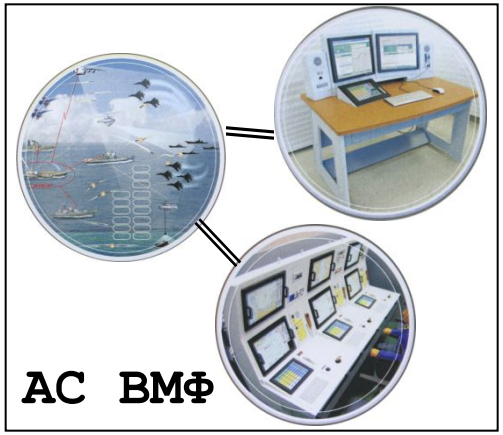
АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ, ИНТЕНСИВНО ИСПОЛЬЗУЮЩИЕ ПО

Инструментально – технологические среды разработки АС
IBM Rational, Eclipse, Microsoft Framework Solution



АС ВМФ

АС предприятий

АС производственных процессов

АС ВС

Бортовые системы

Коллективное проектирование непременно приводит к необходимости взаимодействия и обеспечения совместимости компонентов систем.

Поэтому разработка, как правило, ведется в условиях определённой инструментально-технологической среды проектирования, современные образцы которых охватывают обширную область поддержки технологий проектирования: от простых средств анализа и документирования до полномасштабных средств автоматизации.

Жизненный цикл программного продукта.

Стандартизация процессов разработки программных систем

Первый стандарт был утвержден в 1985 году, а в 1994-1996 годах был разработан и принят новый, значительно более детальный и глубокий **стандарт — MIL-STD-498**. Он представляет собой комплект из трех документов общим объемом около 600 страниц и устанавливает терминологию, процессы, задачи и объекты, используемые при разработке и сопровождении проектов программных систем. Основные положения этого военного стандарта согласованы с международным стандартом **ISO/IEC 12207:1995 («Процессы жизненного цикла программных средств»)**, но вместе с тем являются более конкретными и приближенными к практике.

В нашей стране попытка стандартизации процессов создания программных систем вылилась в создание комплекта документов под общим названием **«Единая система программной документации», который увидел свет еще в 1977 году** (да, были времена, когда мы оказывались впереди планеты всей не только в области балета) и периодически корректировался вплоть до последнего времени.

Определение. Под жизненным циклом понимается последовательность процессов, действий и задач, которые осуществляются в ходе разработки, эксплуатации (использования) и сопровождения программного продукта в течение всей его жизни, от определения требований до завершения использования.

Практически все стандарты (даже военные) предусматривают возможность их адаптации к особенностям конкретного проекта при условии соблюдения основных требований к технологии и показателям качества продукта.

Замечание. Необходимо подчеркнуть, что положения стандартов остаются справедливыми вне зависимости от предназначения, уровня сложности и состава коллектива разработчиков программного продукта, то есть даже в тех случаях, когда речь идет о небольшой программной утилите, а коллектив разработчиков состоит из одного человека.

ПРОЕКТИРОВАНИЕ

Начальный этап в разработке программного продукта (приложения) является наиболее критичным, поскольку на этой фазе определяется общая концепция создаваемого продукта. Если проект в своей основе неудовлетворителен, впоследствии трудно будет что-либо кардинально изменить в лучшую сторону.

Эта часть процесса разработки включает не только определение **цели** и **характеристик приложения**, но и понимание того, кто является его **потенциальными пользователями — их задачи, намерения, цели**.

Это предполагает учет таких показателей, как например:

- возраст пользователей,
- их пол,
- экспертные знания,
- уровень опыта,
- физические ограничения,
- специальные потребности и т.д.

Продумайте структуру приложения и метафоры, которые могут быть применены при ее реализации. Решению указанной проблемы способствует наблюдение за работой пользователей при выполнении ими задач в данной предметной области.

Согласно Государственного стандарта ГОСТ 34.601-90 [107] проектирование АС осуществляется в несколько этапов:

1. Формирование требований к АС;
2. Разработка концепции;
3. Разработка ТЗ на создание АС;
4. Разработка эскизного проекта;
5. Разработка технического проекта;
6. Создание рабочей документации;
7. Ввод в эксплуатацию;
8. Сопровождение АС.

В представленной последовательности проектной процедуре «Разработка ТЗ» предшествуют два крупных этапа, предусматривающих интенсивное взаимодействие представителей различных областей человеческой деятельности, вовлеченных в разработку.

На всем протяжении такого взаимодействия осуществляется такие действия, как анализ реализуемости АС, системный анализ, связанные с активным извлечением, анализом и управлением требованиями. Далее из полученного множества выбираются принципиальные проектные решения, формируется целостное концептуальное представление и документальное описание изделия (этап концептуального проектирования).

Техническое задание

Техническое задание представляет собой высокоуровневое определение целей проекта, которое формулируется в виде документа объемом две-три страницы (не считая титульного листа). Техническое задание обычно пишется до перехода к этапу детального определения требований, хотя в зависимости от потребностей заказчика и вашего проекта иногда имеет смысл создать комбинированный документ, оптимальным образом отвечающий вашим задачам.

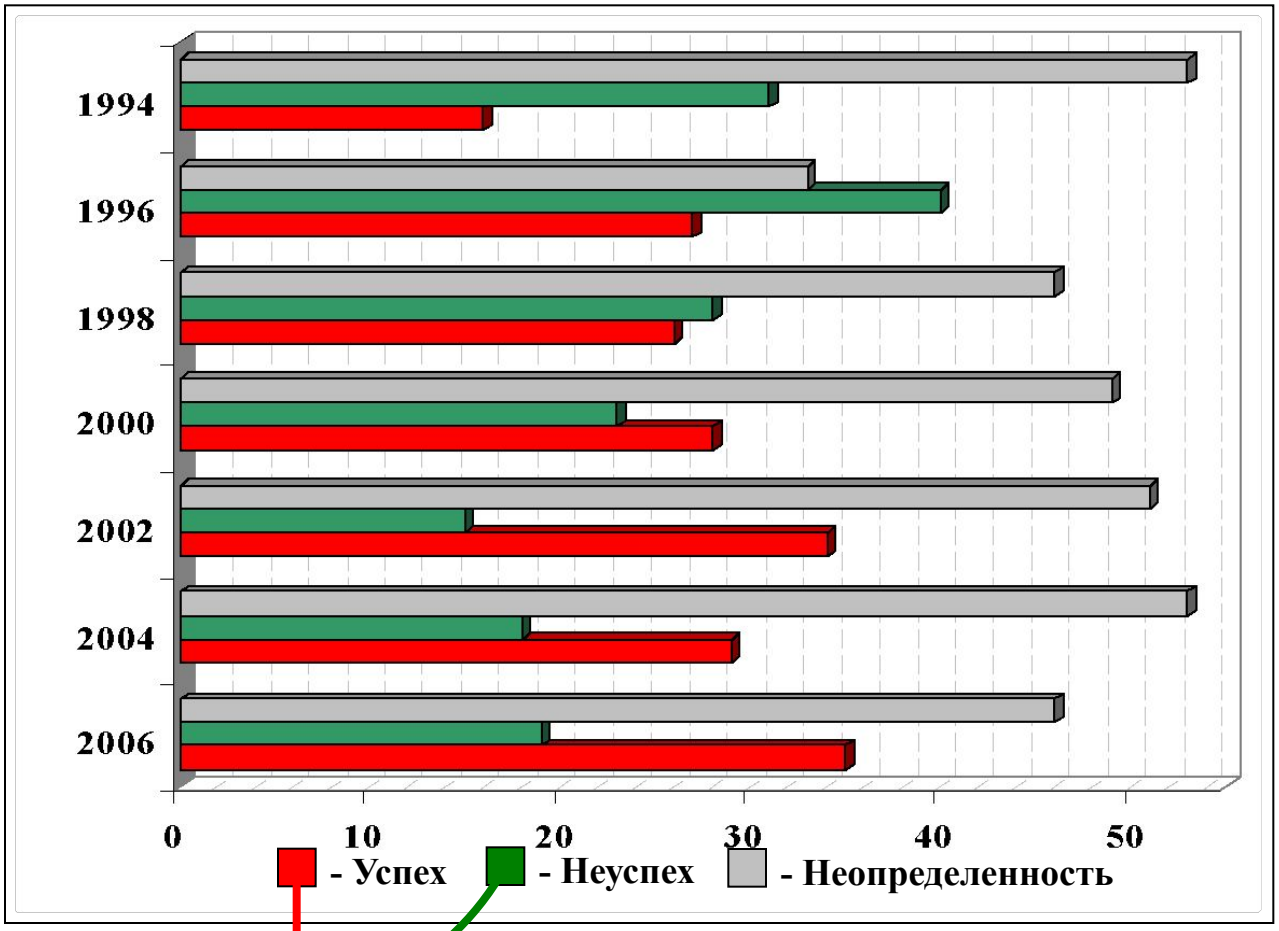
В общем случае техническое задание должно использоваться для достижения консенсуса между вашей командой и заинтересованными лицами со стороны заказчика.

Техническое задание определяет, что подается на **вход проекта и что получается на выходе**, а также действующие условия и ограничения.

На этой стадии заказчик нередко просит предоставить «приблизительную оценку той работы, которую вам предстоит выполнить. Давать какие- либо определенные ответы несколько рискованно. Постарайтесь избежать любой конкретики или обязательств до выяснения подробностей. Пока не написано коммерческое предложение и/или документ с требованиями, определить точную стоимость проекта попросту невозможно.

УСПЕШНОСТЬ РАЗРАБОТОК АВТОМАТИЗИРОВАННЫХ СИСТЕМ

Относительная стоимость устранения ошибки на разных стадиях жизненного цикла



$\frac{1}{3}$ от общего числа проектов

Требования

Еще одна область высокого проектного риска связана с недостатками в понимании проектных требований. План предусматривает соответствующее время и методы, чтобы собрать и изучить основные требования к продукту, а также назначить им приоритеты. В особом внимании нуждаются требования к ПИ, практичности, согласованности, обучению и интеграции.

Во время сбора требований многие детали, относящиеся к пользовательскому интерфекйсу, никогда не упоминаются в явном виде. Об их существовании многие пользователи даже не подозревают. Например, такое требование, как соответствие стандартам пользовательского интерфейса для определенной платформы или уровня согласованности и интеграции набора программ оказывает существенное влияние на работу бригады разработчиков продукта.

Требования в отношении практичности и согласованности должны быть измеримыми и поддаваться проверке на соответствие. Например, выполнение задачи за время, которое на 25% меньше, чем для унаследованной системы, является весьма реальным и измеримым требованием.

Работа с требованиями

Рис. 1.1 Процесс работы над требованиями

Шаблоны требований

Перейти от абстрактных рекомендаций по созданию хороших, полных и логичных спецификаций требований к непосредственному оформлению текста документа требований позволяют шаблоны.

Шаблон VISION (**Автор – Карл Вейгерс**) определяет вид и границы проекта. Сам шаблон представляет собой развернутую структуру глав и параграфов («Бизнес требования», «Видение решений», «Масштабы и ограничения»), с пояснениями и рекомендациями, представленными как в текстовой форме, так и в табличной (Таблицы 1.1. и 1.2.). В самом начале шаблона VISION располагается таблица вносимых изменений в текст уже готового документа (Таблица 1.1.), в котором наряду с именем и датой фиксируется основание для изменения документа Спецификации требований. Наличие такой информации позволяет сохранить в тексте логику и последовательность.

Шаблон Volere, в свою очередь, предоставляет всеобъемлющий список так называемых компонент, которые рекомендуется использовать в качестве основы для написания документа спецификаций требований. Шаблон может быть обработан с применением базового инструментария автоматизированной обработки требований или в любом текстовом редакторе. Текстовый документ шаблона предусматривает секции для каждого типа требований, соответствующие проектируемой системе (См. Рис. 1.2) [69].

Эскиз требования, это, по сути, руководство для написания каждого даже самого малого требования. На рисунке 1.2. представлена «карточка образцов». Особенностью шаблона является наличие раздела «Комната ожидания» (“Waiting room”), в котором документируются идеи, не вошедшие в текущий выпуск, но могут быть полезными для последующих проектов, и, таким образом, ни одна из идей не будет потеряна.

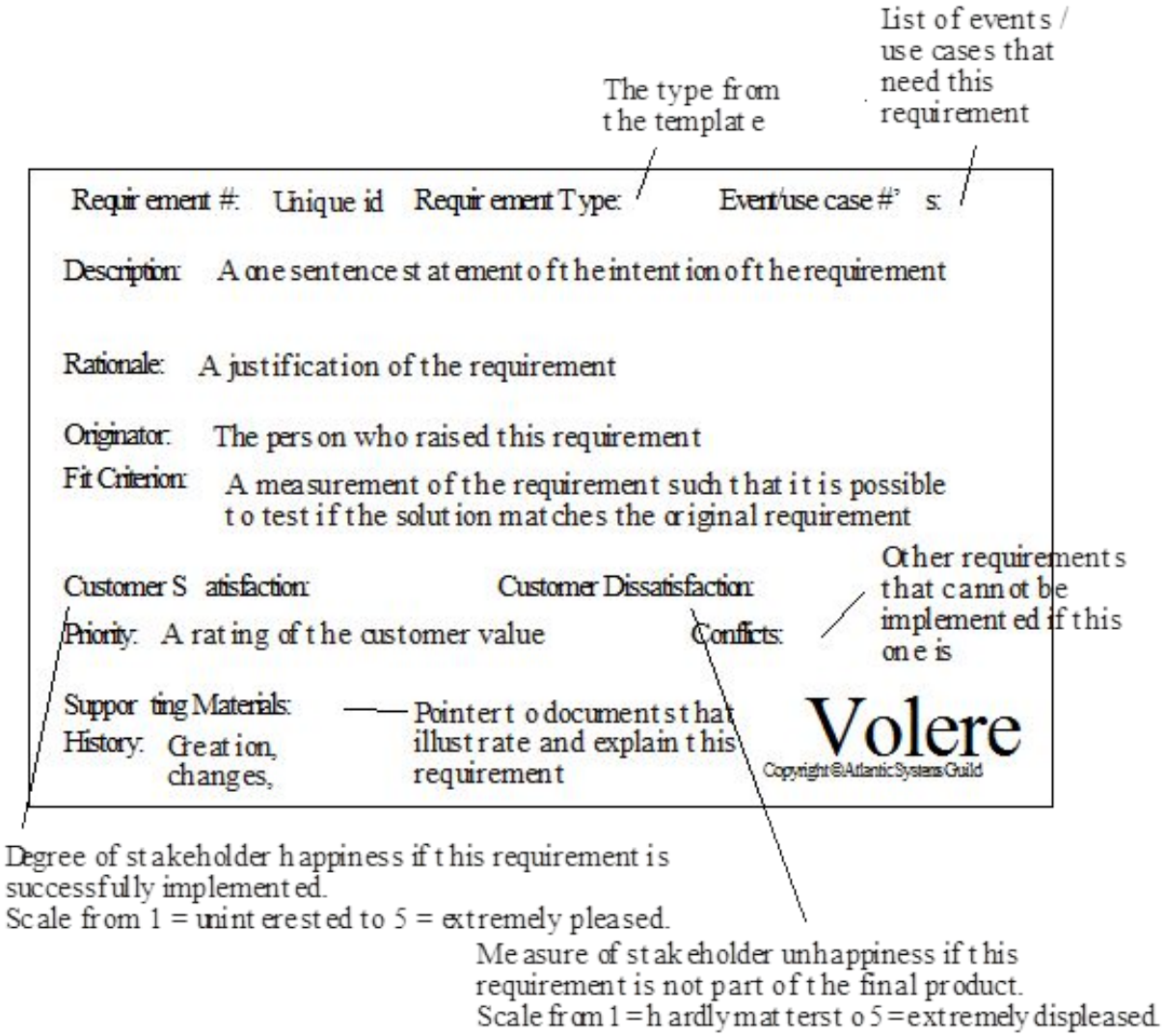


Рис. 1.2 Эскиз карточки простого требования в шаблоне Volere

Некоторые вопросы, связанные с требованиями, могут показаться неожиданными, другие — общими проблемами для ПИ. Ниже приведены очевидные вопросы, которые должны войти в требования.

- Стиль ПИ.
- Платформа и другие стандарты ПИ для приложения.
- Совместимость с ведущим ПО, работающим на данной платформе (например, приложение X или пакет У).
- Содержание экрана (например, данные и функции, необходимые в ключевые моменты выполнения задач).
- Поведение экрана (например, входной фокус на первом элементе управления при отображении экрана).
- Характеристики внешнего вида экрана (например, использование графики для отображения данных, представления и эстетические свойства).
- Методы взаимодействия пользователей с системой (например, доступ к командам, способы образования комбинаций клавиш и т.д.).

- Возможности работы с клавиатурой, включая поведение средств табуляции и циклическую работу клавиши табуляции.
- Обратная связь пользователя в ответ на состояние системы и время отклика.
- Пользовательский контроль над различными функциями.
- Запоминание результатов операций расположения и изменения размеров окна, а также данных, состояния и контекста.
- Возможности навигации для приложения.
- Сохранение данных пользователя при навигации.
- Запоминание промежуточных данных пользователя при навигации.
- Интерактивное обучение, поддержка производительности и справочная система.
- Предотвращение ошибок и восстановление системы после ошибок.
- Методы прямого ввода для устранения диалога.
- Проверка правильности значений полей, а также идентификация требуемых полей.
- Стандартное использование цвета, индикаторов, графики и т.д.
- Средства обеспечения доступа для пользователей с физическими недостатками.

Многие из перечисленных выше вопросов ПИ зачастую явно и конкретно не фиксируются в руководствах по стилю ПИ, документах описания требований или

В ряде положительных эффектов от применения в проектировании АС спецификаций требований (СТ) можно отметить следующие:

1. **Установление оснований для соглашения между клиентами и поставщиками** о том, что разрабатываемое ПО должно делать. Сюда входят полное описание функций, которые будут выполнены программным обеспечением, указанным в СТ, что поможет потенциальному пользователю определить, выполняет ли программное обеспечение их потребности или как программное обеспечение должно быть изменено, чтобы выполнить их потребности.

2. **Уменьшение объема усилий, затраченных на разработку.** Подготовка спецификаций требований вынуждает различные заинтересованные группы в организации клиента строго рассматривать все требования до начала разработки проекта и уменьшает в дальнейшем объем работ по перепроектированию, перекодированию, и повторному тестированию. Внимательное рассмотрение требований в спецификации требований может выявить упущения, недоразумения и нелогичности на ранних стадиях разработки, когда эти проблемы более легко исправляются.

3. **Обеспечение оснований для утверждения бюджета и сроков окончания разработки.** Описание изделия, которое будет разработано так, как определено в СТ является реалистическим основанием для оценки проектных затрат и может использоваться для согласования предложений или оценки

4. О**блегчение передачи и сопровождения.** В таком ключе СТ обеспечивают более легкую передачу изделия программного обеспечения новым пользователям или установку на новые машины, а также обеспечивают базу для дальнейшего улучшения готового изделия.

5. **Обеспечение основы для испытаний и проверок.** Организации могут развивать собственные планы испытаний и проверок намного более продуктивно при наличии хорошей документации

В международном стандарте «IEEE Std 830-1993. Рекомендации по разработке спецификаций требований программного обеспечения» определены ключевые требования к качественной спецификации:

- Характеристика недвусмысленности (Unambiguousness) означает отсутствие лексических, синтаксических и семантических ошибок;
- Характеристика полноты (Completeness) подразумевает описание всех значимых предметных областей в полном объеме;
- Характеристика целостности (Consistency) означает, отсутствие разного рода конфликтов и противоречий внутри документов СТ;
- Характеристика применимости (Usable) означает, что документ СТ должен без лишних деталей описывать весь жизненный цикл АС.

В настоящее время разработчиками программного обеспечения для АС активно используются программные средства, поддерживающие ту или иную методологию проектирования и разработки программного обеспечения. Наиболее известная методология – Rational Unified Process (RUP), предоставляющая команде разработчиков гибкую систему методов и средства коммуникации для общения на достаточном уровне формализации, с возможностью автоматизированного создания документов в соответствии с ГОСТами.

RUP реализует следующие принципы:

1) обеспечивается строгий подход к распределению задач и ответственности внутри оганизации-разработчика;
2) построение различных моделей проектируемой системы в ходе разработки, позволяет взглянуть на нее с различных точек зрения, может служить основой для дальнейшего улучшения готового изделия;
3) присутствует возможность использования опыта предыдущих разработок.

Таким образом, обеспечивается создание АС точно в срок и в рамках установленного бюджета качественного ПО, отвечающего требованиям функциональности конечных пользователей. В качестве языка моделирования в общей базе знаний RUP используется Unified Modeling Language (UML), который, кроме того, является международным стандартом

Концептуальное проектирование

Цель этапа концептуального проектирования заключается в определении основных архитектурных и технических решений, в формировании целостного документального описания и представление АС.

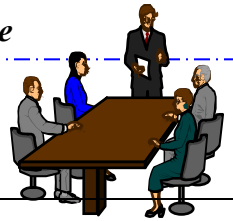
Основные концептуальные формы существования: **в форме системы требований**, в **форме архитектуры** и в **форме обобщённого проекта**.

Создание каждой из этих форм поддерживается национальными и международными стандартам, технологиями и ИТ средами проектирования. (ГОСТ 19, 34, IEEE [**Institute of Electrical and Electronics Engineers**] Институт инженеров по электротехнике и электронике 830 [Recommended Practice for Software Requirements Specifications], 1223[Guide for Developing of System Requirements Specifications]), **ISO/IEC 15288, 12207 (**«Information Technology — Software Life Cycle Processes»**)**

Так как ошибки этапа проектирования (концептуального) являются самыми «дорогими», то систему инструментально-технологических средств проектирования АС целесообразно дополнять средствами обнаружения, исправления и предотвращения ошибок в формулировках требований, в архитектурных описаниях АС и деталях концептуального проекта.

КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Интервьюирование
Мозговой штурм
Совещания
Анкетирование



Формирование требований

ГОСТ 34, 19

IEEE 830, 1223

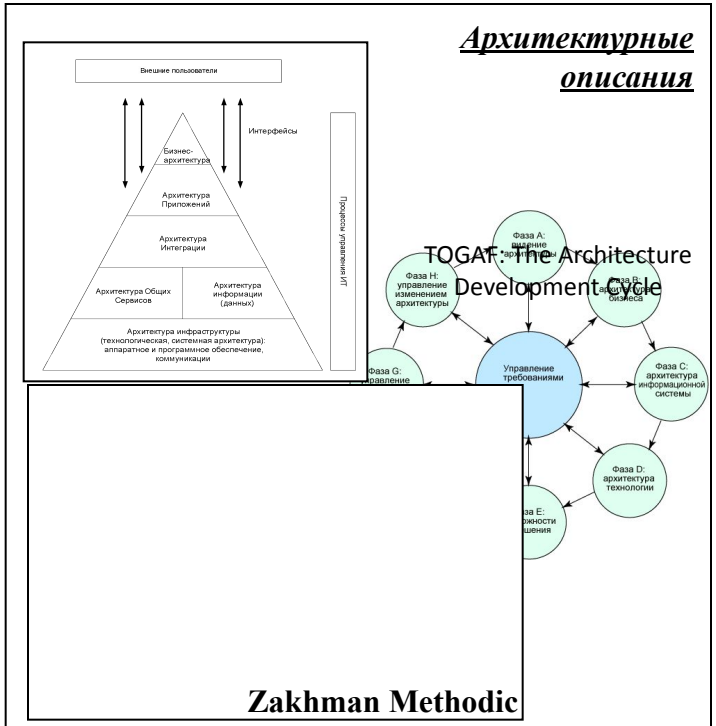
Руководства к разработке требований

Регламентирует взаимосвязи между управлением требованиями, разработкой требований и другими областями процессов

SEI-CMM

SWEBOK – *База знаний программной инженерии*

Архитектурные описания



IEEE 1471

Обобщенный проект

ISO/IEC 15288, 12207

ГОСТ Р ИСО МЭК 12207

ИНСТРУМЕНТАЛЬНО-ТЕХНОЛОГИЧЕСКИЕ СРЕДЫ ПО РАБОТЕ С ТРЕБОВАНИЯМИ

Языки формальных спецификаций

Алгебраическая семантика

OBJ, LOTOS, LARCH, SDL, Estelle, AMN

Моделе-ориентированная семантика

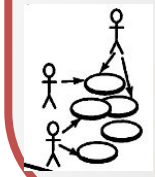
Z, VDM, B, Сети Петри,

Языки исполнимых спецификаций

Estelle (C), Eiffel (Pascal)

Графические языки спецификаций

DFD, ERD, STD, UML

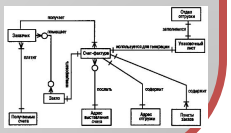


```
obj PEANO is
        sort Nat.
op 0:-> Nat.
op s_ : Nat->Nat.
Op _+_: Nat Nat->Nat
```

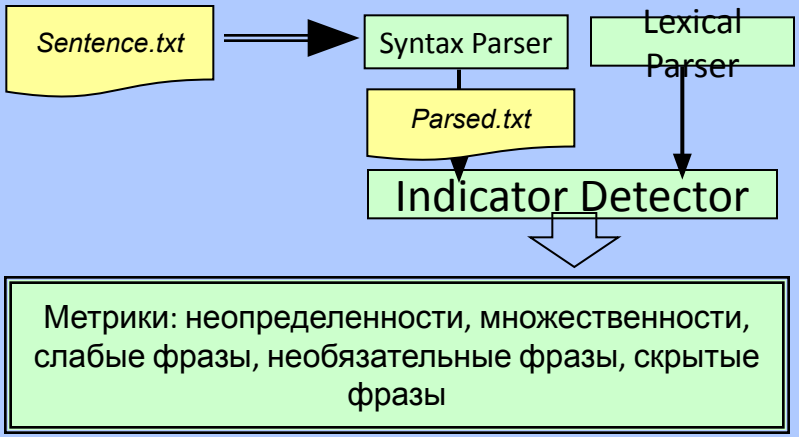
iroot: N→ N

A: N ·

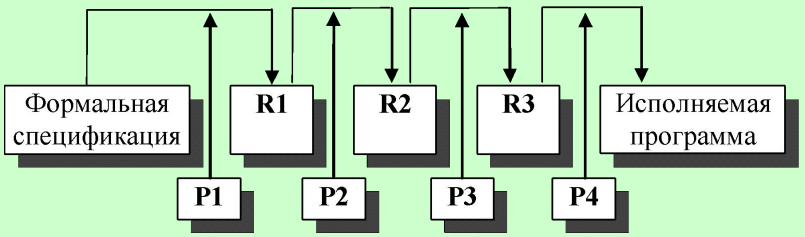
…



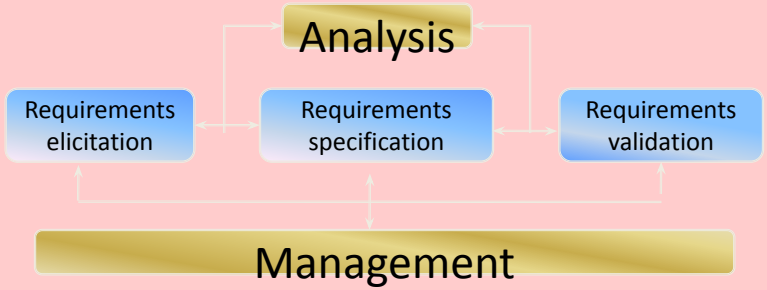
Инструментальные среды оценки качества требований



Системы автоматической генерации кода



Автоматизированные системы управления требованиями



Проектирование человеко-машинных интерфейсов

Создание такого вида систем в России регламентировано ГОСТами серии 34 и 19. Очевидно, что с развитием аппаратной базы и увеличением объема и сложности задач, которые необходимо решать с применением вычислительной техники, возрастает и сложность АС, что закономерно влияет на объем и сложность проектной документации, а также снижает способность разработчика решать задачи интуитивно, по мере их возникновения.

Недооценка сложности проекта и пренебрежение к изменению требований к АС ¬– основные причины определяющие, почему количество ошибок и промахов в разработках АС, интенсивно использующих ПО, все еще достаточно велико. Это обстоятельство послужило причиной появления ряда международных стандартов на работы и процессы проектирования (IEEE, ISO, SEI, CIMM и др.).

Сравнение успешности разработок в зависимости от размера проектов

	Завершились успешно	Потребовали дополнительных ресурсов	Завершились не успешно
Очень большие	2%	7%	17%
Большие	6%	17%	24%
Средние	9%	26%	31%
Небольшие	21%	32%	17%
Маленькие	62%	16%	11%
Всего	100%	100%	100%

Подходы к проектированию ПИ

Для получения эффективного результата разработки ПИ интерфейса используют различные подходы к проектированию:

1. **Подход, ориентированный на пользователя** (User Centered Design, UCD) — основным содержанием этого подхода является ориентация на пользователя, т. е. в первую очередь необходимо узнать, что хочет пользователь получить от проектируемого интерфейса. Далее в процессе проектирования полученные требования реализуются в продукте. При сборе информации используются методы наблюдения за работой пользователя, проводятся интервью.

2. **Системный подход** (System). Пользователь рассматривается как маленькая интеллектуальная часть системы «человек - программный продукт».

3. **Итеративный подход** (Agile) — метод последовательных приближений. Суть итеративного подхода заключается в создании изначально самого простейшего прототипа с целью показать заказчику н затем постепенно дорабатывать прототип, основываясь на реакции заказчика после каждого шага доработки.

5. *Экспертный подход (Genius).* Заключается в следующем: эксперт собирает важную, по его мнению, информацию, ведёт переговоры с заказчиком, задаёт нужные вопросы. На основе полученной информации создаётся интерфейс.

6. Целеориентированный подход проектирования *(Goal Centered Design).* Разработка интерфейса ориентируется на цель, которая будет достигаться данным программным продуктом.

7. Средоориентированный подход. Разрабатывается среда интерфейса как место деятельности оператора.

При разработке интерфейса целесообразно гибко пользоваться указанными подходами, учитывая при выборе методов: назначение разрабатываемого продукта, целевую аудиторию, время и бюджет разработки.

Деятельностный подход (Activity Centered). Изучается деятельность пользователя в целом, и постепенно оптимизируются ее отдельные моменты.

При анализе деятельности, предшествующем проектированию инструментального интерфейса, необходимы:

- выявление целей деятельности,
- способов достижения той или иной цели,
- установление уровня понимания этой цели работником,
- определение его мотивов.

При проектировании деятельности пользователей инструментальных интерфейсов важно обеспечить как устойчивость внимания и сосредоточенность, так и возможность переключения между видами работы. Задача проектировщика состоит в том, чтобы минимизировать сложность деятельности в рамках интерфейса, обеспечить системность интерфейса за счет того, что решение сходных задач должно реализовываться подобными действиями за счет одинаковых (или подобных) операций.

Деятельностный подход к проектированию человеко-компьютерного взаимодействия для конкретной проблемы предполагает глубокое изучение работы будущих пользователей в “докомпьютерном” варианте, анализ всех возникающих задач и описание деятельности по их решению.

Важно выявление основных целей и мотивов данной деятельности, описание отдельных этапов деятельности и выявление всех сущностей, с которыми работники имеют дело. Также необходим “деятельностный” анализ новой ситуации, возникающий после компьютеризации работы. Деятельность работника состоит из набора осознанных, мотивированных достижением цели действий, которые в свою очередь, сводятся к наборам операций. На каждом уровне иерархии необходимо выявление и четкое определение целей, которые связаны с осуществлением деятельности.

Проектирование и дизайн интерфейсов (UI Modeling Company)

Этап I. Предпроектный анализ

Работы по проектированию интерфейса начинаются с предпроектного анализа. На рабочей сессии с клиентом мы описываем видение проекта (vision), в котором рассказывается о его сути и целях, а также перечисляем предполагаемую функциональность системы в виде кратких сценариев взаимодействия. В дополнение к этому проводится анализ потребностей и контекста работы целевой аудитории, которая описывается в виде ключевых персонажей. Также составляется первоначальная карта сайта, которая показывает примерную структуру будущей системы. На написание и утверждение этих базовых документов обычно уходит около 3 дней, после чего мы планируем остальные работы и даем точную оценку сроков и стоимости их выполнения.

Этап II. Сбор требований

На следующем этапе мы готовим подробный перечень функциональности (user stories). Он позволяет учесть все функциональные требования и лучше понять особенности будущей системы. На его основе мы делаем вывод, какие из функций требуют целого процесса, какие — просто отдельной страницы, а каким будет достаточно простой кнопки. Ориентируясь на составленных ранее персонажей, мы обновляем карту сайта и составляем схему навигации. После этого рисуются диаграммы переходов между страницами — они объединяют страницы системы в рамках конкретных процессов. Теперь мы знаем, как пользователи будут работать с продуктом в целом и как именно выполнять конкретные задачи.

Этап III. Проектирование интерфейса

Третий этап — самый важный. Здесь создаются структурные схемы страниц (wireframes), которые показывают, какая информация и элементы управления должны располагаться на страницах системы.

Это еще не дизайн, но уже основа для него — wireframes являются техническим заданием для дизайнера. Общение с клиентом на этом этапе достаточно плотное — уточнение вопросов и утверждение чертежей идет по нескольку раз в день. Но и результатов хватает — в зависимости от сложности проекта выходит от нескольких десятков до пары сотен схем страниц.

Этап IV. Дизайн интерфейса

Завершающим этапом становится визуальный дизайн интерфейса. Сперва на основе пары ключевых страниц отрабатывается креативная концепцию. После того как общая стилистика одобрена клиентом, отрисовываются дизайн-макеты ключевых страниц системы. На этом этапе продукт обретает внешний вид.

Для проектов, которые планируют активно развиваться, готовится руководство по стилю интерфейса (style guide). Он описывает принципы визуального оформления продукта и позволит сохранить его целостность в процессе доработок.

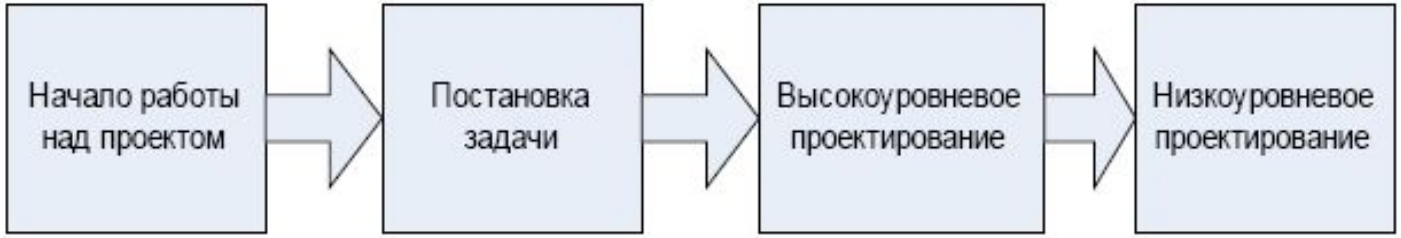
Этап V. Подготовка спецификации

Готовится предварительное техническое задание на разработку системы. Оно объединяет в себе полученные ранее документы, расширяет и перечисляет дополнительные требования к системе — функциональные, архитектурные, эксплуатационные. По желанию могут быть составлены подробные сценарии взаимодействия, которые пошагово описывают процесс работы пользователя с системой.

Финальный этап. Приемка

Приемка работ может проходить одним большим пакетом замечаний или разбиваться на несколько более мелких этапов. Сроки, в которые замечания должны быть выставлены, оценены и исправлены оговариваются в договоре.

Этапы проектирования



Проектирование опыта взаимодействия или проектирование взаимодействия (UX design) — это новая область научно-практической деятельности, которая в последние годы выделяется как самостоятельная дисциплина, сосредоточенная на проектировании поведения пользователя продукта.

Проектирование взаимодействия — это описание возможного поведения пользователя и определение того, как система будет реагировать на его поведение, и приспосабливаться к нему.

Проектирование взаимодействия касается не столько эстетических аспектов, сколько понимания потребностей пользователей и принципов их познавательной деятельности. Форма и эстетическая привлекательность продукта должны работать в гармоничной связке при достижении целей пользователей посредством правильно спроектированного поведения продукта.

Начало работ над проектом

На этом этапе определяются объёмы работ, планируются затраты и т. п. Длительность этого этапа, как правило, не превышает 5-8% от общего времени разработки. Для адекватной оценки ресурсов (времени, денег, количества специалистов) требуемых для разработки (переработки) интерфейса, необходимо хорошо представлять себе объём информации, с которой следует ознакомиться. К ней относится информация о предметной области и прототипах. Она получается из литературных источников и опросов экспертов. Результатом этой работы является количественная оценка ресурсе ёмкости проекта.

Чтобы предлагать адекватные интерфейсные решения, необходимо иметь ясное представление о предметной области системы. Предметная область изучается по литературе, кроме того, весьма полезны беседы с опытными пользователями, другими сотрудниками (экспертами) для выяснения всех деталей и характеристик предметной области. Вместе с «жалобами» заказчика на текущую версию системы результаты этого этапа составляют основное содержание работы над проектом (экспертная оценка часто обнаруживает проблемы, которые заказчику не видны, маскируясь под другие). Проблемы, выявленные на данном этапе, должны быть решены в новом интерфейсе. Удачные решения желательно сохранить, чтобы имеющимся пользователям не пришлось переучиваться (и чтобы сократить затраты на переделку).

Цели проекта

Хорошо сформулированная цель должна быть:

- **Понятной**. Избегайте использования узкоспециализированной терминологии.
- **Ясной**. Избегайте туманных формулировок; подбирайте выражения, которые были бы уместными при определении приоритетов требований.
- **Измеримой**. Используйте конкретные утверждения, которые можно проверить независимо, чтобы определить степень успешности проекта.

Расплывчатые, неформальные цели могут дать представление о желаниях клиентов и о более масштабных целях. Они создают основу для формулирования более четких проектных целей, например:

- повысить доход от интернет- продаж на 10%;
- повысить доход от интернет- рекламы на 20%;
- увеличить количество текущих и потенциальных пользователей в нашей базе данных как минимум до 20 000;
- предоставить основной группе пользователей качественный и популярный контент (от вас потребуются дополнительные усилия, чтобы конкретизировать способ измерения уровня «качества» и «популярности», но структурные элементы уже готовы).

Постановка задачи. Сбор информации о разрабатываемом продукте

На этой стадии анализируют данные о пользователях, формализуется функциональность и определяются критерии оценки проекта.

Залогом успешного проектирования пользовательского интерфейса является наличие наиболее полной информации об аудитории пользователей: их целях, задачах, предпочтениях, привычках и представлениях, и о заказчиках. Чем более полная информация о продукте будет собрана и передана проектировщикам, тем более чёткое и правильное представление о его качествах будет у них сформировано и, соответственно, тем эффективнее будет проходить процесс разработки на всех последующих стадиях.

Данные о пользователях и о проекте должны содержать следующие позиции:

- характеристики пользователей: их опыт работы с компьютером, знание предметной области, мотивы, размер важность групп пользователей, примеры (типовые ситуации) использования;
- цели и задачи пользователей;
- задачи проекта: что послужило причиной создания проекта, этапы создания проекта, какие результаты должны быть получены, какая информация необходима и когда;
- технология разработки и платформа, на которой будут работать пользователи;

Эта работа предполагает доступ к имеющимся и потенциальным пользователям системы, экспертам и проектной документации. На этом этапе разрабатываются пользовательские профили, модели пользователей. Обязательно должна присутствовать информация о субъективных ожиданиях пользователей системы. Без этого трудно или невозможно предугадать отношение пользователей к будущей системе.

Формализация действий пользователей начинается с описания различных типовых сценариев. Для этого формализуются данные, необходимые пользователям для выполнения работы, последовательность самой работы, критерии завершённости этой работы. В результате должно появиться словесное описание взаимодействия пользователя с системой, не конкретизируя, как именно проходит взаимодействие, но уделяя возможно большее внимание всем целям пользователей. Количество сценариев может быть произвольным, главное, что они должны включать все типы задач, стоящих перед системой, и быть сколько-нибудь реалистичными. Сценарии очень удобно различать по имена существующих в них вымышленных персонажей.

Исследование целевой аудитории

Оценивать результат проектирования в конечном итоге следует исходя из того, насколько успешно он отвечает требованиям пользователей и компании — инициатора разработки. Если у проектировщика нет ясного представления о пользователях, для которых выполняется проектирование, если у него отсутствует понимание имеющихся ограничении, организационных задач и бизнес целей, которые являются движущей силой разработки, то шансов на хороший результат очень мало — неважно, насколько при этом хороши навыки и творческие способности проектировщика.

Наиболее популярные методы сбора данных используют количественные методы. Эта группа методов даёт ответ на вопрос «Сколько?», а информация, получаемая в результате применения количественных методов, обрабатывается с использованием статистических методов анализа. Типичный результат использования количественных методов сбора информации — получение процентного распределения, какая часть выборки потребляет тот или иной продукт, знает данную марку, алгоритмы использования, свойства продукта и т. п.

Качественные исследования

Качественные методы сбора данных позволяют получить информацию, которая отвечает на вопрос «Почему?» и не может быть обработана при помощи статистических методов анализа.

Иными словами, из информации, собранной с использованием какого-либо из качественных методов, нельзя вывести какие-либо проценты и распределения, она позволяет лишь понять различные факторы и мотивы определенных действии, выявить модели поведения и т. п.

Качественные (неформализованные) методы ориентированы не на массовый сбор данных, а на достижение углубленного понимания исследуемых явлении. Отсутствие формализации делает невозможным массовый охват обследуемых объектов, в результате чего число единиц обследования часто снижаются до минимума. Отказ от широты охвата компенсируется «глубиной» исследования, т. е. детальным изучением явления в его целостности и непосредственной взаимосвязи с другими явлениями.

Качественные исследования позволяют получить глубокую, развернутую информацию о предмете исследования. В отличие от количественных, качественные исследования фокусируются не на статистических измерениях, а опираются на понимание, объяснение и интерпретацию эмпирических данных п являются источником формирования гипотез и продуктивных идей.

Качественные исследования помогают понять предметную область, контекст и ограничения продукта, более действенным способом, чем количественные исследовании. Они также помогают выявить шаблоны поведения потенциальных пользователей продукта быстрее и проще по сравнению с количественными методами.

Качественные методы позволяют изучить:

- поведение, взгляды, склонности потенциальных пользователей продукта;
- предметную область — технический, экологический, н деловой контексты разрабатываемого продукта;
- используемый лексикон и прочие социальные аспекты предметной области;
- способы применения существующих продуктов.

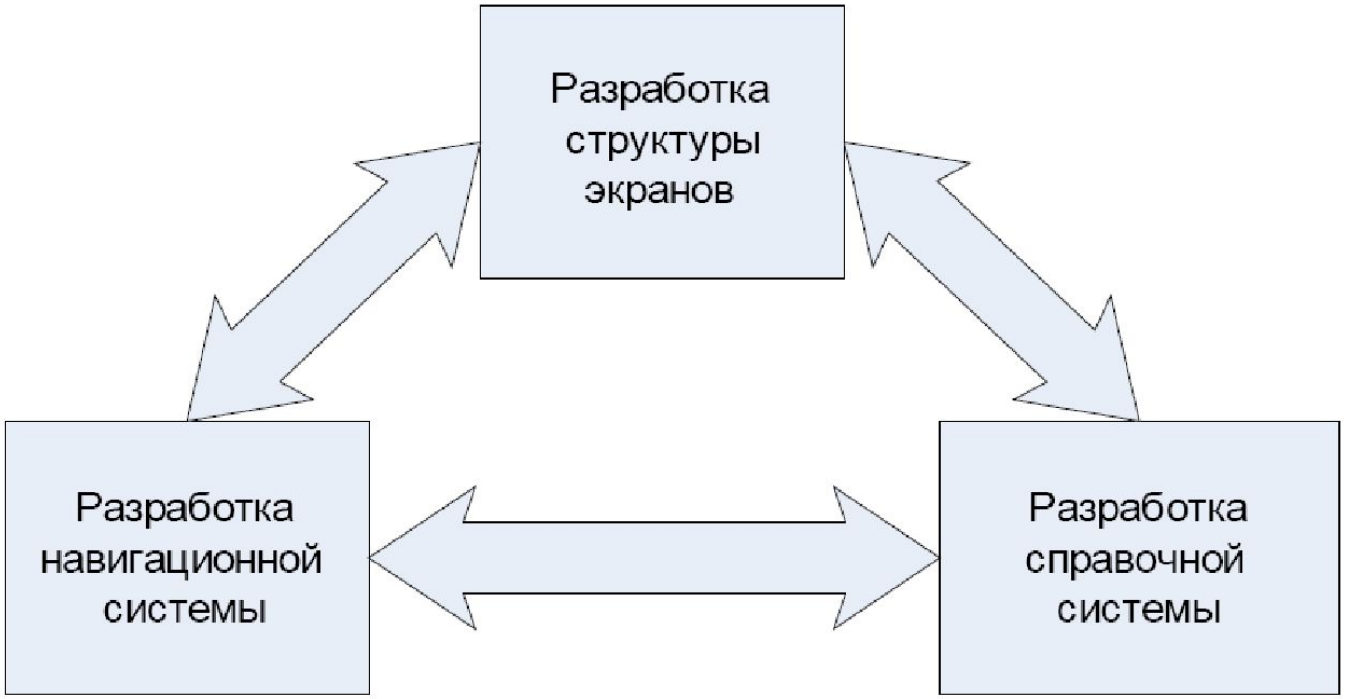
Качественные исследования способствуют ходу проектирования, поскольку:

- обеспечивают доверие н уважение к команде проектировщиков;
- объединяют команду общим для всех пониманием особенностей предметной области и проблем пользователей;
- дают руководителям возможность принимать решения по тем или иным вопросам проектирования продукта на основе данных — вместо догадок н личных предпочтений.

Высокоуровневое проектирование

Схематично работу на данном этапе можно представить, как показано на рисунке, в виде трёх взаимосвязанных блоков.

Основной задачей разработки является преобразование потребностей потенциальных пользователей и целей конечного продукта в конкретные требования к контенту и функциональности разрабатываемого продукта.



На этапе разработки набора возможностей следует выделить два основных направления:

1. Функциональная спецификация. Документ, содержащий требования по функциональности разрабатываемого продукта.
2. Требования к контенту.

Функциональная спецификация — документ, который заключает в себе информацию о том, как будет работать продукт с точки зрения пользователя.

Функциональная спецификация обычно содержит в себе:

1. Сценарии пользователей.
2. Краткий обзор разрабатываемого продукта.
3. Набор возможностей продукта.
4. Перечень того, что не должен содержать продукт.

Необходимо помнить о том, что функциональная спецификация должна быть доступна всем участникам процесса разработки, и по мере продвижения процесса, она может корректироваться в связи с выявлением новых значимых возможностей и особенностей разрабатываемого продукта.

Сценарии

Первым шагом при создании функциональной спецификации является написание **сценариев пользователей**.

Сценарий пользователя — краткое, простое описание того, как пользователь пытается удовлетворить потребности с помощью разрабатываемого продукта. Представив процедуру, через которую могли бы пройти потенциальные пользователи, можно более точно выработать требования к программному продукту.

Эффективность сценария определяется в большей степени его охватом, чем глубиной. Иначе говоря, важнее, чтобы сценарий описывал процесс от начата до конца, чем чтобы он описывал каждый шаг в исчерпывающих подробностях.

На основе выявленных **сценариев** работы осуществляется *разработка структуры экранов,* т. е. определяется количество экранов, функциональность каждого из них, навигационные связи между ними, формируется структура меню и других навигационных элементов.

По сути, на этом этапе выделяются отдельные функциональные блоки. Под функциональными блоками будем понимать функцию пли группу функций, связанных по назначению или области применения — в случае программы, и группу функции фрагментов информационного наполнения — в случае сайта.

Существует три основных вида связи между блоками:

1. логическая связь;
2. связь по представлению пользователей;
3. процессуальная связь.

Логическая связь определяет взаимодействие между фрагментами системы с точки зрения разработчика. Полученные связи очень существенно влияют на навигацию в пределах системы (особенно когда система многооконная).

Поэтому, чтобы не перегружать интерфейс, стоит избегать как слишком уж отдельных блоков (их трудно найти), так и блоков, связанных с большим количеством других.

На основе разработанной структуры экранов на этом этапе выбирается наиболее адекватная *навигационная система* и разрабатывается её детальный интерфейс.

Навигационная система показывает механизм распределения функций и задач между окнами программы.

Навигационная система основывается на информационной архитектуре и связана с созданием организационных и навигационных схем, обеспечивающих экономичное и эффективное перемещение, как между различными задачами, так и внутри отдельной задачи.

Информационная архитектура имеет прямое отношение к вопросам информационного поиска — проектированию систем, позволяющих пользователям легко находить нужную информацию. Однако архитектура web-сайтов, как отдельного вида информационных систем, часто призвана решать более широкие задачи, чем просто помощь в поиске информации: во многих случаях сайтам приходится обучать, информировать и убеждать пользователей.

Низкоуровневое проектирование

Низкоуровневое проектирование заключается в детальной проработке поставленных задач и в проверке качества разработанных решений (рис).

Концептуальная структура придает грубую форму массе требований, которые возникают из наших стратегических целей. На уровне компоновки проводится дальнейшее уточнение этой структуры, выделяя специфические аспекты дизайна интерфейса и навигации, а также информационного дизайна, которые сделают неосязаемую структуру вполне конкретной.

Этап разработки структуры определяет, как будет работать разрабатываемый продукт. На этапе компоновки определяется форма, которую примет эта функциональность. Кроме конкретизации представления информации, на этапе компоновки осуществляется переход к вопросам, принципиально требующим более глубокой детализации. Если на уровне структуры оперируют крупномасштабными понятиями архитектуры п взаимодействия, то на уровне компоновки рассмотрение сконцентрировано практически исключительно на отдельных страницах и составных частях.

При *проектировании основных экранов* производится полное описание их интерфейса (без обработки исключительных ситуаций), организация информации на экранах. К отчёту прилагаются макеты экранов с описаниями функциональности каждого интерфейсного элемента. Разрабатывается презентационный или псевдореальный прототип ПИ, а в конце этого этапа прототип вполне может быть и реальным. При юзабилити-тестировании на основе критериев оценки ПИ и сценариев действий пользователей разрабатываются тестовые задания, которые выполняются пользователями на прототипе с фиксацией всех значимых характеристик деятельности (таких, как производительность труда, количество человеческих ошибок).

После этого выполняется подсчет соответствующих показателей и сравнение их с заданными (или желаемыми). На основании полученных данных интерфейс либо дорабатывается, либо считается разработанным. К *второстепенным экранам* относятся диалоговые окна и всевозможные сообщения. Их интерфейс полностью описывается, равно как описываются п исключительные ситуации, влияющие на интерфейс. При финальном юзабплн-тп-тестпровании на какой-то версии прототипа разрабатываются п выполняются тестовые задания, оставшиеся после предварительного тестирования. На основании полученных данных интерфейс либо дорабатывается, либо считается разработанным, т. е. это итерационная процедура (как п юзабплптп-тестирование в целом).

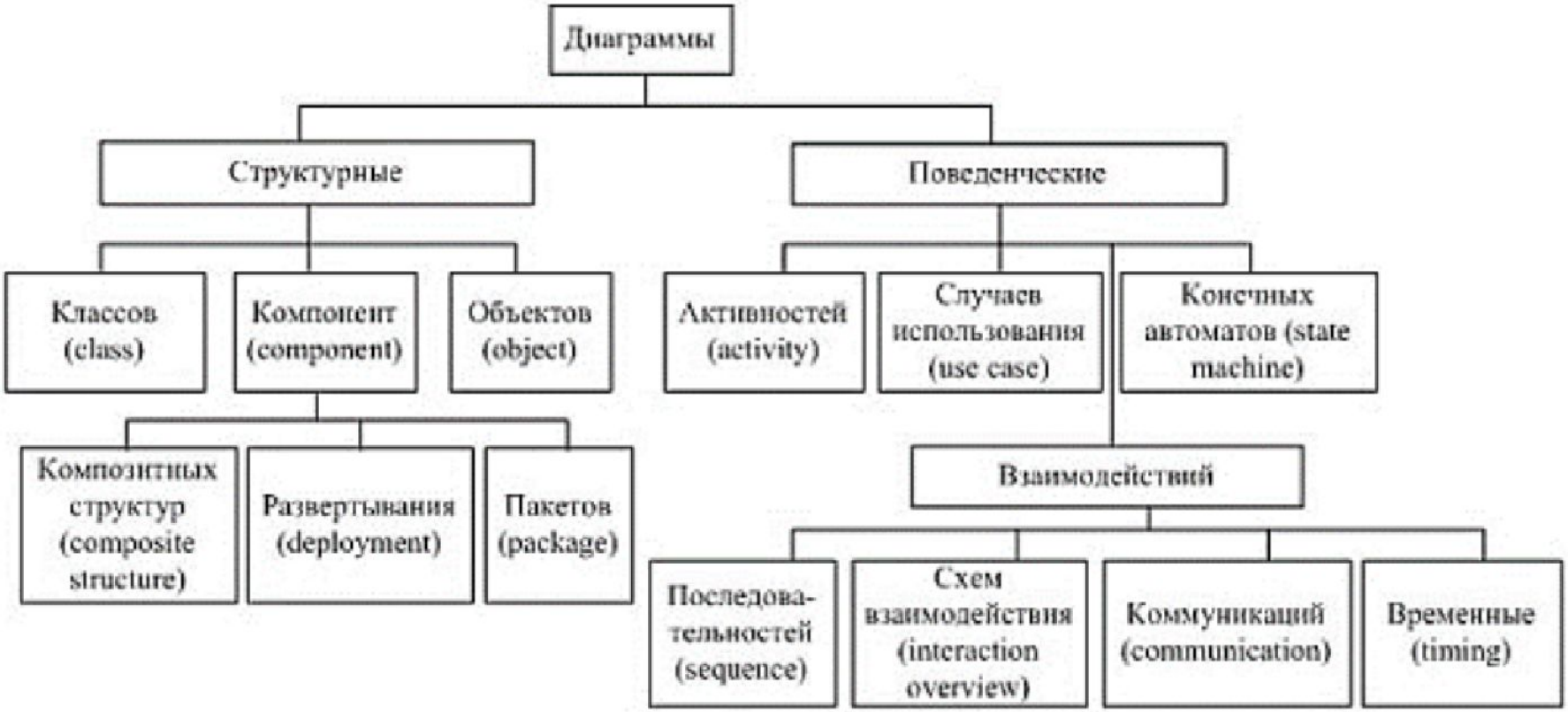
Субъекты, заинтересованные в реализации проекта

Стейкхолдеры это люди, которые будут использовать готовую систему – пользователи, люди, которые будут непосредственно работать над созданием системы – команда разработчиков, а также люди, формирующие бюджет проекта и, поэтому особенно заинтересованы в успешном завершении проекта.

- ***Пользователи.*** **Э**та группа, состоит из людей, которые являются актерами в диаграммах USE-Case.
- ***Спонсоры.*** Сюда можно отнести менеджеров, финансистов, акционеров, руководителей департаментов, специалистов по маркетингу и других персон, которые инвестируют средства в изготовление изделия.
- ***Разработчики.*** Люди, непосредственно принимающие участие в разработке и сопровождении системы: менеджеры проекта, тестировщики, служба поддержки, проектировщики, кодировщики, разработчики технической документации, производственная служба и другие разработчики.
- ***Администрация.*** Эксперты в практических аспектах проблемы или области решения. Это могут быть представители законодательных организации, организаций по стандартизации, правительственных служб и департаментов, внутренних и внешних регулятивных органов, специалисты в определенной области знаний, эксперты в определенной области технологий.
- ***Покупатели.*** Люди или организации, которые приобретают готовую систему. Эта группа может включать покупателей, экспертов по оценке, бухгалтеров, агентов, действующих в интересах покупателя.

Unified Modeling Language

Представляет собой универсальный язык моделирования для использования во всех областях человеческой деятельности. Принят в качестве международных стандартов ISO/IEC 19501:2005 (для версии языка 1.4.2) и ISO/IEC 19505-1, 19505-2 (для версии языка 2.4.1). Универсальность языка объясняется наличием большого количества различных типов моделей, позволяющих описывать как структурные, так и поведенческие аспекты системы (рисунок). Поведенческие модели построены на базе концепции сетей



Семейство инструментов на базе языка UML. Унифицированный язык моделирования (UML), наиболее популярный язык спецификаций, применяется не только для разработки структур приложений, поведения и архитектуры, но и моделирования бизнес процессов и структур данных. UML как метод используется для изучения поведения систем; UML как язык используется для "вычленения" знаний о предметной области; UML как моделирующий язык используется для понимания (и, возможно, формализации) закономерностей функционирования систем; UML как унифицированный язык используется для координации деятельности разработчиков.

Огромное количество средств проектирования и автоматической генерации кодов программ из UML спецификаций объясняется популярностью унифицированного языка моделировании и универсальностью UML.

В инструментариях UML представленных на сайте Международного консорциума производства средств вычислительной техники OMG , каждый использует в качестве метода моделирования язык UML, SDL.

Многие обеспечивают поддержку обратного проектирования, то есть построение документации по имеющемуся коду на языках PHP,C++ , ANSI C , Perl, Java , C#, CORBA IDL , Delphi , ADA, Eiffel.

Средства поддержки проектирования интерфейсов

Такие средства должны обеспечивать:

- поддержку проектирования и реализации различных типов диалога;
- инструментальную поддержку проектирования и автоматическую генерация кода всех компонентов интерфейса;
- раздельную разработку и модификацию интерфейса и прикладной программы с последующим их связыванием;
- поддержку раздельной модификации различных компонентов пользовательского интерфейса;
- повторную используемость отдельных компонентов пользовательского интерфейса;
- поддержку оценивания проекта интерфейса;
- интеллектуальную поддержку разработчика, освобождающую его от изучения новых языков.

Существующие на рынке инструментальные средства, как правило, поддерживают диалог на основе экранных форм (с использованием технологии WYSIWYG (*What You See Is What You Get*) и автоматической генерацией кода); ни одно средство не поддерживает проектирование и реализацию различных типов диалога.

Интегрированные пакеты, в состав которых входят Построители интерфейсов, поддерживают проектирование, реализацию и повторное использование только визуального компонента интерфейса. Реализация остальных компонентов интерфейса осуществляется на языке программирования интегрированного пакета разработки – C++, Паскаль и др.

Остальным требованиям средства данной группы не удовлетворяют.

Системы управления пользовательским интерфейсом (***User Interface Management Systems***) поддерживают спецификацию интерфейса (каждая на своем языке) и автоматическую генерацию по ней некоторых компонентов интерфейса, а также раздельное проектирование пользовательского интерфейса и прикладной программы, однако, остальным требованиям инструменты данной группы не удовлетворяют.

Средства разработки интерфейса, основанные на моделях (***Model-Based Interface Development Environment***), поддерживают проектирование, модификацию, повторное использование и автоматическую генерацию кода всех компонентов интерфейса, а также раздельное проектирование и модификацию интерфейса и прикладной программы. Однако, они требуют от разработчика изучения либо специальных декларативных языков для описания компонентов интерфейса (например, MIMIC), либо предлагают ему несколько известных формализмов (CORBA IDL, CTT, UML, Petri-net и др.) для описания таких компонентов, что снижает интеллектуальную поддержку разработчика.

Некоторые средства анализируют модели интерфейса.

Моделеориентированные средства не удовлетворяют требованию «открытости», поэтому, к тому времени, как такое средство выходит на рынок, оно требует модификации, по этой же причине устаревают и средства анализа.

Разработка ПИ на основе онтологий

Новым является определение модели пользовательского интерфейса, которая содержит лишь ту информацию, которая может подвергнуться изменению в его жизненном цикле. Проектирование интерфейса связано с четырьмя основными классами систем понятий:

- системой понятий пользователя, в терминах которой он осуществляет свое взаимодействие с прикладной программой;
- одной из систем понятий, в терминах которой определяются различные типы диалога;
- системой понятий для определения сценарий диалога;
- системой понятий, в терминах которой осуществляется связь между прикладной программой и пользовательским интерфейсом.

Таким образом, модель интерфейса содержит четыре компоненты, каждая из которых определяется в терминах одной из этих систем понятий. В терминах этих же систем понятий обеспечивается интеллектуальная поддержка разработчика с использованием структурных и графических редакторов, а также определяются связи между компонентами.

Для обеспечения поддержки проектирования и реализации различных типов диалога разработчику интерфейса предлагаются три системы понятий, каждая из которых поддерживает проектирование одного из типов диалога – система понятий графического пользовательского интерфейса, поддерживающего разработку интерфейсов, основанных на формах или WIMP-интерфейсов (windows, icons, menus, pointing devices), система понятий графических статических сцен, и система понятий для формирования текстов (в дальнейшем количество систем таких понятий предполагается увеличить).

Методологии

Большинство методологий состоят из одних и тех же этапов:

- **Планирование** общей стратегии, подхода и структуры команды.
- **Формулирование** требований к проекту.
- **Проектирование** взаимодействия и визуальной концепции и их развитие до уровня подробных спецификаций.
- **Разработка**, тестирование и совершенствование решений.
- **Развертывание** разработанного продукта с использованием различных средств коммуникаций, проведение обучения и запуск в оговоренные сроки.
- **Сопровождение** проекта на основе рекомендаций по улучшению.

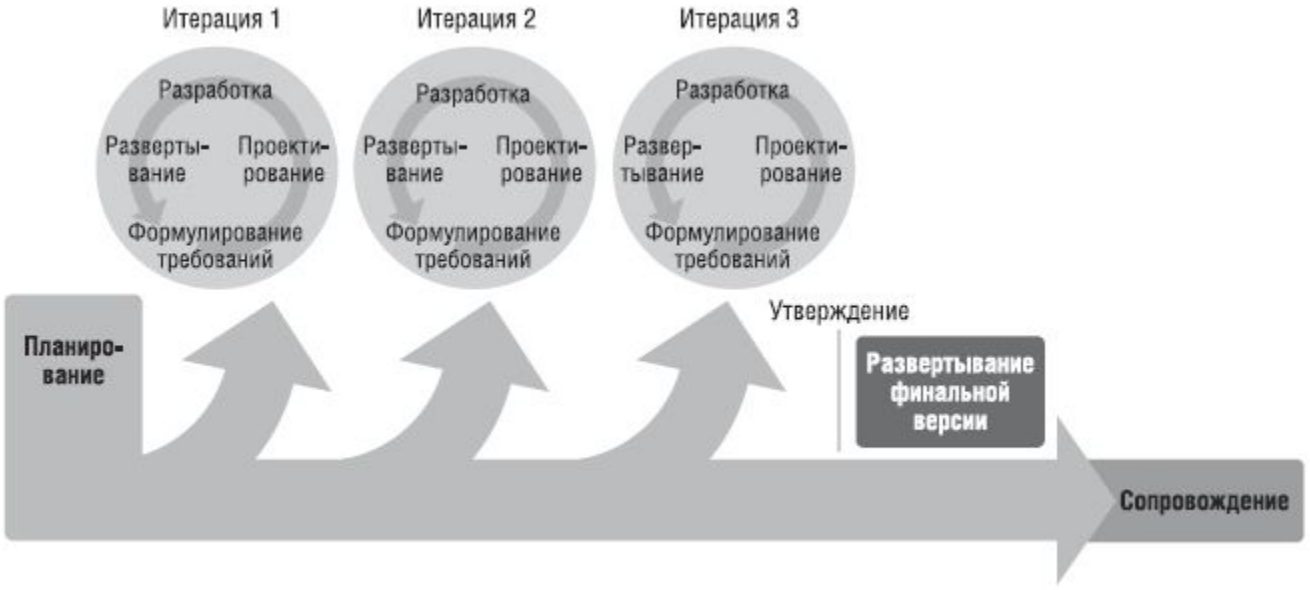
Каскадная методология

В *каскадной* (*waterfall*) *методологии* этапы проекта рассматриваются как отдельные *фазы*, причем следующая фаза начинается только после утверждения предыдущей.

Например, фаза проектирования реально начинается только после того как требования будут утверждены представителями бизнеса, которые подписывают документы с требованиями в конце фазы формулирования требований. Недостаток «чистой» каскадной методологии состоит в допущении, что завершение каждой следующей фазы требует минимальных изменений в результатах предшествующей фазы. Таким образом, если в фазе проектирования вдруг появятся новые требования (а это вполне обычное явление), вам придется вносить изменения в документы, утвержденные в конце фазы формулирования требований. Это может привести к нарушениям планов и гр

Гибкие методологии

Поскольку постоянны только изменения, проектные команды находятся в постоянном поиске методологий, превосходящих по гибкости каскадную модель. Во многих методологиях используется более гибкий подход, когда некоторые этапы выполняются параллельно, например версии веб-сайта публикуются по ускоренному итеративному графику с использованием ***гибких*** (***agile***) или ***быстрых*** (***rapid***) методологий. Гибкие методологии обычно сильнее ориентированы на оперативную совместную работу и в меньшей степени требуют подробного документирования и формальных утверждений.



Как выбор методологии влияет на работу

Знание методологии, выбранной для проекта, помогает понять ряд вещей:

- **Какие вопросы и когда следует задавать.** Например, при работе по «чистой» каскадной методологии вам придется приложить дополнительные усилия к тому, чтобы в требованиях, созданных в фазе формулирования требований, содержалась вся информация, необходимая для фазы проектирования.
- **Как следует организовать совместную работу членов проектной команды и насколько тесным должно быть их взаимодействие**. Например, гибкая методология требует очень тесного сотрудничества, а в каскадных методологиях участники в основном работают сами по себе и вступают в контакт друг с другом один или несколько раз в неделю.
- **Насколько подробной и формальной должна быть документация.** Документы, предоставляемые на утверждение, должны быть формальными – почти как юридические договоры. Каскадные методологии, в которых переход к следующей фазе возможен только после утверждения результатов предыдущей, обычно требуют более формальной документации. Однако формальные документы могут потребоваться и в гибких методологиях, например для фиксирования информации в главных точках принятия решений .
- **Важнейшие контрольные точки, включающие в себя утверждение результатов представителями бизнеса и развертывание для разных групп пользователей.** Методология предопределяет, что именно участники должны совершать в определенных точках проекта: владельцы проекта

Структура среды информационной системы

Обобщенная структура любой ИС представляется состоящей из двух взаимодействующих частей:

- функциональной части, включающей прикладные программы, которые реализуют функции прикладной области;
- среды или системной части, обеспечивающей исполнение прикладных программ.

С этим разделением тесно связаны две группы вопросов стандартизации:

1. стандарты интерфейсов взаимодействия прикладных программ со средой ИС (Application Program Interface - API);
2. стандарты интерфейсов взаимодействия самой ИС с внешней для нее средой (External Environment Interface - EEI).

Эти две группы интерфейсов определяют спецификации внешнего описания среды ИС - архитектуру с точки зрения конечного пользователя, проектировщика ИС, прикладного программиста, разрабатывающего функциональные части ИС.

Спецификации внешних интерфейсов среды ИС и, как будет видно далее, спецификации интерфейсов взаимодействия между компонентами самой среды, - это точные описания всех необходимых функций, служб и форматов определенного интерфейса. Совокупность таких описаний составляет модель открытых систем (Reference model).