Фазы жизненного цикла информационных систем

Управление жизненным циклом информационных систем

3.3. Проектирование

Проектирование

- Техническое проектирование
- Рабочее проектирование и прототипирование

Методы проектирования

Существует несколько основных методов проектирования ИС. Так, каноническое проектирование основано на пошаговом выполнении определенного порядка действий, при этом большинство процессов выполняются вручную или с минимальной автоматизацией. При этом проектирование зачастую осуществляется непосредственными исполнителями проекта.

Проектирование с использованием CASE-средств предполагает широкое применение программных инструментов, способных облегчить работу. Фокус внимания смещается на анализ и непосредственное проектирование, а инструменты позволяют автоматизировать создание документов, отчетности и генерацию кода.

Типовое проектирование применяется, когда речь идет об имеющихся типовых решениях, которые требуется адаптировать под потребности конкретной компании. В этом случае уникальный продукт не создается; напротив, происходит адаптация типового решения под реальные нужды бизнеса. Несмотря на кажущуюся простоту, типовое проектирование в сложных случаях немногим легче предыдущих методов.

В целом же проектирование ИТ-системы предполагает детальную подготовку к этапу настройки и развертывания модулей системы, для чего необходимо решить несколько задач:

- сформировать список модулей и функций системы, необходимых для поддержки определенных на этапе анализа автоматизируемых бизнес-процессов;
- сформировать список справочников систем (будущей и, если применимо, текущей) для дальнейшего переноса и обновления данных;
- определить примерный сценарий работы системы по категориям пользователей для формирования необходимого набора диалогов и процедур проектируемой системы (включая реакции на все возможные и даже очень маловероятные действия со стороны пользователя);
- определить элементы интерфейса пользователей (в случае гибкого или разрабатываемого «с нуля» решения) для достижения удобства работы с системой;

- сформировать список отчетов и панелей мониторинга (включая их формы и обязательные для реализации элементы); эти отчеты в дальнейшем будут использоваться как для учетных целей, так и для целей мониторинга системы ее администратором (в части формирования и сбора статистики по нагрузке на ее отдельные модули, свободным ресурсам, активности пользователей и т.п.);
- определить перечень настроек функциональных компонентов системы в соответствии с выделенными на предыдущем этапе требованиями;
- определить необходимость, возможности и пути интеграции с существующими и планируемыми к реализации системами на предприятии заказчика, чтобы своевременно предусмотреть технические средства для интеграции.

На этапе детального проектирования важно крайне четко определить все функциональные возможности системы и определить ее место в общей программной архитектуре предприятия.

Пример

Если предполагается, что система взаимодействия с клиентами CRM будет получать данные из личных кабинетов клиентов на портале компании, необходимо предусмотреть подобную интеграцию. Если в компании уже внедрена и работает ERP-система, CRM должна получать и предоставлять ей данные.

С организационной точки зрения к этому моменту важна подготовка иерархической структуры работ, базового календарного плана и дорожной карты, которые позволили бы управлять проектом на протяжении всей его реализации.

3.3.1. Техническое проектирование

Цель данного этапа — подготовка к этапу настройки модулей системы в целом. Данный этап предусматривает разработку организационной и функциональной структур модуля системы, определение уровня автоматизации информационного процесса предприятия, определение структуры информационного и технического обеспечения и требований к их элементам, постановку и алгоритмизацию задач обработки данных, разработку системы ведения нормативно-справочной базы.

Результаты работ оформляются в виде технического проекта (проектного решения на автоматизацию модуля системы) и представляют собой задание на программирование. Техническое проектирование, создание, настройка, доработка и внедрение модуля системы осуществляются только на основании соответствующих утвержденных технических заданий (проектных решений).

Технический проект (проектное решение по автоматизации модуля) — документ, содержащий описание функциональных компонентов системы, необходимых для реализации бизнес-процессов, описанных на этапе подготовки проектного решения, а также необходимых доработок системы и (при необходимости) процедур интеграции с модулями (или внешними системами), внедренными ранее.

Перечень документов, создаваемых на стадии «Технический проект», определяется документом ГОСТ 34.201—89. Основной задачей этой стадии является разработка архитектуры системы и технических решений по ее реализации. Перечень документов, служащих базой технического проекта, определяется условиями договора и технического задания, но чаще всего в него включаются следующие документы.

1. Пояснительная записка:

- общие положения, например, стадии и сроки, цели и задачи, используемые при разработке системы объекты ИС;
- описание процесса деятельности;
- основные технические решения, например, структура системы, основные функции, режимы функционирования, требования к персоналу;
- мероприятия по подготовке объекта автоматизации к вводу системы в действие, например, подготовка информации к импорту, создание рабочих мест.

2. Входные и выходные данные системы, например, импортируемые и вводимые вручную пользователями данные (названия и источники документов), формируемые системой документы.

3. Схема функциональной структуры:

- описание функций подсистем, например, описание функций по отдельным модулям (например, модуль расчета заработной платы, модуль учета труда);
- информационные связи между элементами и с внешней средой.

4. Описание автоматизируемых функций:

- исходные данные;
- цели АС и автоматизируемые функции;
- характеристика функциональной структуры, например, список подсистем, описание процесса реализации функций, требования (к надежности, защите информации от несанкционированного доступа, сохранности информации);
- типовые решения.

5. Постановка задач и алгоритмы решения:

- характеристики комплекса задач;
- используемая информация;
- результаты решения, например, информация для выдачи выходных сообщений и данные для использования исключительно внутри системы;

6. Программное обеспечение:

- структура ПО;
- функции частей ПО;
- методы и средства разработки ПО, например, инструментальные средства проектирования модели предметной области, структур баз данных, разработки, применяемые методологии разработки, проектирования системы или интерфейсов;
- операционная система, например, совместимые или предпочтительные ОС для клиентской части ПО и серверов (баз данных);
- средства, расширяющие возможности ОС.

7. Информационное обеспечение:

• состав информационного обеспечения:

— **внутримашинная** информационная база — набор электронных таблиц и других объектов баз данных для обработки и хранения данных,

— **внемашинная** информационная база — информация, поступающая в бумажном виде (например, нормативно-справочная информация, годовые бюджеты);

• организация информационного обеспечения:

— принципы организации информационного обеспечения, например, БД должна представлять собой взаимосвязанные реляционные таблицы,

— обоснование выбора носителей данных, например, расчет числа запросов, которые необходимо обрабатывать, на основе статистики предыдущих периодов по числу пользователей, объему информации, частоте обращения к ней,

— принципы и методы контроля в маршрутах обработки данных, например, контроль форматов данных, контроль заполнения обязательных полей,

— описание решений, обеспечивающих информационную совместимость АС с другими системами;

- организация сбора и передачи информации:

— основные источники информации, например, информация из внешних систем, бумажные носители информации, передаваемые по электронной почте файлы,

— общие требования к организации сбора, передачи, контроля и корректировки информации;

- построение системы классификации и кодирования:

— классификации объектов, принятые для применения в АС, например, Общероссийский классификатор валют (ОК 014—94), Коды представления названий стран (ИСО 3166—93), внутриорганизационный справочник сотрудников,

— классификации объектов, принятые для применения в АС;

- организация внутримашинной информационной базы:

— принципы построения,

— структура, например, физическая структура БД системы (с полями, типами данных);

- организация внемашинной информационной базы (информации в бумажном виде от внешних организаций).

8. Комплекс технических средств системы:

- структура комплекса технических средств, например, принципы построения кластеров;
- вычислительный комплекс, например, серверы БД, приложений, веб-сервер, персональные компьютеры;
- абонентские пункты;
- аппаратура передачи данных.

9. Ведомость документов.

В результате подготовки технического проекта должен быть определен перечень компонентов модуля системы, необходимых доработок, процедур интеграции с существующей системой (при ее наличии и необходимости),

необходимый для реализации бизнес-модели предприятия To-Be. Так же однозначно утверждается перечень работ по настройке и доработке модуля системы, а также перечень и формы отчетов и первичных документов, получаемых из модуля системы. Помимо отчетов должны быть определены процедуры для справочников системы (при ее наличии), которые необходимо использовать для импорта, и тех справочников, для которых необходимо разработать механизмы двустороннего обновления информации.

Важно отметить, что после детального определения и согласования всех вышеописанных параметров необходимо к моменту начала следующего этапа выбрать и подготовить также среду разработки и тестирования для проведения развертывания системы.

3.3.2. Рабочее проектирование и прототипирование

В целях снижения риска создания не соответствующей требованиям и техническому проекту системы перед разработкой возможно создать ее прототип (макет) для проверки предлагаемых архитектурных/функциональных/интерфейсных решений на практике с будущими пользователями.

Прототип — «черновая» реализация интерфейса и базовой функциональности системы для анализа принципов ее работы и тестирования совместно с будущими пользователями в целях дальнейшей доработки и совершенствования.

Обратная связь в рамках прототипирования, организованная на первых этапах жизненного цикла, очень важна для устранения замечаний по итогам проектирования и внесения доработок в интересах пользователей и заказчиков системы. В первую очередь создается макет, а уже потом проводятся тестирование, обсуждение и внесение корректировок в спецификации и прототип. Рассмотрим основные виды прототипов.

В качестве первой классификации можно выделить *«горизонтальное»* и *«вертикальное»* прототипирование. Первый случай («горизонтальное» прототипирование) предполагает моделирование исключительно пользовательского интерфейса и форм без фокусирования на логике обработки информации. Имитируются результаты действий или запросов при нажатии на элементы управления и осуществляются переходы между формами системы для формирования представления о реакции системы на те или иные действия пользователя (и какие не предусмотренные в текущей реализации действия потенциальный пользователь может совершить). Это позволяет выявить и устранить противоречия между сформированными на этапе анализа требованиями и их реализацией в техническом проекте. А в рамках «вертикального» прототипа, в отличие от предыдущего варианта, фокус переходит на саму структуру системы для проверки корректности и работоспособности сформированного архитектурного решения.

Второй (и более известной) классификацией является *«быстрое»* и *«эволюционное»* прототипирование. В первом случае (как и следует из названия) по технологии RAD, более подробно рассматриваемой в гл. 6, посвященной методологиям разработки, создается макет определенных компонентов системы для более предметного и эффективного диалога между разработчиками кода (интерфейсов) и пользователями. Важно, что подобный прототип не призван в дальнейшем дорабатываться и становиться частью системы, соответственно, достигается значительная экономия времени и ресурсов, так как нет необходимости фокусироваться на технических деталях быстродействия или энергоэффективности системы.

Напротив, эволюционное прототипирование, по сути являясь итерационной разработкой, подразумевает создание в несколько итераций кода и тестирование полноценной рабочей системы с реализованными алгоритмами и целевым уровнем производительности.

3.4. Разработка

3.4. Разработка

- Закупка ПО
- Настройка конфигураций
- Создание ролей пользователей
- Миграция данных
- Разработка сценария тестирования
- Тестовая эксплуатация
- Доработка по результатам тестирования
- Прием результатов тестирования

Ко времени старта реализации системы, как уже было сказано, должна быть выбрана и подготовлена среда разработки, а также выбрана методология, в соответствии с которой будет осуществляться управление разработкой. От нее зависит очень многое: принцип взаимодействия команды разработки и внедрения (между собой и с основными стейкхолдерами проекта), следование одной из моделей жизненного цикла ИС (каскадной, спиральной и пр.), длительность самого процесса разработки и тестирования, а также прочие важные аспекты процесса реализации системы. Подробнее различные методологии как управления проектами, так и непосредственно разработки, будут рассмотрены в следующей главе.

3.4.1. Закупка ПО

Каждая организация выбирает собственные, оптимальные именно с учетом ее специфики, критерии выбора между рядом платформ и решений. Наиболее распространенными критериями являются:

- функциональность системы;
- надежность и стабильность работы решения (устойчивость к различным режимам работы и степени активности пользователей);
- наличие и качество встроенных средств администрирования и управления;
- стоимость (внедрения, лицензий, поддержки);
- удобство организации услуг по поддержке и сопровождению системы;
- наличие отраслевого решения;
- наличие успешного опыта реализации проектов на базе данной платформы (надежность системы);
- опыт и компетенции организаций-подрядчиков;
- учет корпоративной специфики и соответствие корпоративным стандартам.

Список критериев очень индивидуален. Он зависит от специфики проекта и, как правило, формируется из разработанных на этапе проектирования требований к системе.

Следующим шагом будет организация непосредственно процедуры закупки (например, путем открытого запроса предложений), выбор поставщика и заключение договора на поставку ПО.

Типовая структура договора

1. **Предмет договора.** Например, поставка, установка ПО, монтаж оборудования, обучение персонала заказчика.
2. **Стоимость договора и порядок оплаты.** Сумма оплаты, сроки и размер платежей, форма оплаты (наличный/безналичный платеж).
3. **Права и обязанности сторон.** Порядок поставки, установки и ввода в эксплуатацию ПО, предоставления информации со стороны поставщика. Порядок назначения ответственных лиц, предоставления информации со стороны покупателя.
4. **Ответственность сторон.** Условия выплаты неустойки / расторжения контракта в случае невыполнения или ненадлежащего выполнения обязательств (в том числе несвоевременной поставки или оплаты). Порядок разрешения споров, в том числе в случае действия обстоятельств непреодолимой силы.
5. **Гарантийное сопровождение и авторский надзор.**
6. **Условия конфиденциальности.**

3.4.2. Настройка конфигураций

К моменту инсталляции системы на стенде предприятия заказчиком должна быть обеспечена устойчивость работы ЛВС стенда, проведена закупка программно-технического обеспечения в виде основного внедряемого программного продукта (самой системы) и ключей электронной защиты, а также завершены все работы предшествующих фаз жизненного цикла. Программно-техническое обеспечение для установки и тестирования системы, в том числе клиентские лицензии, предоставляет заказчик (из числа закупленных ранее либо через осуществление их закупки на этом этапе).

В процессе инсталляции модули системы устанавливаются на стендовый сервер с проведением основных необходимых настроек под подлежащие автоматизации бизнес-процессы предприятия заказчика. В зависимости от типа и особенностей системы в ней могут быть реализованы разные механизмы дополнения и изменения объектов в системе.

Создание кода программы. Единого языка программирования, использующегося для всех приложений, просто не бывает. Сейчас для многих стандартных пользовательских приложений используется комбинация SQL и C# для серверной программной части и HTML/CSS/JavaScript для создания интерфейса пользователей.

Пример

Современные ERP-системы наиболее часто используют собственные языки программирования, в связи с чем поддерживающие их специалисты должны обладать знаниями не только по работе с формами и пользовательским интерфейсом, но и в первую очередь по структуре системы и быть способными безопасно внести необходимые изменения в код. Вот некоторые сочетания ERP-систем и используемых в этих системах внутренних языков программирования:

- SAP – ABAP;
- Oracle – PL/SQL, Java;
- Microsoft Dynamics AX – среда разработки MorphX с языком программирования X+;
- Microsoft Dynamics NAV – графическая среда разработки C/SIDE с языком программирования C/AL.

В современных коммерческих продуктах практически всегда присутствуют средства автоматизированной настройки системы («конфигуратор», «конструктор»). В некоторых случаях код программы может являться самодостаточным способом настройки системы. Разница между ними состоит в возможностях внесения изменений (бизнес-логика и правила работы приложения, как правило, требуют большей и более сложной работы, нежели просто добавление нового поля). Для более простых функций обычно используются стандартные конфигураторы системы, не требующие знания языка программирования.

Таким образом, может быть выстроена следующая иерархия возрастания сложности и функциональных возможностей внесения изменений: *Интерфейс пользователя* → *Конфигуратор* → *Код программы*.

Разумеется, крайне важно, чтобы конфигурация была тщательным образом документирована, особенно в части «надстроек» над стандартными функциональными возможностями, описываемыми в том числе в поставляемой вендором документации. Доработку кода необходимо проводить так, чтобы исключить наличие так называемого закрытого кода, недоступного для изменения. В противном случае, если в ходе разработки была создана определенная схема расчета или логика процесса, при необходимости внесения изменений ИТ-специалистам придется с нуля создавать требуемый код/функциональность. Именно поэтому специалисты должны быть обучены не только технике конфигурирования продукта, но и особенностям конкретной конфигурации.

Настройка параметров системы в режиме конфигуратора. Примером системы, работающей в подобном режиме конструктора, является Salesforce.com. Эта система позволяет разработчику определять множество параметров, таких как: формат данных создаваемых полей, их взаимосвязи, формат отображения, источники данных (ручной ввод, вычисление по формуле или получение информации из других источников).

В том или ином виде конфигураторы присутствуют практически во всех системах, но объем их возможностей (и требования к квалификации работающих с ними специалистов) значительно различаются.

Настройка полей и создание библиотек системы. Наиболее простой из типов конфигурирования. По сути, таким образом предоставляется возможность расширить информацию об объектах системы и расширить возможности их взаимодействия. Дополнительные прописываемые поля, типы данных и прочие элементы позволяют кастомизировать каждый объект, а также описать возможные для него действия.

Пример

Возможность дополнить карточку сотрудника в системе набором новых полей с выбором типа каждого поля и установкой для них ограничений. Так, если в стандартной конфигурации существуют только три поля: Идентификатор сотрудника, Фамилия и Имя, то можно добавить любые дополнительные поля, например, Дата рождения, Должность, Дата принятия в штат. Для системы в дальнейшем не будет различий, были ли созданы поля автоматически или вручную, и они все в полной мере могут быть использованы в функциях поиска, фильтрации, создания отчетов.

Настройка логики бизнес-процессов. Вторым видом наиболее часто используемых объектов конфигурации являются сами процессы. Например, в системе могут прописываться пути движения документов, согласно которым после утверждения документа первым согласующим он автоматически направляется второму согласующему с отправкой уведомления о статусе «владельцу» документа в системе.

Пример

Приведем пример настройки логики бизнес-процесса выплаты бонусов сотрудникам филиала компании на основе результатов продаж. Бонусы планируются к выплате на периодической основе.

На примере конфигурации в системе SAP в таком случае объектами настройки могут быть:

- создание вида начислений к счету;
- определение схемы перерасчета, схемы учета результатов для заказа;
- определение вида заказа для отражения бонуса;
- ведение профиля расчета, присвоение профиля расчета виду заказа.

Интерфейс пользователя, расположение элементов на вкладках. Конфигураторы большинства систем позволяют изменять расположение объектов, скрывать некоторые из них, создавать персонализированные фильтры и настройки для групп пользователей. Данный шаг подробнее рассматривается ниже.

По результатам конфигурирования системы подрядчик подготавливает описание оптимальных настроек сервера БД и рекомендации по развертыванию рабочих мест, которое и проводится на следующем шаге. Это также позволяет выявить и устранить ошибки и неточности в программном продукте, затем подготовив описание *тестового сценария* (контрольного примера действий пользователя) для проверки корректности и актуальности осуществленных настроек.

После инсталляции и настройки модулей системы справочники и данные должны быть развернуты, формы ввода данных адаптированы, подготовлен акт сдачи-приемки работ.

3.4.3. Создание ролей пользователей

Роли пользователей (и определяемые ими права доступа) являются именно тем инструментом, который определяет:

- какой именно функциональностью системы будет пользоваться тот или иной сотрудник (группа сотрудников);
- какие данные будут ему доступны;
- какие он будет иметь права на чтение, редактирование информации, ее экспорт (!) и удаление?

Отдельно отметим экспорт информации. Эта функциональность, предоставляемая системой, тесно взаимосвязана с информационной безопасностью. Поэтому возможность выгрузки конфиденциальных данных должна быть предоставлена только ограниченному кругу лиц.

С технической точки зрения подобное разграничение доступа позволяет одному пользователю иметь несколько ролей по отношению к разным объектам системы.

Роли пользователей зависят от профиля компании и специфики системы, однако в целом они различаются по правам доступа к каждой из категорий данных. В списках доступа чаще всего задаются следующие возможности (по возрастанию объема прав).

- **R (read, чтение).** Возможность видеть элементы страницы и просматривать прикрепленные файлы.
- **W (write, запись).** Возможность изменять и редактировать значения полей страницы, добавлять файлы (возможность чтения, разумеется, сохраняется).
- **D (delete, удаление).** Возможность удалять страницу вместе с прикрепленными файлами (доступ на чтение/запись при этом сохраняется).
- **A (admin, администрирование).** Возможность добавлять/удалять дополнительные поля, менять формат представления, предоставлять и отзывать доступ к элементам (при сохранении возможности чтения, записи и удаления).

Получается логическая связка следующего вида: Учетная запись пользователя — Роль пользователя — Права доступа (комбинация прав и элементов, к которым осуществляется доступ).

Пример

Как показано на рис. 3.1., несколько сотрудников (например, работающих на равных позициях одного подразделения) могут обладать одной ролью в системе (например, аккаунт-менеджера). В таком случае все права доступа будут прописываться не для каждого отдельного пользователя, а для ролей (что гораздо эффективнее с точки зрения трудозатрат, в особенности при внесении изменений).

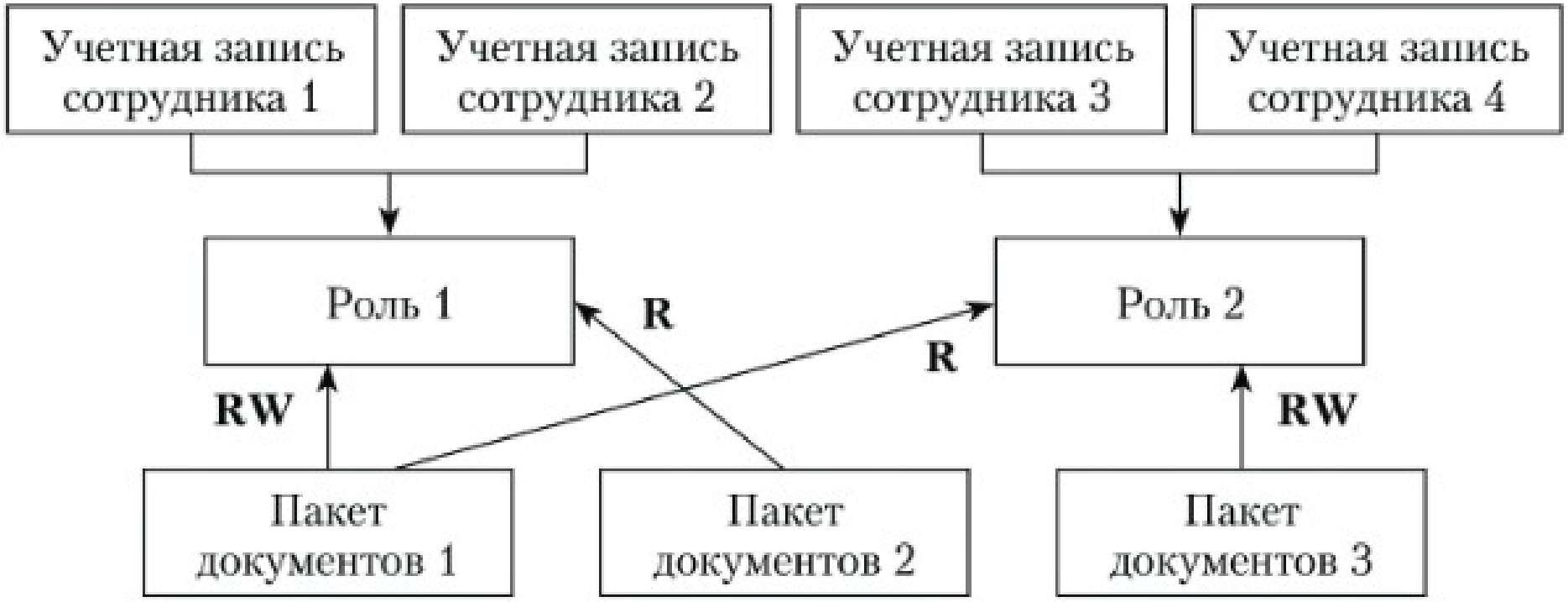

Рис. 3.1. **Учетная запись сотрудника, роль и пакеты документов**

Далее на рисунке показаны пакеты документов, к которым разрешен доступ, однако тот же самый принцип действует и для полей (групп полей) и страниц в системе. Так, в системе CRM аккаунт-менеджеру будет доступна возможность просмотра, редактирования и удаления мобильных телефонов его клиентов, но может быть отказано в доступе на внесение изменений в финансовые детали осуществления платежей. И напротив, сотрудники бухгалтерии с ролью «специалист по расчетам» смогут редактировать любые финансовые данные, но не будут видеть мобильные телефоны и историю сообщений руководства компании-клиента, которые доступны аккаунт-менеджеру.

3.4.4. Миграция данных

Данные на протяжении своего жизненного цикла претерпевают огромное число действий над ними: создание, обработка/редактирование, резервирование, копирование, перемещение в новую виртуальную среду, новую систему. Миграция данных является очень важной составляющей процесса внедрения системы — по сути, на этом этапе создается механизм, позволяющий автоматически переносить основные справочники и другие данные в модули системы.

Важнейшим шагом в первую очередь является непосредственно перенос (первичная загрузка) и настройка **основных справочников**. По сути, это все те данные, которые являются едиными для различных модулей и систем предприятия и используются ими совместно и одновременно: справочники поставщиков, клиентов, складов, материалов, продукции (номенклатуры) и др. Именно благодаря наличию подобных основных данных не происходит дублирования записей, а в при внедрении систем не приходится заниматься ручным поиском, сбором и вводом этой информации. В целях тестирования основные справочники новой системы заполняются тестовыми данными, достаточными для тестирования модуля и принципов поступления и передачи из него информации, и затем переносятся в систему. Если по результатам выверки тестовых данных не наблюдается ошибок, то можно говорить об успешной миграции справочников и на первый план выходит уже другой вопрос, а именно: поддержание всех этих данных в актуальном состоянии.

Другой категорией данных для переноса является информация из функционирующих на предприятии программных решений и *legacy*-систем, на смену которым внедряется новый программный продукт. При корректной подготовке на стадии проектирования передача данных в новую систему организовывается автоматизированно, через специальные коннекторы и шины данных. В противном случае миграция данных становится «ночным кошмаром», когда они должны быть в ручном режиме выгружены из существующих систем, приведены к единому формату и после определения соответствия полей и типов данных загружены в новую систему. Но ведь данные в существующих системах могут обновляться ежесекундно. Насколько допусти́м этот простой на время миграции и как избежать всех проблем, ассоциированных с распределенным онлайн-вводом, не говоря уже о проблеме ресурсоемкости и неимоверной трудоемкости данного процесса, связанного с огромным риском потери информации?

В таких случаях миграция организуется в онлайн-режиме. Такой способ предполагает перенос данных при непрерывной работе всех серверов и приложений. Это особенно важно для критических для бизнеса приложений с высокими требованиями к доступности: Oracle, Exchange, биллинговых систем, систем электронной коммерции. Новые данные внедряются в уже существующий поток информации, объединяются с имеющимися в системе данными на логическом уровне через отдельный адрес диска, который после успешной миграции удаляется или архивируется.

К наиболее распространенным рискам миграции данных и причинам их возникновения можно отнести:

- **недостаточные компетенции/опыт** команды проекта в миграции данных (в том числе в части формирования документации об уже осуществленных проектах);
- **отсутствие единого ответственного за данные**, что может привести к невозможности определить наиболее корректные и актуальные данные;
- **низкое качество переносимых данных** (дублирование, некорректные форматы данных);
- **нестабильность целевой системы для переноса данных** — если разработка еще не завершена и миграция данных выполнена преждевременно, могут возникнуть ошибки или потеря информации;
- **несвоевременность организации синхронизации данных** — актуализация данных должна производиться в новой системе сразу после миграции;
- **низкая степень документированности процесса миграции** — дан-

ный риск относится не только к невозможности использования получен-
ного опыта для выполнения дальнейших проектов, но и в большей степени
связан с невозможностью в случае необходимости продемонстрировать
аудиторам или компетентным контролирующим органам отчеты по датам
и масштабам перемещения данных. А значит, потенциально может возник-
нуть огромное количество проблем, связанных с доказательством досто-
верности предоставленной отчетности. Особенно это актуально, если дан-
ные для отчетности выгружались в период параллельной работы старой
и новой систем, при наличии различий в объемах данных, хранившихся
в них в конкретный момент.

Таким образом, миграция данных — длительная, кропотливая, требующая большого внимания работа, связанная с высоким риском потери данных. В девяти из десяти случаев основной сложностью, с которой сталкиваются команды проекта при миграции, является дублирование данных. Ниже перечислены основные причины появления подобных «данных-двойников»:

- различия в написании объектов;

Пример

- Организационно-правовые формы и формы собственности («АО» Вымпел, ЗАО «Вымпел», ВЫМПЕЛ). Это приводит к путанице: не ясно, идет речь о разных предприятиях или об одной и той же компании?
- Использование латиницы и сокращений: Ivanov Ivan, Иванов И.

- несовпадение информации из-за устаревания данных;

Пример

В одной системе в качестве адреса ООО «Вымпел» указывается г. Воронеж, а в другой — г. Москва. Разница может быть обусловлена переездом компании либо же разной трактовкой понятия «адрес» сотрудниками, вводившими информацию в систему: в первом случае — юридический адрес компании, во втором — фактический.

- несовпадение форматов данных.

Пример

- Неиспользование выпадающих списков для полей с ограниченным числом значений: г. Балашиха в одной системе и «Московская область» в другой.
- Использование текстовых полей для написания цифр (например, в случае миграции Excel-таблиц): 84951234567 добав 1111.
- Стиль написания цифр: +7 (495) 123-45-67, 84951234567.

Во всех этих случаях необходима тщательная выверка всех данных, проводящаяся частично в автоматизированном режиме (например, для замены «+7» в телефонных номерах на «8», удаления лишних пробелов и других символов), частично в ручном режиме пользователями.

3.4.5. Разработка сценария тестирования

Сценарий тестирования включает описание начальных условий тестирования, входных данных, действий пользователя и ожидаемого результата. При помощи сценариев тестирования проводится как тестирование работоспособности функций ИС/ПО, так и тестирование работы при различной нагрузке. Иными словами, сценарий тестирования может объединять в себе элементы функционального и нефункционального тестирований.

Как правило, обычно создается не один сценарий тестирования, а набор таких сценариев. После этого они группируются по какому-то признаку, будь то последовательность реализации тестовых сценариев, применение в различных видах тестирования и т.п.

В идеальном случае результаты реализации сценариев тестирования должны полностью совпадать с ожидаемыми результатами. В противном случае все расхождения должны быть объяснимыми. В ряде случаев может потребоваться устранение расхождений.

Среди других характеристик сценария тестирования:

- проверка наиболее «слабых» мест системы для повышения вероятности обнаружения ошибок;
- выполнение необходимого минимума действий для проверки (неизбыточность);
- явное выявление ошибок и недоработок;
- понятность и логичность описания примера для пользователей.

Сценарии тестирования также могут относиться к конкретной проверяемой области: конфигурации системы, удобству использования, безопасности, взаимодействию или интеграции компонентов. Однако может создаваться и общий пример, с единым описанием и дальнейшей детализацией, в зависимости от специфики проверяемой области и задач тестирования.

Таким образом, ко времени старта тестовой эксплуатации должны быть подготовлены следующие элементы модели тестирования:

- **набор исходных данных**, условий тестирования, планируемых результатов;
- **методика испытаний**: технический документ, описывающий настройку системы для проведения теста и принцип оценки итоговых результатов;
- **сценарии тестирования**;
- **тестовый скрипт** (опционально: в случае использования автоматизированных инструментов тестирования).

Формальным основанием для старта тестовой эксплуатации могут служить:

- протокол совещания группы внедрения о передаче модуля системы в тестовую эксплуатацию;
- регламент реализации сценариев тестирования;
- перечень замечаний ключевых пользователей, собранных при предварительной реализации сценариев тестирования, с описанием решений по их устранению, отраженный в протоколе совещания группы внедрения.

3.4.6.Тестовая эксплуатация

Между первоначальной настройкой системы на стенде по сформированным и утвержденным требованиям и ее полномасштабным вводом в эксплуатацию лежит один из самых сложных с точки зрения согласования интересов всех сторон этап тестовой эксплуатации. Как правило, прием системы в эксплуатацию и обучение пользователей проводятся исключительно после подтверждения согласия заказчика с «итоговой версией». Разумеется, для проведения опытной эксплуатации составляется отдельный регламент (методика). Все замечания, получаемые в ходе работы с системой выбранной группы ключевых пользователей, заносятся в отдельный журнал запросов на изменение с расстановкой приоритетов исполнения.

Основными блоками/классами тестов являются компонентное, модульное (*unit testing*), интеграционное и системное тестирование.

1. **Компонентное/модульное тестирование** (*component/unit testing*). Его основная цель состоит в идентификации ошибок реализации модулей в ходе проверки работоспособности (например, при различных поступающих на вход данных). Проверяются отдельные объекты, функции, классы, модули.

2. **Интеграционное тестирование.** Данный тип тестирования проводится для всей системы в целом. Его основная задача — проверка корректности передачи информации и взаимодействия между всеми модулями (компонентами) системы и слоями архитектуры (в частности, прикладным ПО, инфраструктурными сервисами, ОС). Особенно данный тип тестирования актуален для клиент-серверных и распределенных систем.

3. **Системное тестирование** (в том числе функциональное тестирование). Определение степени соответствия созданной ИС исходным требованиям (функциональным и нефункциональным). Проводится проверка корректности настройки внешних интерфейсов системы. Оцениваются различные характеристики надежности, производительности, отказоустойчивости системы, в дальнейшем также проверяемые в ходе нагрузочного тестирования.

В качестве дополнительных проверок могут создаваться тестовые окружения для ИС при **эмуляции** работы отдельных компонентов систем, внешнего окружения и действий пользователей. В частности, эмуляция позволяет смоделировать работу еще не созданных компонентов системы (модуля, сервиса) для более раннего проведения тестирования.

Тестирование в зависимости от специфики системы и предметной области может проводиться в ручном, автоматизированном или полуавтоматизированном режиме.

1. **В ручном режиме** специалисты по тестированию самостоятельно проходят весь сценарий тестирования. Этот способ применяется для крайне динамичной разработки в условиях постоянных изменений. По результатам составляется протокол тестирования, в который заносятся все замечания, комментарии и предложения.

2. **Автоматизированное тестирование** значительно упрощает и ускоряет процесс тестирования, предоставляет возможность провести большее число проверок, в том числе до и после внесения доработок в тестовую среду.

3. **Полуавтоматизированное тестирование** выполняется по заданному сценарию под постоянным контролем тестировщика. Таким образом, оно сочетает преимущества автоматизированного тестирования с возможностью в случае необходимости изменить сценарий его выполнения и организовать дополнительные проверки.

Тестирование как наиболее важный элемент обеспечения и контроля качества может и должно быть организовано по специальной методике в соответствии со стандартами. Среди наиболее часто применяющихся стандартов:

- ГОСТ Р ИСО/МЭК 12119–2000. Пакеты программ. Требования к качеству и тестирование;
- ГОСТ Р ИСО/МЭК 12207–2010. Процессы жизненного цикла программных средств;
- ГОСТ Р ИСО/МЭК 14764–2002. Сопровождение программных средств;
- ГОСТ Р ИСО/МЭК 15288–2005. Системная инженерия. Процессы жизненного цикла систем;
- ГОСТ Р ИСО/МЭК 15504–2009. Оценка процессов;
- ГОСТ 28195–89. Оценка качества программных средств. Общие положения;
- ГОСТ Р ИСО/МЭК 9126–93. Оценка программной продукции. Характеристики качества и руководства по их применению;
- **ГОСТ серии 19.** Единая система программной документации (ЕСПД);
- **ГОСТ серии 34.** Разработка автоматизированной системы управления;
- **IEEE 829.** Стандарт документирования тестирования программного обеспечения и систем.

Отдельно можно отметить унифицированный процесс RUP, который по своей сути исходит из идеи постоянного регулярного тестирования на каждой итерации.

Срок тестовой эксплуатации, как правило, не превышает одного месяца и не меняется при отсутствии или несвоевременном предоставлении замечаний. Среди выполняемых для организации тестовой эксплуатации мероприятий (все они проводятся по каждому функциональному модулю в отдельности):

- подготовка перечня замечаний ключевых пользователей, собранных в процессе окончательного тестирования контрольного примера (по каждому модулю системы), с описанием решений по их устранению, отраженного в Протоколе совещания Группы внедрения;
- предварительное тестирование командой внедрения контрольного примера (по отдельным модулям!) с занесением замечаний в журнал тестирования и последующим их устранением (в случае необходимости с дополнительными настройками и доработками);

- финальное тестирование с привлечением ключевых пользователей (здесь помимо функциональности крайне важно также проверить степень удобства и интуитивности интерфейса системы для пользователей);
- разработка общего руководства по работе с системой и инструкций пользователей по модулям системы (включая назначение и условия работы с модулем, описание операций по подготовке и самой работе);
- обучение пользователей базовым принципам работы с системой на контрольном примере (в отдельности для каждого модуля системы). В процессе обучения пользователь знакомится с интерфейсом системы, получает информацию по основным принципам работы и основным возможностям системы. В случае наличия разработанных методических руководств пользователю дополнительно предоставляется необходимая документация.

3.4.7. Доработка по результатам тестирования

Основная цель данного этапа — устранить замечания, возникшие в результате тестирования на контрольном примере, и при необходимости осуществить дополнительные настройки и доработки модулей системы. Как уже было сказано, на данном этапе важно разделять требования, критичные для успешной работы с системой, и дополнительные (*nice-to-have*) пожелания пользователей-экспертов, которые можно будет включить в следующие итерации работы и версии системы.

Что касается порядка организации данных типов проверки, то для первых сборок проводится поверхностное *smoke-тестирование*, направленное на критическую функциональность. Этот короткий цикл тестов должен подтвердить, что после сборки кода (нового или исправленного) приложение будет успешно выполнять свои основные функции. Код проверяется на предмет наличия быстронаходимых критических дефектов, которые необходимо срочно исправить, отправив на доработку. В случае их отсутствия и успешного прохождения тестирования приложение передается для проведения полного цикла тестирования.

Так как при *smoke*-тестировании в максимально короткие сроки покрывается максимально широкий объем функциональных возможностей, то существует и другой вид тестов, дополняющих первый (функции проверяются скорее «вглубь», с максимально тщательным тестированием каждого аспекта). Это обусловлено тем, что при проведении любого рода изменений необходимо обеспечивать их корректность и отсутствие негативного влияния на работоспособность и эффективность приложения.

С этой целью проводится *регрессионное тестирование* (*regression testing*), один из наиболее известных видов которого получил название *sanity-тестирования*. Это тестирование определенных участков кода, проводимое по результатам внесенных изменений. Его основная задача — удостовериться в том, что внесенные изменения не затронули работоспособность системы и не послужили причиной новых ошибок. К примеру, проверяется, действительно ли исправленные ранее ошибки исправлены, не приводят ли сделанные изменения к нейтрализации исправлений старых ошибок (регрессия багов, англ. *bug regression*) и есть ли новые ошибки.

В результате должны быть:

- проведены тестирование, выявление и устранение замечаний по контрольным примерам для каждого модуля системы (замечания обработаны, и либо сделаны соответствующие изменения в модуле, либо они отклонены с указанием причин);
- проведены дополнительные настройки и доработки модуля системы (в срок до четырех недель с момента передачи модуля системы в тестовую эксплуатацию и получения перечня окончательных доработок и настроек модуля системы);
- составлен протокол выполненных работ по устранению документированных замечаний ключевых пользователей заказчика по результатам тестирования контрольного примера по каждому модулю системы;

- ключевые пользователи обучены базовым основам работы с модулем системы на данных контрольного примера и готовы к его практической эксплуатации на своих рабочих местах, а также развертыванию на основном рабочем сервере и проведению опытно-промышленной эксплуатации (ОПЭ);
- сформирован протокол выполненных работ по обучению ключевых пользователей заказчика базовым основам работы с модулем системы на данных контрольного примера;
- подготовлен перечень окончательных доработок и настроек каждого модуля системы.

3.4.8. Прием результатов тестирования

Результаты тестовых испытаний заносятся в единый отчет для рассмотрения заказчиком. При этом организуется *приемочное тестирование* (*acceptance testing*), представляющее собой комплексную проверку компонентов системы на предмет их работы с плановыми параметрами производительности и выполнением всех функциональных требований, в том числе на различных конфигурациях.

Прием результатов испытаний проводится при выполнении следующих условий:

- специалисты группы внедрения со стороны заказчика владеют функциональностью модуля системы и готовы к его развертыванию на рабочем сервере и опытно-промышленной эксплуатации;
- все замечания, возникшие в ходе первоначального тестирования контрольных примеров, задокументированы и устранены;
- ключевые пользователи готовы к работе с модулем на своих рабочих местах;
- все замечания ключевых пользователей, собранные во время окончательного тестирования контрольного примера, обработаны, и либо сделаны соответствующие изменения в модуле, либо замечания отклонены с указанием причин отклонения;
- заказчиком утверждены итоговые документы.