 – Диаграмма активности процесса управления уровнем сервиса

Бизнес-*пользователь* формулирует требования к ИТ-услуге (установить поддержку электронной почты в режиме 24 х 7). *Менеджер* процесса управления уровнем сервиса совместно с менеджером процесса управления мощностями уточняет данные о

дополнительной потребности в сотрудниках службы сопровождения. В рамках процесса управления затратами уточняется *смета* дополнительных расходов на такой сервис. Соответствующие данные передаются на рассмотрение бизнес-пользователей, при их согласии на выделение дополнительных ресурсов новый уровень сервиса и новые ресурсы фиксируются в соглашении об уровне сервиса.

Если бизнес-*пользователь* не согласовывает требуемые ресурсы и *затраты* на ИТ-сервис, то необходимо провести пересмотр требований к ИТ-сервису.

Процесс управления мощностями (*Capacity Management – CAP*) предназначен для оптимизации использования ресурсов ИТ-инфраструктуры в соответствии с требованиями бизнеса к уровню обслуживания и тенденциями развития инфраструктуры. Четкое *определение* параметров предоставления услуг и их связи с элементами инфраструктуры, формализованные требования к готовности и бесперебойности предоставления услуг, прогнозирование развития в рамках управления мощностями – все это создает основу для корректного определения стоимости предоставления каждой услуги.

Основная задача этого процесса – обеспечение устойчивой работы ИТ-сервиса с требуемым уровнем производительности при максимально возможных объемах обрабатываемых данных, оговоренных в *SLA*, как в текущий момент, так и будущем.

Процесс управления мощностями должен обеспечивать оптимизацию расходов, времени приобретения и размещения ИТ-ресурсов с целью обеспечения выполнения условий *SLA*. Данный процесс предполагает *управление ресурсами*, производительностью, спросом на ИТ-*моделирование*, планирование мощностей, управление нагрузкой и *определение* необходимого объема технических средств для работы приложений.

Процесс управления мощностями выполняет следующие функции:

- инвентаризует ИТ-ресурсы;
- картографирует загрузку ИТ-сервисов и требования к ней, фиксирует результаты;
- ведет анализ проблем;
- дает рекомендации в отношении аутсорсинга (в области пропускной способности);

- анализирует производительность в условиях реальной загрузки;
- определяет систему планирования пропускной способности и измерения последней;
- осуществляет постоянное улучшение процесса.

Реализация процесса управления мощностями позволяет планировать использование ресурсов и ввод в эксплуатацию оптимальным способом благодаря следующим факторам:

- рациональное управление использованием ИТ-ресурсов и технологий с целью уменьшения стоимости предоставления ИТ-услуг и снижения рисков отказов;
- структурирование процесса ввода в эксплуатацию и перераспределения ИТ-ресурсов в соответствии с потребностями бизнеса;
- анализ зависимости требований к количеству и производительности ИТ-ресурсов от специфики и вариативности бизнес-цикла;
- повышение окупаемости инвестиций за счет оптимизации использования ИТ-ресурсов, своевременного согласования требований к производительности и возможностей ИТ-ресурсов, сокращения капитальных расходов на оборудование, повышения готовности систем и увеличения производительности конечных пользователей.

Процесс управление мощностями позволяет анализировать и прогнозировать развитие ИТ-инфраструктуры предприятия за счет следующего:

- формирования в *централизованном хранилище данных* о производительности ИТ-ресурсов для *анализа тенденций*, изменений потребностей и планирования инвестиций в ИТ-инфраструктуру;
- согласования достижимого качества предоставления ИТ-услуг с учетом возможностей ИТ-ресурсов;
- моделирования и планирования сценариев оптимизации ИТ-инфраструктуры для определения требований к производительности ИТ-ресурсов при изменениях и развитии бизнеса;
- централизации и автоматизации динамического перераспределения ИТ-мощностей;
- устранения избытка или нехватки ИТ-ресурсов;
- оценки возможностей виртуализации ИТ-ресурсов;

- динамического перераспределения аппаратных и программных ресурсов на основе оперативных или прогнозируемых потребностей в производительности ИТ-ресурсов для обеспечения необходимого уровня бизнес-услуг.

Процесс управления доступностью (*Availability Management* – AVM) контролирует способность службы ИС обеспечить экономически эффективный и устойчивый уровень доступности ИТ-сервисов, удовлетворяющий требованиям бизнеса.

Цель процесса управления доступностью состоит в том, чтобы оптимизировать способность ИТ-инфраструктуры, ИТ-сервисов и организаций внешних поставщиков поставлять оптимальный *по* стоимости уровень доступности, который позволит бизнесу удовлетворить свои бизнес цели. Эта цель достигается путём определения требований бизнеса *по* доступности и соответствия этих требований способностям ИТ-инфраструктуры и организаций внешних поставщиков услуг.

Под доступностью понимается способность ИТ-сервиса исполнять требуемую функцию в установленный момент или за установленный период времени. Доступность подкреплена надежностью и восстанавливаемостью ИТ-инфраструктуры и эффективностью работы организаций внешних поставщиков. Надежность ИТ-сервиса может быть точно определена как независимость от оперативного сбоя. Восстанавливаемость касается способности компонента ИТ-инфраструктуры содержаться или возвращаться к операционному состоянию.

Основная задача данного процесса – *определение требований* бизнеса к доступности и реализация этих требований в инфраструктуре ИТ и организации сопровождения. В тех случаях, когда требования бизнеса превышают возможности службы ИС, управление доступностью обеспечивает предоставление бизнесу возможных альтернатив и связанных с ними затрат.

Процесс управления доступностью осуществляет следующие функции:

- инвентаризация ресурсов ИТ;
- определение узких мест ИТ-сервисов с точки зрения доступности;
- анализ проблем;
- выработка рекомендаций в отношение аутсорсинга;

- анализ доступности ИТ-сервисов, в том числе при отказе оборудования, ПО, каналов связи и т.д.;
- регистрация проблем доступности, угрожающие невыполнением *SLA* и подготовка рекомендаций по их устранению;
- формирование системы планирования доступности и измерения последней;
- осуществление постоянного улучшения процесса.

Опишем возможный вариант *диаграммы активности* процесса управления доступностью. На уровне процесса управления проблемами обнаружена известная ошибка. В рамках процесса управления доступностью сотрудник ИС-службы анализирует влияние компонентов ИТ-инфраструктуры на доступность различных сервисов и риск невыполнения *SLA по* этим сервисам при возникновении ошибки. На основе анализа подготавливаются предложения *по* изменениям ИТ-инфраструктуры. Если предложения принимаются, то подготавливается *график* проведения изменений.

Процесс управления непрерывностью предоставления ИТ-сервисов (IT *Service* Continuity *Management* – ITSCM) обеспечивает выполнение требований к устойчивости предоставляемых сервисов, в первую *очередь* необходимых для функционирования критичных бизнес-процессов.

Под устойчивостью понимается способность ИС-службы и ИТ-инфраструктуры организации поддерживать сервисы в работоспособном состоянии в случае чрезвычайных ситуаций – пожара, наводнения, других стихийных бедствий и техногенных катастроф. В *SLA* должны быть зафиксированы требования к предоставлению сервисов в чрезвычайных ситуациях и ресурсам для их обеспечения. Соответствующие данные должны быть предоставлены процессом управления уровнем сервиса.

Цель процесса управления непрерывностью предоставления ИТ-услуг – *поддержка* непрерывности бизнеса в целом. Такая *поддержка* означает, что, во-первых, *инфраструктура* и ИТ-услуги, в том числе услуги *по* поддержке (служба *Service* Desk), должны быть восстановлены за заданный период времени после возникновения чрезвычайной ситуации. Во-вторых, на время восстановления предоставление ИТ-услуг должно поддерживаться на «аварийном» уровне, приемлемом для ведения бизнеса, то есть на уровне, минимально необходимом для функционирования бизнеса. Поскольку целью процесса является *поддержка* бизнеса, то сфера действия

процесса должна определяться в первую *очередь* исходя из целей бизнеса.

Согласно *ITIL* процесс отвечает за решение следующих основных задач [10]:

- оценка воздействия нарушений в предоставлении ИТ-услуг при возникновении чрезвычайной ситуации;
- определение критичных для бизнеса ИТ-услуг, которые требуют дополнительных превентивных мер по обеспечению непрерывности их предоставления;
- определение периода, в течение которого предоставление ИТ-услуги должно быть восстановлено;
- определение общего подхода к восстановлению ИТ-услуги;
- разработку, тестирование и поддержку плана восстановления ИТ-услуги с достаточным уровнем детализации, который поможет пережить чрезвычайную ситуацию и восстановить нормальную работу за заданный промежуток времени.

Процесс управления финансами ИТ-службы (Financial *Management*) отслеживает фактические *затраты* в разрезе заказчиков, ИТ-сервисов и пользователей и на этой основе рассчитывает внутренние цены на услуги ИС-службы. Процесс взаимодействует с процессом управления уровнем сервиса для определения цен сервисов.

Основная цель процесса состоит в следующем:

- сформировать информацию о полных стоимостях предоставляемых ИТ-сервисов, с целью повышения производительности и эффективности работы ИТ-службы;
- упорядочить поведение клиентов, предоставляя им информацию о действительной стоимости ИТ-сервисов;
- обеспечить возврат затрат на предоставление ИТ-сервисов.

Основная задача процесса управления затратами – расчет издержек, связанных с ИТ-сервисами, цен сервисов для бизнес-пользователей и *поиск* путей снижения затрат.

Функциями данного процесса являются:

- прогноз затрат и выручки (последняя определяется на основании внутренних цен на услуги);
- разработка бюджета сервисов;
- анализ использования сервисов и связанных с этим издержек, поиск путей их снижения;

- калькулирование счета и выставление его бизнес-пользователям, получение платежей;
- расчет совокупной стоимости владения (ССВ) ИТ-сервисов;
- установление системы ценообразования и выставление счетов за услуги;
- установление системы управления затратами;
- установление механизма привлечения инвестиций;
- осуществление постоянного улучшения процесса.

Процесс управления финансами касается экономических вопросов предоставляемых ИТ-услуг. Например, данный процесс подготавливает информацию о расходах, возникших при предоставлении услуг. В результате при определении необходимых изменений ИТ-инфраструктуры возможен учет финансовых факторов (соотнесение расходов и доходов – цены и результата). Эта *деятельность* повышает информированность о расходах (где возникают издержки и какие) и может использоваться также при составлении бюджета. Управление финансами ИТ-службы описывает различные методы выставления счетов, включая *определение* цели выставления счетов за ИТ-услуги и *определение* ценообразования, а также аспекты бюджетирования.

Процесс управления безопасностью (*Security Management*) обеспечивает внедрение, *контроль* и техническую поддержку инфраструктуры безопасности, а также разработку и *контроль* соблюдения стандартов безопасности существующих, разрабатываемых и планируемых ИТ-сервисов. В ряде случаев он рассматривается вне рамок процессов предоставления ИТ-сервисов

Основная задача процесса управления безопасностью – планирование и *мониторинг* безопасности ИТ-сервисов.

Функции процесса управления безопасностью таковы:

- разработка корпоративной политики безопасности в части ИС, обеспечение необходимого уровня безопасности в этой области;
- анализ проблем безопасности и рисков в этой области;
- *аудит безопасности* и оценка инцидентов в этой области;
- установление *процедур безопасности*, включая защиту от вирусов;
- выбор систем и инструментов поддержания безопасности;
- постоянное улучшение процесса.

Таким образом, блок процессов поддержки ИТ-сервисов обеспечивает разработку новых ИТ-сервисов при обеспечении

целостности и согласованности ИТ-инфраструктуры предприятия. ИТ-*инфраструктура* как целое оптимизируется *по* пропускной способности и затратам при заданном уровне производительности и устойчивости ИТ-сервисов. Вновь разработанные ИТ-сервисы передаются на одобрение в *процесс управления* изменениями и в случае одобрения предложений передаются в блок процессов разработки и внедрения сервисов.

В терминах функций ИС-службы блок процессов поддержки ИТ-сервисов является ядром выполнения функции планирования и организации *работ*, с одной стороны, и мониторинга – с другой. В функции планирования реализуются задачи планирования основного объекта управления – ИТ-сервисов. В функции координации *работ* процессы данного блока обеспечивают согласование потребностей бизнес-подразделений, возможностей информационных систем и стоимости сервиса для бизнес-*подразделения*. Результатом такого согласования становится спецификация ИТ-сервиса. В области мониторинга данные роли обеспечивают *контроль* процессов ИС-службы с точки зрения основных инженерных областей – безопасности, устойчивости и пропускной способности.

Вопросы для самоконтроля

1. Как характеризуется роль ИС-службы в современном бизнесе?
2. Чем модель *ITSM* отличается от традиционного *функционального подхода* к организации ИТ-службы?
3. Перечислите особенности проекта *ITIL*?
4. Какие разделы управления ИТ-сервисами описаны в текущей версии библиотеки *ITIL*?
5. Какие направления управления ИТ-услугами описаны в проекте *ITIL* Refresh?
6. Какие процессы включены в блок поддержки ИТ-сервисов?
7. Какие процессы включены в блок предоставления ИТ-сервисов?
8. Поясните назначение процесса управления инцидентами.
9. Поясните понятие «инцидент».
10. Приведите основные функции процесса управления инцидентами.
11. Поясните назначение процесса управления проблемами.
12. Поясните понятие «проблема».

4 Повышение эффективности ИТ-инфраструктуры предприятия на примере методологии Microsoft

4.1 Библиотека документов MOF

Библиотека передового опыта организации управления ИТ-инфраструктурой предприятия представляет общие рекомендации и различные организации вносят свой вклад в развитие этого направления. Microsoft на основе обобщения документации *ITIL*, стандарта *ISO* 15504, описывающего критерии оценки зрелости процессов, опыта заказчиков и партнеров Microsoft, опыта организации эксплуатации во внутренних ИТ-подразделениях Microsoft разработала библиотеку документов Microsoft Operations Framework (MOF) [11].

В состав MOF входят следующие документы и руководства:

- Модель процессов эксплуатации (MOF);
- Модель групп эксплуатации (MOF Team Model for Operations);
- Дисциплина управления рисками эксплуатации (*Risk Management Discipline* for Operations);
- функцииуправленияуслугами (*SMF* – Service Management Functions).

Модель процессов эксплуатации и *функции управления услугами* описывают высокоуровневые *операции*, выполняемые при эксплуатации информационных систем, и основываются на четырех принципах:

- структуризация;
- быстрый цикл развития, итеративный подход;
- управление посредством периодических *контрольных мероприятий*;
- интегрированное управление рисками.

Принцип *структуризации* упрощает *интеграцию процессов, управление жизненным циклом информационной системы* и сопоставление ролей с выполняемыми функциями.

Принцип *быстрого цикла развития* способствует повышению качества работы информационной системы предприятия посредством эффективного проведения изменений при оценке рисков.

Принцип *контрольных мероприятий* обеспечивает регулярную оценку оперативной деятельности *по* эксплуатации ИТ-

инфраструктуры и предоставлению ИТ-сервисов, а также результативности и эффективности действий *по* внесению изменений в информационную систему.

Принцип *интегрированного управления рисками* предполагает распространение процедур управления рисками во все операционные процессы и роли, а также формирование упреждающей политики управления рисками.

4.2 Модель процессов MOF

Модель процессов MOF сформирована из четырех категорий-квадрантов [11], в которых объединены ключевые задачи эксплуатации информационных систем (рисунок 4.1).

В модели выделены следующие квадранты: изменения; эксплуатация; поддержка и оптимизация.

Квадрант «Изменения» (MOF Changing Quadrant) предназначен для формализации и упорядочивания процессов изменения ИТ-инфраструктуры и ИТ-сервисов. В нем описаны следующие процессы: управление изменениями и управление релизами.

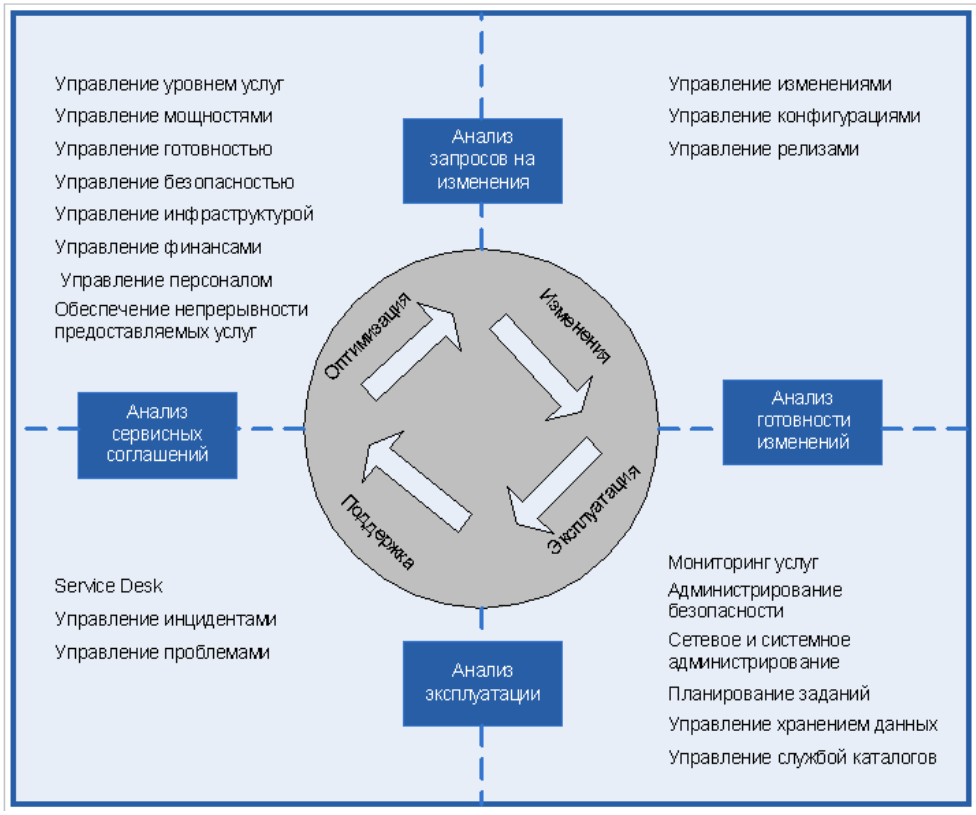


Рисунок 4.1– Модель процессов MOF

Функциональность квадранта «Изменения» в отличие от аналогичных процессов *ITIL* отличается более детальной проработкой диаграмм процессов и инструкций *по* их применению.

Квадрант «*Эксплуатация*» (MOF Operating Quadrant) описывает процессы технической инфраструктуры информационной системы (рисунок 4.2).

Для квадранта «*Эксплуатация* « выделены два уровня процессов. На верхнем уровне находятся следующие процессы:

- системное администрирование;
- администрирование безопасности;
- мониторинг ИТ-сервисов.

Данные процессы описывают принципы организации процессов эксплуатации технических и программных систем.

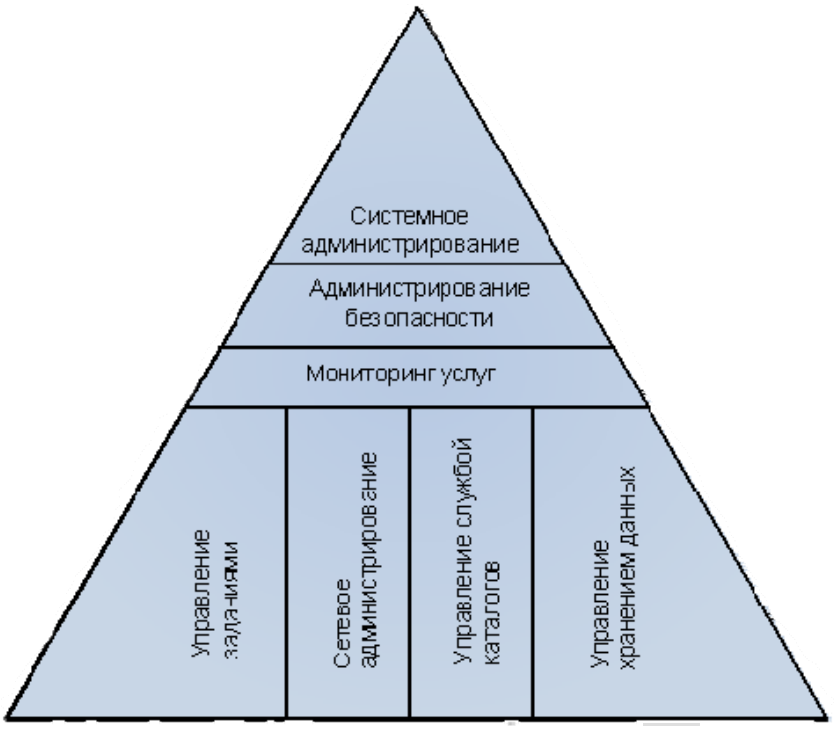


Рисунок 4.2 – Квадрант «Эксплуатация»

На втором уровне находятся следующие процессы:

- управление заданиями;
- сетевое администрирование;
- управление службой каталога;
- управление хранением данных.

Эти процессы описывают процессы эксплуатации конкретных подсистем.

Следует отметить, что процессы квадранта «*Эксплуатация*» ориентированы на использование продуктов Microsoft.

Квадрант «*Поддержка*» (MOF Supporting Quadrant) описывает процессы поддержки пользователей и ИС-службы. В нем описаны следующие процессы:

- Service Desk;
- управление инцидентами;
- управление проблемами.

Документация *по* процессам данного квадранта в целом соответствует содержанию аналогичных процессов *ITIL*, но в некоторых случаях детализируются диаграммы процессов и рекомендации *по* их применению.

Квадрант «*Оптимизация*» (MOF Optimizing Quadrant) описывает процессы предоставления ИТ-сервисов и оптимизации их предоставления. В данном квадранте описаны следующие процессы:

- управление уровнем предоставления ИТ-сервисов;
- финансовый ИТ-менеджмент;
- управление мощностями;
- управление готовностью;
- управление непрерывностью предоставления ИТ-сервисов;
- управление персоналом ИТ-подразделений;
- управление безопасностью;
- оптимизация ИТ-инфраструктуры.

Если первые пять процессов, в основном, соответствуют с небольшими дополнениями процессам *ITIL*, то процесс «Управление персоналом» базируется на опыте Microsoft *по* управлению персоналом, мотивации, обучения и удержания квалифицированных кадров. Содержание процессов «*Управление безопасностью*» и «*Оптимизация* ИТ-инфраструктуры» содержат описание передового опыта обеспечения безопасности и оптимизации ИТ-инфраструктуры.

Модель групп эксплуатации формализует и описывает распределение ролей между участниками процесса эксплуатации ИС и обеспечение взаимодействия с внешними и внутренними группами проектирования. В модели групп MOF описаны следующие роли:

- группа управления изменениями в ИТ-среде;
- группа управления физической инфраструктурой и инструментами управления инфраструктурой (операциями);

- группа поддержки;
- группа управления портфелем ИТ-сервисов;
- группа управления ИТ-инфраструктурой;
- группа безопасности;
- группа взаимодействия с поставщиками услуг и продуктов (партнеры).

Как правило, роли распределяют между подразделениями ИТ-службы предприятия, но иногда они назначаются бизнес-подразделениям, внешним консультантам и партнерам.

Для малых предприятий в рамках организационной структуры ИТ-службы возможны совмещения некоторых *ролей сотрудниками*. Рекомендации *по* совмещению ролей приведены в таблице 4.1. Ячейки таблицы помечены символами, имеющими следующий смысл: Д – допустимо *совмещение ролей*; Н/Д – не допустимо *совмещение ролей*; Н/Р – не рекомендуется совмещение ролей.

Таблица 4.1 – Возможности *совмещения ролей* участниками процесса эксплуатации ИС

	Безо-пас-ность	Управ-ление измене-ниями	Управление инфра-структурой	Под-держка	Парт-неры	Управ-ление опера-циями	Управ-ление ИТ-серви-сами
Безопасность		Н/Р	Д	Н/Д	Н/Р	Д	Н/Р
Управление изменениями	Н/Р		Д	Н/Д	Д	Д	Н/Р
Управление инфраструк-турой	Д	Д		Д	Д	Д	Н/Р
Поддержка	Н/Д	Н/Д	Д		Д	Д	Д
Партнеры	Н/Р	Д	Д	Д		Д	Н/Р
Управление операциями	Д	Д	Д	Д	Д		Д
Управление ИТ-сервисами	Н/Д	Н/Д	Н/Д	Д	Н/Р	Д	

Дисциплина управления рисками эксплуатации описывает процессы выявления риска и *принятия решений по* устранению риска. При этом риском считается возможность нарушения предоставления ИТ-сервиса, а *управление рисками* – это регулярная *деятельность*, обеспечивающая актуальность мер *по* минимизации выявленных

рисков или предупреждению в каждый момент выполнения операций *по* эксплуатации.

В дисциплине определены шесть этапов управления рисками.

На этапе «Выявление» идентифицируют существующие риски и фиксируют их как можно раньше.

Этап анализа и определения приоритетов определяют потенциальные угрозы от рисков и устанавливают приоритеты с целью выделения ограниченных ресурсов на снижение наиболее существенных рисков.

Этап «Планирование» предполагает разработку плана действий для снижения влияния рисков на эксплуатацию ИС и внесение изменений в другие процессы управления ИТ-инфраструктурой с целью снижения уровня рисков.

Этап «*Мониторинг* и отчетность» состоит в отслеживании статуса конкретных рисков, исполнении соответствующих им планов, подготовки отчетов для персонала и руководства о статусе наиболее опасных рисков и планов действий *по* управлению ими.

Этап управления рисками предполагает *исполнение* плана действий *по* конкретным рискам и формирование соответствующей отчетности.

На этапе «Обучение» осуществляется накопление и применение опыта управления рисками.

Вопросы для самоконтроля

1. Какие *уровни зрелости* ИТ-инфраструктуры предприятия предложены компанией Microsoft?
2. Как характеризуется базовый уровень зрелости ИТ-инфраструктуры в модели Microsoft?
3. Как характеризуется стандартизированный уровень зрелости ИТ-инфраструктуры в модели Microsoft?
4. Как характеризуется рационализированный уровень зрелости ИТ-инфраструктуры в модели Microsoft?
5. Как характеризуется динамический уровень зрелости ИТ-инфраструктуры в модели Microsoft?
6. Какие документы и руководства входят в состав библиотеки документов Microsoft Operations Framework (MOF)?
7. На каких принципах основывается модель процессов эксплуатации и функции управления услугами MOF?
8. Какие категории квадрантов входят в модель процессов MOF?

5 Основные методологии разработки ПО

Для успешного выполнения ИТ-проекта недостаточно выбрать эффективные технологии и средства разработки, обеспечить необходимый бюджет и найти квалифицированных разработчиков. В любой организации существуют правила и методики, по которым участники проекта (заказчики, аналитики, разработчики, тестеры, технические писатели) распределяют между собой задачи, взаимодействуют друг с другом, создают проектные артефакты (спецификации, исходный код, документацию). Эти правила могут быть четко организованными или хаотичными, быть формально документированными или существовать в головах проектной команды, но в любом случае именно их совокупность называется процессом разработки. Процесс – частный случай более общего понятия методологии разработки ПО. Примерами методологий являются структурное программирование или объектно-ориентированный анализ и дизайн. Далее ограничимся наиболее распространенными процессами разработки; методологии рассматриваются в тех случаях, когда они являются неотъемлемым атрибутом процесса.

5.1 Модель водопада

Модель водопада (waterfall model или последовательная разработка) – наверное, самый известный, исторически появившийся одним из первых процесс разработки. Он был описан в статье Ройса (W.W.Royce) в 1970 году (рисунок 5.1). Основная идея заключается в том, что процесс разработки делится на четко определенные фазы, выполняемые строго последовательно. Название «водопад» появилось из-за внешнего вида диаграммы, изображающей процесс. Отдельные задачи, из которых состоит любой проект (не обязательно выполняемый по модели водопада), можно разделить между несколькими процессными областями.

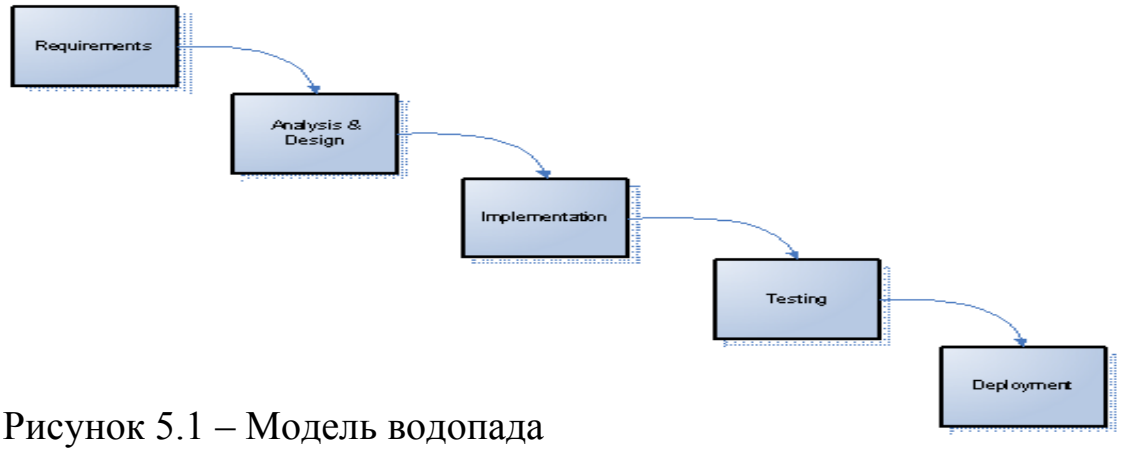


Рисунок 5.1 – Модель водопада

Классическая водопадная модель включает следующие области:

• разработка требований (requirements): сбор бизнес-требований заказчика и их преобразование в функциональные требования к программному продукту;

• анализ и дизайн (analysis and design): разработка модели предметной области (domain model), проектирование схемы базы данных, объектной модели, пользовательского интерфейса и т.п.;

• реализация (implementation): создание продукта по спецификациям, разработанным на предыдущем этапе;

• тестирование (testing): включает проверку соответствия функциональности программного продукта потребностям пользователей (validation), а также поиск дефектов в реализации;

• развертывание (deployment): обучение пользователей, инсталляция системы, перевод в промышленную эксплуатацию.

В модели водопада каждая из процессных областей представляет собой отдельную фазу проекта. Фазы выполняются строго последовательно, т.е. анализ и дизайн начинаются после завершения разработки требований, началу реализации предшествует завершение дизайна и т.д.

Основные соображения для использования такой модели разработки вполне очевидны. Как известно, стоимость исправления ошибок, сделанных в ходе выполнения проекта, зависит от того, насколько быстро эти ошибки обнаруживаются и исправляются. Ошибку в требованиях достаточно просто исправить на этапе разработки требований, но если о ней становится известно после завершения развертывания, последствия могут быть катастрофическими. Модель водопада стремится уменьшить, насколько это возможно, число таких долгоживущих ошибок. Для

этого разработка дизайна не должна начинаться, пока требования не будут определены с достаточным качеством, кодирование не начинается до появления полного дизайна системы и т.д.

Насколько эффективным оказался такой подход? Он хорошо работает в проектах, где требования могут быть четко определены и зафиксированы. В таких проектах модель водопада позволяет обеспечить заданный уровень качества (который может быть весьма высоким) и соблюдать бюджетные и временные ограничения. Благодаря этому она часто используется в больших организациях (таких как Министерство обороны США и NASA) при строгих требованиях к надежности создаваемого ПО.

Однако практика показала следующие недостатки этого процесса:

- процесс плохо работает в проектах с нечеткими требованиями. Даже если проектная команда считает, что полностью проработала и документировала функциональный дизайн системы, он может значительно отличаться от ожиданий пользователей. С большой вероятностью это расхождение будет обнаружено не на этапе рецензирования функциональной спецификации (редкий заказчик способен представить поведение реальной системы, читая документ с описанием ее функциональности), а во время внедрения продукта;
- процесс крайне неэффективен при постоянных изменениях требований (что как правило случается в проектах, длящихся больше одного-двух месяцев). Каждое изменение заставляет возвращаться к фазе определения требований и повторять весь процесс с начала;
- сложно управлять рисками некоторых типов (таких, как риски, связанные с использованием новых технологий или риски некорректного определения требований). Подобные риски могут проявить себя только на этапе реализации (если не тестирования), когда число возможных путей исправления ситуации намного меньше, чем в начале проекта;
- весьма ограничены возможности оценки и корректировки важных атрибутов проекта – скорости разработки, качества продукта, обоснованности принятых архитектурных решений. Адекватно оценить эти атрибуты становится возможным только на поздних этапах проекта.

Модель водопада является разумным выбором для типовых, стандартных проектов или при наличии жестких требований к

качеству (например, при создании mission critical-систем). Тем не менее, ее недостатки весьма существенны, и для разработки коммерческого ПО, как правило, существуют значительно более эффективные альтернативы.

5.2 Итеративная разработка

Процесс итеративной (или инкрементальной) разработки стал эволюционным развитием модели водопада. Процесс состоит из серии повторяющихся итераций (их число зависит от конкретного проекта), каждая из которых фактически является полноценным мини-проектом с фазами определения требований, анализа, дизайна и т.д. В результате очередной итерации продукт приобретает новую функциональность или улучшения в существующей функциональности. Полный набор требований, зафиксированный границами проекта, оказывается реализованным после завершения финальной итерации.

Основываясь на специфике проекта и требованиях заказчика, разработчики могут выбирать, что они хотят получить в результате очередной итерации:

- полноценную систему с ограниченной функциональностью, готовую для промышленной эксплуатации;
- функциональные и архитектурные прототипы, непригодные для промышленной эксплуатации, но позволяющие оценить функциональный дизайн, пользовательский интерфейс, производительность и т.д.

Итеративная разработка обладает рядом преимуществ по сравнению с последовательной моделью:

- реализация наиболее важных функций может быть завершена в ходе нескольких первых итераций. После их завершения (то есть намного раньше окончания всего проекта) заказчик сможет начать использование системы;
- уже в начале проекта пользователи получают возможность оценить функциональность системы и ее соответствие своим потребностям. Необходимые изменения и дополнения могут быть сделаны в течение следующих итераций;
- основные проектные риски могут (и должны) быть разрешены на первых итерациях. Например, архитектурное решение, приводящее

к неприемлемой производительности может быть обнаружено и исправлено уже в первой итерации.

Важно понимать, что все эти преимущества проявляются только при тщательном планировании итераций, в противном случае легко получить ухудшенный вариант модели водопада.

Итеративная модель используется во многих процессах разработки, включая RUP и гибкие методологии, описанные.

5.3 Методология RUP

Один из самых известных процессов, использующих итеративную модель разработки – Rational Unified Process (RUP). Он был создан во второй половине 1990-х годов в компании Rational Software. Основными разработчиками были Филипп Крачтен (Philippe Kruchten), Грейди Буч (Grady Booch), Джеймс Рамбо (James Rumbaugh) и Айвар Якобсон (Ivar Jacobson). Кстати, последние трое являются также создателями нотации UML.

Термин RUP означает как методологию разработки, так и продукт компании IBM (ранее – Rational) для управления процессами разработки. Методология RUP описывает абстрактный общий процесс, на основе которого организация или проектная команда должна создать специализированный процесс, ориентированный на ее потребности.

Можно выделить следующие основные характеристики процесса RUP.

Разработка требований. Для описания требований в RUP используются прецеденты использования (use cases). Это не слишком удивительно, учитывая, что один из создателей RUP, Айвар Якобсон, является также автором концепции прецедента использования. Полный набор прецедентов использования системы вместе с логическими отношениями между ними (прецеденты могут включать и расширять другие прецеденты) называется моделью прецедентов использования.

Каждый прецедент использования – это описание сценария взаимодействия пользователя с системой, полностью выполняющего конкретную пользовательскую задачу. Разумеется, не имеет смысла документировать в виде прецедентов нефункциональные требования (к производительности, качеству т.д.). Однако согласно RUP все функциональные требования должны быть представлены в виде прецедентов использования. Считается, что модель прецедентов дает более целостное представление о функциональности системы по

сравнению с традиционным описанием требований (перечислением функций, которыми должна обладать система).

Итеративная разработка. Проект в RUP состоит из последовательности итераций с рекомендованной продолжительностью от двух до шести недель. Основной единицей планирования итераций является прецедент использования. Перед началом очередной итерации определяется набор прецедентов использования, которые будут реализованы к ее завершению.

Итеративная модель, подробно описанная выше, позволяет вносить необходимые изменения в требования, проектные решения и реализацию в ходе проекта.

Архитектура. Можно сказать что RUP – ориентированная на архитектуру методология. Считается, что реализация и тестирование архитектуры системы должны начинаться на самых ранних стадиях проекта. RUP использует понятие исполняемой архитектуры (executable architecture) – основы приложения, позволяющей реализовать архитектурно значимые прецеденты использования. Основы исполняемой архитектуры должны быть реализованы как можно раньше. Это позволяет оценить адекватность принятых архитектурных решений и внести необходимые коррективы еще в начале проекта. Таким образом, для первых нескольких итераций необходимо выбирать прецеденты, которые требуют реализации большей части архитектурных компонентов.

RUP поощряет использование визуальных средств для анализа и проектирования. Как правило, используется нотация и, соответственно, средства моделирования UML (такие как Rational Rose). Модель предметной области документируется в виде диаграммы классов, модель прецедентов использования – при помощи диаграммы прецедентов, взаимодействие компонентов системы между собой описывается диаграммой последовательности и т д.

Жизненный цикл проекта RUP состоит из четырех фаз. Последовательность этих фаз фиксирована, но число итераций, необходимых для завершения каждой фазы, определяется индивидуально для каждого конкретного проекта. Фазы RUP нельзя отождествлять с фазами водопадной модели– их назначение и содержание принципиально различны.

Начало (Inception). Фаза Начало обычно состоит из одной итерации. В ходе выполнения этой фазы необходимо:

- определить видение и границы проекта;
- создать экономическое обоснование (business case);
- идентифицировать большую часть прецедентов использования и подробно описать несколько ключевых прецедентов;
- найти хотя бы одно возможное архитектурное решение;
- оценить бюджет, график и риски проекта.

Если после завершения первой итерации заинтересованные лица приходят к выводу о целесообразности выполнения проекта, проект переходит в следующую фазу. В противном случае проект может быть отменен или проведена еще одна итерация фазы Начало.

Проектирование (Elaboration). В результате выполнения этой фазы на основе требований и рисков проекта создается основа архитектуры системы. Проектирование может занимать до двух-трех итераций или быть полностью пропущенным (если в проекте используется архитектура существующей системы без изменений). Целями этой фазы являются:

- детальное описание большей части прецедентов использования;
- создание оттестированной (при помощи архитектурно значимых прецедентов использования) базовой архитектуры;
- снижение основных рисков и уточнение бюджета и графика проекта.

В отличие от модели водопада, основным результатом этой фазы является не множество документов со спецификациями, а действующая система с 20-30% реализованных прецедентов использования.

Построение (Construction). В этой фазе (длящейся от двух до четырех итераций) происходит разработка окончательного продукта. Во время ее выполнения создается основная часть исходного кода системы и выпускаются промежуточные демонстрационные прототипы.

Внедрение (Transition). Целями фазы Внедрения являются проведение бета-тестирования и тренингов пользователей, исправление обнаруженных дефектов, развертывание системы на рабочей площадке, при необходимости – миграция данных. Кроме того, на этой фазе выполняются задачи, необходимые для проведения маркетинга и продаж.

Фаза внедрения занимает от одной до трех итераций. После ее завершения проводится анализ результатов выполнения всего проекта: что можно изменить для улучшения эффективности в будущих проектах?

Рабочий процесс. В терминах RUP участники проектной команды создают так называемые артефакты (work products), выполняя задачи (tasks) в рамках определенных ролей (roles). Артефактами являются спецификации, модели, исходный код и т.п. Задачи разделяются по девяти процессным областям, называемым дисциплинами (discipline). В RUP определены шесть инженерных и три вспомогательные дисциплины. В них входят:

- бизнес-моделирование (Business Modeling) – исследование и описание существующих бизнес-процессов заказчика, а также поиск их возможных улучшений;
- управление требованиями (Requirements Management) – определение границ проекта, разработка функционального дизайна будущей системы и его согласование с заказчиком;
- анализ и проектирование (Analysis and Design) – проектирование архитектуры системы на основе функциональных требований и ее развитие на протяжении всего проекта;
- реализация (Implementation) – разработка, юнит-тестирование и интеграция компонентов системы;
- тестирование (Test) – поиск и отслеживание дефектов в системе, проверка корректности реализации требований;
- развертывание (Deployment) – создание дистрибутива, установка системы, обучение пользователей;
- управление конфигурациями и изменениями (Configuration and Change Management) – управление версиями исходного кода и документации, процесс обработки запросов на изменение (change requests);
- управление проектом (Project Management) – создание проектной команды, планирование фаз и итераций, управление бюджетом и рисками;
- среда (Environment) – создание инфраструктуры для выполнения проекта, включая организацию и настройку процесса разработки.

Общая схема методологии RUP представлена на рисунке 5.2.

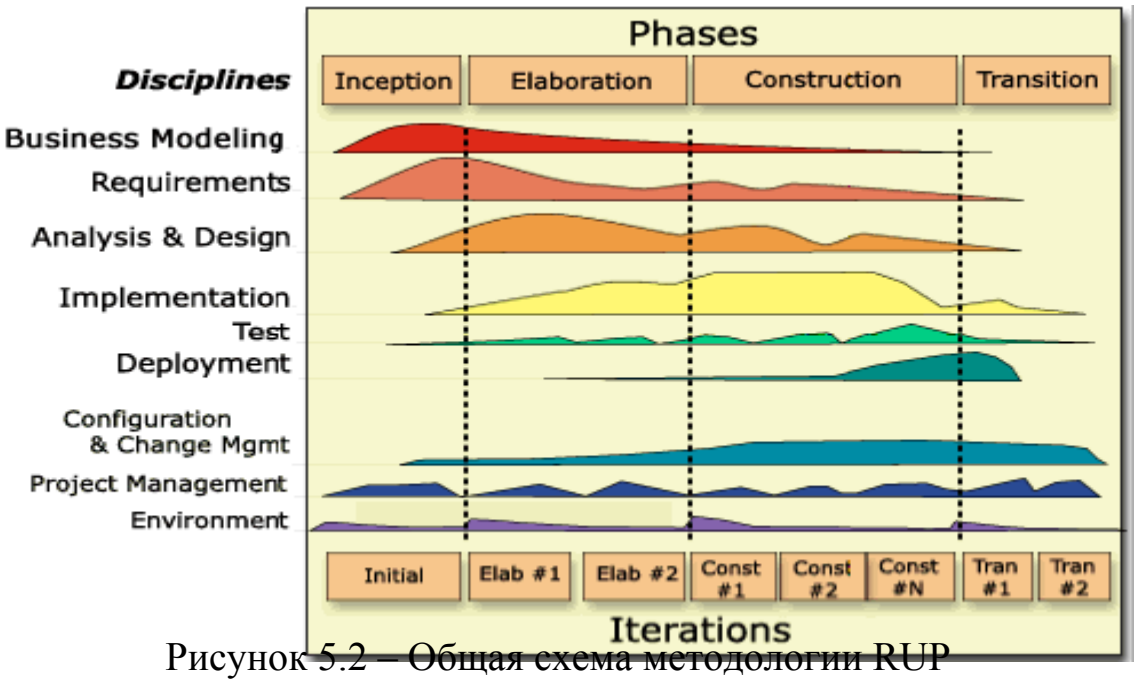


Рисунок 5.2 – Общая схема методологии RUP

В ходе жизненного цикла проекта распределение усилий проектной команды между дисциплинами постоянно меняется. Например, как правило, в начале проекта большая часть усилий затрачивается на анализ и дизайн, а ближе к завершению – на реализацию и тестирование системы. Однако в общем случае задачи из всех девяти дисциплин выполняются параллельно. Рисунок 4.2 иллюстрирует выполнение проекта среднего размера.

Практика. RUP считают тяжеловесным процессом с высоким уровнем формализма. Это не совсем так, поскольку процесс RUP может (и должен) быть настроен под специфику конкретной организации и проекта. Небольшие проекты требуют низкой степени формализма, крупные проекты с географически распределенной командой – высокой. С практической точки зрения, надо дать ответы на вопросы наподобие следующих:

- какие артефакты будет использоваться проектной командой? Например, будет ли создана формальная спецификация приемочного тестирования?
- каким образом артефакты будут документированы? Одна и та же модель может существовать в уме разработчиков, быть нарисованной на бумаге или созданной в одном из средств UML-моделирования;

▪ насколько формальным будет процесс создания артефактов? Например, для каких документов обязательно проводить рецензирование и кто будет это делать?

Однако широкие возможности настройки не делают внедрение RUP простым. Эффективному использованию RUP препятствует множество сложностей. Часто последовательность фаз начало-проектирование-построение-внедрение ошибочно интерпретируют как фазы анализ-дизайн-реализация-внедрение из водопадной модели – это практически сводит на нет преимущества RUP. Для большинства людей оказывается непривычной концепция документирования требований в виде прецедентов использования – трудности возникают как с написанием, так и с пониманием прецедентов. В результате спецификация требований, созданная проектной командой, может оказаться совершенно невнятным документом, который никто не будет читать.

Для полноценного внедрения RUP организация должна затратить значительные средства на обучение сотрудников. При этом попытка обойтись своими силами скорее всего будет обречена на неудачу – необходимо искать специалиста по процессам (process engineer) с соответствующим опытом или привлекать консультантов.

5.4 Гибкие методологии

В течение 1990-х годов все больше разработчиков ПО начинали искать альтернативу традиционным, как правило, основанным на модели водопада, процессам разработки. К 2000 году существовало уже целое множество так называемых легковесных (lightweight) методологий. (Я использую термин «методология», а не «процесс», поскольку гибкие методологии включают в себя множество практик и технологий, выходящих за рамки описания процессов.)

В 2001 году группа создателей и экспертов по различным легковесным методологиям провела семинар, на котором были сформулированы основные принципы гибкой разработки ПО (так называемый Agile Manifesto). На том же семинаре было предложено новое название легковесных методологий – гибкая разработка (agile software development).

Общими особенностями гибких методологий являются:

▪ ориентированность на людей – как разработчиков, так и заказчиков. Считается, что умение собрать в проектной команде «правильных» людей определяет успех или неудачу проекта в значительно большей степени, чем любые процессы или технологии;

- использование устных обсуждений вместо формальных спецификаций везде, где это возможно. Обсуждения должны быть главным способом коммуникации как с заказчиком, так и внутри проектной команды;
- итеративная разработка с возможно более короткой (в разумных пределах) продолжительностью итерации, при этом в результате каждой итерации выпускается полноценная работающая версия продукта;
- ожидание изменений – в гибком процессе проектная команда не пытается зафиксировать требования в начале проекта и затем следовать жестко определенному плану. Изменения могут быть сделаны на сколь угодно позднем этапе проекта.

По всей видимости, из методологий гибкой разработки самое широкое распространение получило экстремальное программирование (eXtreme Programming, XP), поэтому именно его мы рассмотрим подробнее.

5.4.1 Методология XP. Методология XP была создана Кентом Беком (Kent Beck) в 1996 году в ходе попытки спасти провальный проект по разработке системы расчета зарплаты для компании Крайслер. В 2000 году проект был закрыт, но XP к тому времени уже получила известность и начала распространяться среди разработчиков ПО.

XP наследует все общие принципы гибких методологий, достигая их при помощи двенадцати инженерных практик. Ниже описаны самые интересные из специфических технологий и практик XP.

1. В проектной команде должен постоянно работать так называемый представитель заказчика – он обладает детальной информацией о необходимой функциональности, определяет приоритеты отдельных требований, оценивает качество создаваемой системы. Технически, представитель заказчика может быть и сотрудником фирмы разработчика – менеджером продукта, бизнес-аналитиком и т.п.

2. Пользовательские истории – короткие неформальные описания прецедентов использования системы. В XP истории являются основным и, вместе с приемочными тестами, единственным средством спецификации требований. Поскольку истории очень лаконичны, участникам проекта обычно требуются более детальная информация по функциональности системы – они получают ее непосредственно от представителя заказчика.

3. Разработка через тестирование (test driven development) – в XP становится особенно важным, чтобы весь создаваемый код был покрыт автоматическими юнит-тестами (почему это так, станет понятно дальше). Этого можно добиться при помощи простого правила – новый код может быть написан исключительно для того, чтобы увеличить число успешно проходящих юнит-тестов. Фактически это означает, что перед реализацией новой функции разработчик должен создать соответствующий тест, а написание кода должно завершиться в тот момент, когда начнет проходить новый, а также все существующие тесты. Если после этого окажется, что новая функция реализована не полностью, необходимо создать еще один тест и повторить весь цикл заново.

4. Архитектура системы должна быть максимально простой. XP не рекомендует проектировать в расчете на будущее развитие системы; идеальная архитектура должна не более чем поддерживать существующую функциональность. Цель такого минималистского подхода к проектированию – избежать бесполезных инвестиций в архитектурные решения, которые часто оказываются выброшенными после очередного изменения требований. Вместо этого архитектура постоянно изменяется и развивается вместе с системой.

5. Постоянное изменение архитектуры требует постоянной переработки и улучшения кода – рефакторинга. В XP поощряется коллективное владение кодом – увидев возможность улучшения в любом компоненте системы, разработчик может провести необходимые рефакторинги вне зависимости от того, кто является основным разработчиком компонента. Возможные ошибки, внесенные рефакторингом, должны быть тут же обнаружены автоматическими тестами.

6. Все изменения, сделанные разработчиками, после автоматического тестирования практически сразу попадают в основной репозиторий. Таким образом, этап интеграции как таковой отсутствует или, что то же самое, происходит постоянно. XP называет эту практику непрерывной интеграцией.

7. Парное программирование – наверное, самая противоречивая практика XP. Использование парного программирования не означает, что на двух разработчиков в организации должен быть выделен только один компьютер, однако большая часть написания кода должна проходить в парах, подобно творчеству А. и Б. Стругацких. Считается, что при этом общая эффективность разработки

повышается за счет более продуманных решений, меньшего количества ошибок, тщательного написания юнит тестов и т.п.

8. Продолжительность рабочей недели не должна превышать 40 часов. По сравнению с обычной практикой постоянных переработок, в средне- и долгосрочной перспективе это повышает производительность проектной команды за счет уменьшения стресса и переутомления.

Жизненный цикл. Жизненный цикл проекта в XP состоит из последовательности релизов. Каждый релиз – это полноценная версия продукта, которую может использовать заказчик, и содержащая дополнительную функциональность по сравнению с предыдущим релизом. Релиз появляется в результате одной или нескольких итераций, длящихся от одной до четырех недель.

В XP не рекомендуется тратить много времени на планирование; сам процесс планирования называется игрой (planning game). Подробный план составляется только на очередную итерацию и ближайшие один-два релиза.

Планирование релиза состоит из следующих шагов:

- заказчик формулирует свои требования в виде историй, которые оцениваются по трудоемкости разработчиками. Оценки трудоемкости делаются в так называемых идеальных днях – времени, которое некий воображаемый разработчик потратит на реализацию истории при полном отсутствии отвлекающих факторов, параллельных задач, перерывов и т.п.;
- истории сортируются по приоритету, рискам, сложности реализации;
- определяется фактическая производительность команды (какое число реальных дней соответствует идеальному дню);
- на основании фактической производительности определяется, какие истории войдут в очередной релиз и за какое время он будет завершен.

Итерация планируется аналогичным образом:

- предварительно определяется набор историй для итерации;
- истории разбиваются на задачи (tasks), которые распределяются между разработчиками. В отличие от историй, которые описывают поведение системы с точки зрения пользователя, задачи составляются на техническом уровне, например «модифицировать схему базы данных» или «провести рефакторинг»;
- разработчики оценивают свои задачи. Казалось бы, это ненужное повторение операции оценки историй, однако на этом этапе уточненные оценки должны быть более реалистичными. Поскольку

задачи теперь оценивают их непосредственные исполнители, в оценках учитывается индивидуальная производительность, зависящая от опыта, знакомства с используемыми технологиями и т.п.;

- на основе уточненных оценок задач подсчитывается общая загрузка команды. В зависимости от загрузки некоторые истории могут быть перенесены на следующую итерацию или, наоборот, добавлены в текущую.

Периодически в ходе проекта измеряется фактическая производительность команды. Если она начинает сильно отличаться от значения, которое было использовано при планировании, график выхода релизов должен быть пересмотрен.

Практика. Во многом XP была создана как попытка описать процессы и практики, которые часто как бы сами собой появляются в эффективных, сплоченных командах разработчиков. Может показаться, что процесс в таких командах отсутствует, и, скорее всего, то же самое будут утверждать сами разработчики. Однако это не так, поскольку работа профессиональной команды всегда четко (хотя и неформально) организована и не имеет ничего общего с беспорядком и хаосом.

Это во многом определяет условия, необходимые для эффективного использования XP. Прежде всего, XP имеет шансы работать только в команде опытных, профессиональных разработчиков. Поскольку большую роль в XP играет прямое общение, команда не должна быть разбита на несколько частей – внедрение XP в распределенной географически команде будет крайне рискованным мероприятием. По той же причине возможный размер команды ограничен сверху – по всей видимости, числом в 10-15 человек.

Другие практики XP приносят свои ограничения. Далеко не всегда можно обеспечить постоянное присутствие представителя заказчика в проектной команде (например, если потенциальные пользователи системы делятся на несколько классов с частично конкурирующими требованиями).

Поскольку XP практически не делает попыток предотвратить размывание границ проекта (scope creep), будет не очень хорошей идеей использовать XP в проекте с фиксированной ценой. Фактически проекты XP обладают жестким графиком, но переменными границами, поэтому предпочтительным типом контракта будет повременная оплата (time&materials).

Практика поддержания максимально простой архитектуры может завести в тупик в конце проекта, когда окажется, что для реализации завершающих историй требуется полное перепроектирование системы (оказывается, нам нужно было предусмотреть возможность интеграции с системой SAP R/3, а также перевода на японский язык всего пользовательского интерфейса!). Даже если подобная ситуация не возникнет, нет никакой гарантии, что создаваемая ad hoc архитектура не будет намного более запутанной и сложной, чем было бы продуманное заранее решение.

Подобным образом может неконтролируемо откладываться разрешение некоторых нетехнологических рисков, наподобие неопределенных границ проекта.

Несмотря на все перечисленные ограничения, XP может замечательно работать в подходящих для него условиях. Благодаря крайне низким накладным расходам, в таких ситуациях этот процесс может показать исключительную эффективность.

XP является достаточно гибкой методологией. Не обязательно внедрять XP во всей компании, вполне разумно ограничиться теми командами и проектами, которые могут получить от этого реальный выигрыш. Например, для разработки ядра продукта можно использовать XP, а проекты по внедрению основывать на процессе RUP. Также не обязательно использовать все классические практики XP (на самом деле, мало кто это делает) – как правило, разумно ограничиться теми из них, которые сочетаются с корпоративной культурой и особенностями проектов.

5.4.2 Методолгия Scrum. Одна из самых востребованных методологий. В 1986 японские специалисты Hirotaka Takeuchi и Ikujiro Nonaka опубликовали сообщение о новом подходе к разработке новых сервисов и продуктов (не обязательно программных). Основу подхода составляла сплоченная работа небольшой универсальной команды, которая разрабатывает проект на всех фазах. Приводилась аналогия из регби, где вся команда двигается к воротам противника как единое целое, передавая (пасуя) мяч своим игрокам как вперед, так и назад. В начале 90-х годов данный подход стал применяться в программной индустрии и обрел название Scrum (термин из регби, означающий - схватка), в 1995 году Jeff Sutherland и Ken Schwaber представили описание этого подхода на OOPSLA '95 – одной из самых авторитетных конференций в области программирования. С тех пор метод активно используется в

индустрии и многократно описан в литературе. Scrum активно используется также в России.

Общее описание. Метод Scrum позволяет гибко разрабатывать проекты небольшими командами (7 человек плюс/минус 2) в ситуации изменяющихся требований. При этом процесс разработки итеративен и предоставляет большую свободу команде. Кроме того, метод очень прост – легко изучается и применяется на практике. Его схема изображена на рисунке 5.3.

Рисунок 5.3 – Схема метода Scrum

Вначале создаются требования ко всему продукту. Потом из них выбираются самые актуальные и создается план на следующую итерацию. В течение итерации планы не меняются (этим достигается относительная стабильность разработки), а сама итерация длится 2-4 недели. Она заканчивается созданием работоспособной версии продукта (рабочий продукт), которую можно предъявить заказчику, запустить и продемонстрировать, пусть и с минимальными функциональными возможностями. После этого результаты обсуждаются и требования к продукту корректируются. Это удобно делать, имея после каждой итерации продукт, который уже можно как-то использовать, показывать и обсуждать. Далее происходит планирование новой итерации и все повторяется.

Внутри итерации проектом полностью занимается команда. Она является плоской, никаких ролей Scrum не определяет. Синхронизация с менеджментом и заказчиком происходит после окончания итерации. Итерация может быть прервана лишь в особых случаях.

Роли. В Scrum есть всего три вида ролей.

Владелец продукта (Product Owner) — это менеджер проекта, который представляет в проекте интересы заказчика. В его обязанности входит разработка начальных требований к продукту (Product Backlog), своевременное их изменение, назначение

приоритетов, дат поставки и и прочее. Важно, что он совершенно не участвует в выполнении самой итерации.

Scrum-мастера (Scrum Master) обеспечивает максимальную работоспособность и продуктивную работу команды – как выполнение Scrum-процесса, так и решение хозяйственных и административных задач. В частности, его задачей является ограждение команды от всех воздействий извне во время итерации.

Scrum-команда (Scrum Team) – группа, состоящая из пяти–девяти самостоятельных, инициативных программистов. Первой задачей команды является постановка для итерации реально достижимых и приоритетных для проекта в целом задач (на основе Project Backlog и при активном участии владельца продукта и Scrum-мастера). Второй задачей является выполнение этой задачи во что бы то ни стало, в отведенные сроки и с заявленным качеством. Важно, что команда сама участвует в постановке задачи и сама же ее выполняет. Здесь сочетается свобода и ответственность, подобно тому, как это организовано в MSF. Здесь же «просвечивает» дисциплина обязательств.

Практики. В Scrum определены три практики (рисунок 5.4).

Sprint Planning Meeting. Проводится в начале каждого Sprint. Сначала Product Owner, Scrum-мастер, команда, а также представители заказчика и прочие заинтересованные лица определяют, какие требования из Project Backlog наиболее приоритетные и их следует реализовывать в рамках данного Sprinta. Формируется Sprint Backlog. Далее Scrum-мастер и Scrum-команда определяют то, как именно будут достигнуты определенные выше цели из Sprint Backlog. Для каждого элемента Sprint Backlog определяется список задач и оценивается их трудоемкость.

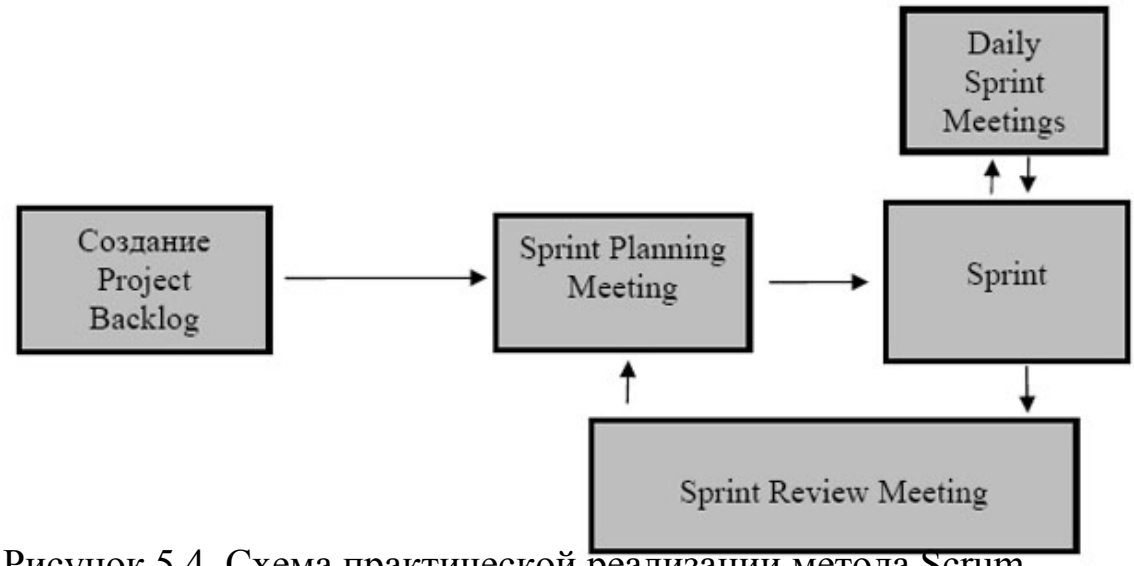


Рисунок 5.4 Схема практической реализации метода Scrum

Daily Scrum Meeting – пятнадцатиминутное каждодневное совещание, целью которого является достичь понимания того, что произошло со времени предыдущего совещания, скорректировать рабочий план сообразно реалиям сегодняшнего дня и обозначить пути решения существующих проблем. Каждый участник Scrum-команды отвечает на три вопроса: что я сделал со времени предыдущей встречи, мои проблемы, что я буду делать до следующей встречи? В этом совещании может принимать участие любое заинтересованное лицо, но только участники Scrum-команды имеют право принимать решения. Правило обосновано тем, что они давали обязательство реализовать цель итерации, и только это дает уверенность в том, что она будет достигнута. На них лежит ответственность за их собственные слова, и, если кто-то со стороны вмешивается и принимает решения за них, тем самым он снимает ответственность за результат с участников команды. Такие встречи поддерживают дисциплину обязательств в Scrum-команде, способствуют удержанию фокуса на целях итерации, помогают решать проблемы «в зародыше». Обычно такие совещания проводятся стоя, в течение 15-20 минут.

Sprint Review Meeting. Проводится в конце каждого Sprint. Сначала Scrum-команда демонстрирует Product Owner сделанную в течение Sprint работу, а тот в свою очередь ведет эту часть митинга и может пригласить к участию всех заинтересованных представителей заказчика. Product Owner определяет, какие требования из Sprint Backlog были выполнены, и обсуждает с командой и заказчиками, как лучше расставить приоритеты в Sprint Backlog для следующей итерации. Во второй части митинга производится анализ прошедшего спринта, который ведет Scrum-мастер. Scrum-команда анализирует в последнем Sprinte положительные и отрицательные моменты совместной работы, делает выводы и принимает важные для дальнейшей работы решения. Scrum-команда также ищет пути для увеличения эффективности дальнейшей работы. Затем цикл повторяется.

Методология Scrum имеет огромное количество самых разных ветвей, способов и методов, которые пополняются с каждым разом, исходя из опыта новых и старых компаний, которые использовали Scrum как основу управления. Где изучается Scrum (методология)? Тренинги, обучение имеют важное значение, так как позволяют познакомиться с основой управления и научиться ею пользоваться. Но чтобы максимально эффективно владеть ею, необходимо

постоянно практиковать свой уровень готовности. На данный момент Scrum (технологию) активно преподают люди, занимающиеся коллективным обучением и профессиональным продвижением. Найти и обучиться данной технологии можно в крупных городах и столицах различных стран Европы, Америки и Азии. В России эта методология только начинает получать свое развитие и уже имеет хорошие результаты в своем продвижении.

Scrum – методология управления проектом, которая может дать опыт контроля и правильного владения ситуацией в различных компаниях. Внедрять ее нужно только после тщательного анализа компании и персонала. Руководители неохотно идут на подобную практику и эксперименты, поэтому помимо знаний и практики вы должны уметь правильно предоставлять данные, уметь оказывать положительное влияние и просто доказывать необходимость использования методологии.

Заключение

В современных условиях сложность информационных систем предприятий возрастает, а методы и технологии эффективного управления их ИТ-инфраструктурой динамически развиваются. Это является следствием появления новых архитектурных подходов к построения информационных систем (сервис-ориентированные архитектуры – *SOA*, архитектуры, управляемые событиями – *EDA*), новых программных систем управления бизнес-процессами (business *process management suite*, BPMS), ориентированных на процессное управление бизнесом, новых программных платформ, инструментальных средств и приложений, а также новых, более жестких требований пользователей в отношении предоставляемых им информационных сервисов. В результате развития методов и технологий создаются новые программные решения для управления ИТ-инфраструктурой предприятия.

Все это определяет остроту вопроса подготовки ИТ-менеджеров – специалистов *по* реализации эффективного управления ИТ-инфраструктурой предприятия. Достойной целью каждого специалиста, который предполагает работать в области управления ИТ-инфраструктурой предприятия является получение сертификата *Service Manager*. Квалификация ИТ-менеджера определяется статусом IT *Service Manager*, который на сегодняшний день является наивысшей ступенью в табели о рангах специалистов в области управления ИТ-сервисами. Сертификат *ServiceManager* признается во всём мире как гарантия того, что его владелец обладает не только теоретическими знаниями в области управления услугами ИТ, но проявляет и менеджерские качества, позволяющие ему эффективно внедрять установки, изложенные в библиотеке *ITIL*.

Данное методическое пособие имело целью представить слушателям обзор существующих подходов к эффективному управлению ИТ- инфраструктурой предприятия, акцентировать их внимание на важной и перспективной сфере деятельности ИТ-менеджера и тем самым помочь сделать первый шаг в достижении статуса IT *Service Manager*.

Список использованных источников

1. Шафер Д., Фатрел Р., Шафер Л. Управление программными проектами: достижение оптимального качества при минимуме затрат. / М.: Вильямс, 2003. – 1136 с.
2. Хелдман К. Профессиональное управление проектом,/ М.: БИНОМ. Лаборатория знаний, 2005. – 517 с.
3. Ройс У. Управление проектами по созданий программного обеспечения. – М.: Издательство «Лори», 2000. – 431 с.
4. Брауде Э. Технологии разработки программного обеспечения. / СПб.: Питер, 2004. – 655 с.
5. Милошевич Д. Набор инструментов для управления проектами. / М.: Компания АиТи; ДМК Пресс, 2008. – 729 с.
6. Черняк Л. Библиотеки передового опыта и парадоксы управления ИТ.//Открытые системы №01, 2005. /[Электронный ресурс]. – 2005.– Режим доступа:http://www.osp.ru/os/2005/01/185188/– Дата доступа : 20.05.2016.
7. Долженко А. Управление информационными системами /А. Долженко [Электронный ресурс]. – 2016. – Режим доступа: http://www.intuit.ru/studies/courses/945/260/lecture/6638http/– Дата доступа : 20.07.2016.
8. Экономическая информатика: Введение в экономический анализ информационных систем. Учебник./ М.:ИНФРА-М, 2005. – 958 с.
9. Дубова Н. ITSM – новая идеология управления ИТ./Н. Дубова [Электронный ресурс]. 2016. – Режим доступа:– http://www.osp.ru/os/2000/10/037.htm/ Дата доступа : 25.03.2010.
10. Введение в ИТ сервис-менеджмент - принципы управления ИТ-услугами и сервисами /[Электронный ресурс]. – 2016. – Режим доступа:http://www.itexpert.ru/rus/biblio/articles/200406222006/200406222044/ Дата доступа : 15.03.2016.
11. MOF - Microsoft Operations Framework /[Электронный ресурс]. – 2016. – Режим доступа:http://www.itsmportal.ru/articles/it-control/2003-12-15%2000:00:00-26.html/ / Дата доступа : 25.03.2012.

Осипенко Александр Николаевич

МЕНЕДЖМЕНТ В ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЯХ

Пособие
по одноименному курсу
для слушателей специальности 1-40 01 73
«Программное обеспечение информационных систем»
заочной формы обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 13.07.17.
Рег. № 96Е.
http://www.gstu.by