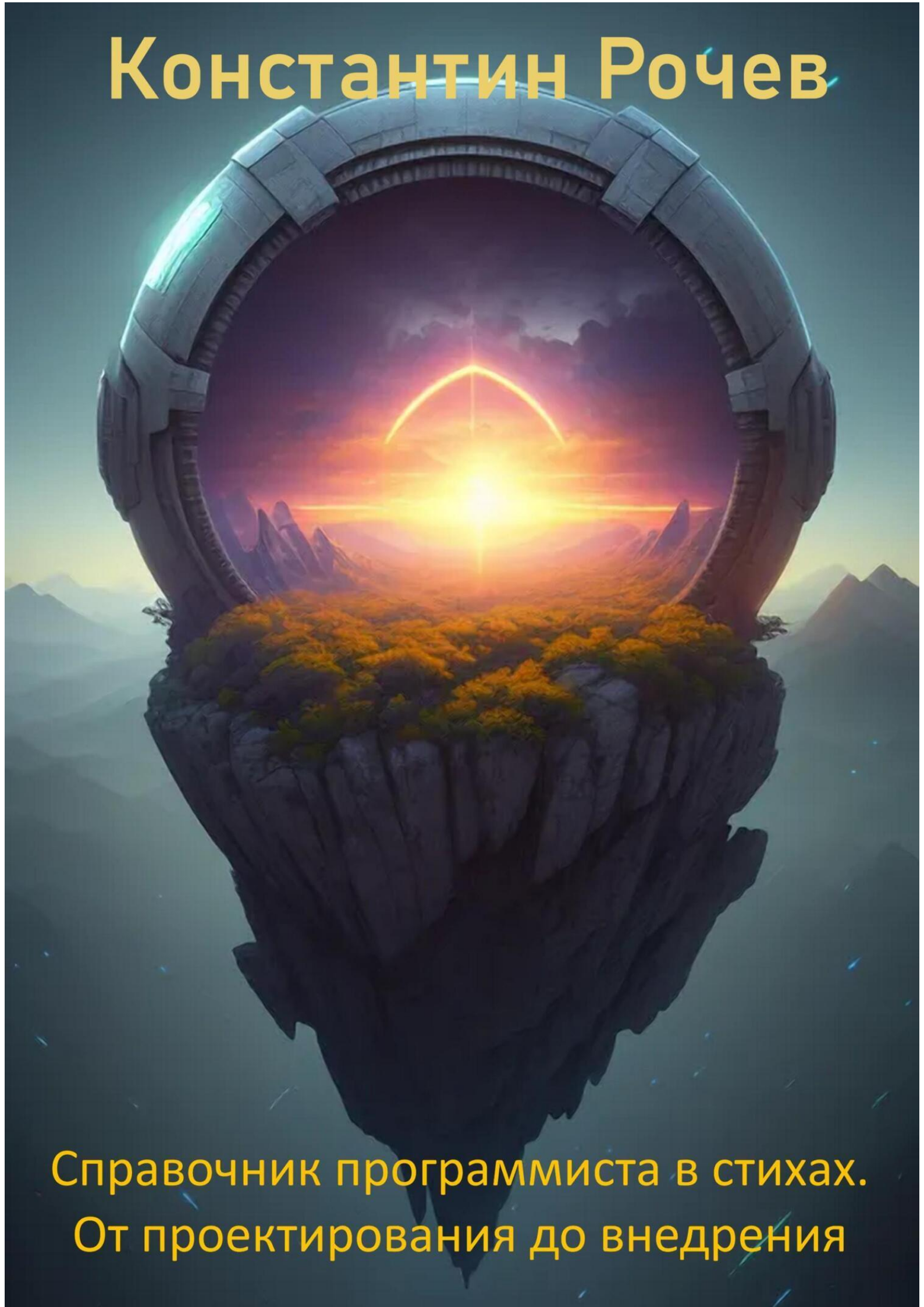
Константин Рочев



Справочник программиста в стихах. От проектирования до внедрения

Константин Рочев

Справочник программиста в стихах. От проектирования до внедрения

«Автор»
2023

Рочев К. В.

Справочник программиста в стихах. От проектирования до внедрения / К. В. Рочев — «Автор», 2023

ISBN 978-5-532-91179-6

В этом сборнике-справочнике изложены особенности всего цикла разработки программ от проектирования до внедрения. Каждому существенному определению или принципу разработки в этой книге посвящено по несколько четверостиший.Здесь представлены рекомендации по анализу и постановке требований, управлению жизненным циклом проекта, проектированию архитектуры и разработке программных систем. Описаны некоторые базовые алгоритмы, типовые подходы и решения к написанию кода, перечислены некоторые программные технологии и стеки. Приведено описание окружения, в котором программное обеспечение будет выполняться: от принципов построения и работы операционных систем до особенностей технических средств и оборудования.По мнению литературоведов чтение и даже прослушивание стихов вызывает состояние близкое к альфа-ритмам сна, в которых информация запоминается лучше всего. За счет этого, такая книга может помочь легче погрузиться в ИТ-сферу, изучить или вспомнить принципы программирования.

ISBN 978-5-532-91179-6

Содержание

Константин Рочев
Справочник программиста в стихах. От проектирования до внедрения

Предисловие от автора

Программировать я начал ещё с момента появления у меня первого компьютера – около 20 лет назад, а профессионально занимаюсь программированием около 15 лет. За это время участвовал в разработке около сотни, программных проектов. Работал один и в команде, в офисах и дистанционно, писал программы от научных приложений до корпоративных систем, от небольших мобильных приложений до MMORPG игр, простирающихся на пол миллиона строк кода. Поэтому хочется верить в то, что, имею какой-то скромный опыт, которым попробую поделиться с Вами, уважаемый читатель, на страницах этого произведения. Каждая область разработки довольно уникальна, и поэтому основную часть книги составляет обобщенный краткий обзор общепризнанных методик, подходов и алгоритмов лишь с небольшим наслоением авторского мнения.

Что касается стихов, их я, как и все, писать начал еще в школе или раньше, но более осмысленно погрузился в это направление в поэтическом клубе Ухтинского университета во время учебы и после. В 2012-2013гг. провёл проект по ежедневному написанию стихов в течение года. И вот, спустя 10 лет, повторяю это действо в более целенаправленном формате, написав за год сборник стихов о программировании. Литературоведы говорят, что чтение стихов вводит мозг в состояние близкое к состоянию сна, благодаря чему информация усваивается гораздо легче. Поэтому желаю Вам приятного погружения в мир проектирования и разработки программных систем на волнах поэтических альфа-ритмов.

Константин Рочев

Рецензии

Рецензия А. Р. Эмексузяна

«Vita brevis, ars longa» – «Жизнь коротка, искусство вечно!»

Поделиться своим мнением на учебное или методическое пособие или учебник для человека, работающего в образовании, дело не кажется чем-то незаурядным, в конце концов, это даже некая составляющая работы и скорее всего даже незаметная ее часть – рутина.

Все обстоит иначе с произведением, которое вы, уважаемый читатель, держите в руках. Это не просто Справочник программиста от проектирования до внедрения, а учебное пособие, написанное в СТИХАХ! Я выделил слово “стихах”, т.к. практически уверен, что это первое (ну, или совсем немногое в своем роде и точно мне неизвестное до сих пор) подобное по форме изложения учебное издание, а если еще и учесть, что оно посвящено программированию, уверен, что данный справочник – уникален! И у вас, дорогой читатель, есть прекрасный шанс изучить и, думаю, местами насладиться столь необычным справочником!

Первая мысль, которая пришла еще до написания данной рецензии, даже порыв – также изложить все в стихотворной форме, дабы не быть оригинальным по сравнению с автором. Однако, будучи человеком далеким от знатока поэзии, хотя и уважающим хорошее произведение, данный порыв был не более, чем молниеносным желанием из области несбыточных мечтаний.

Желание не отстать от автора в оригинальности подачи материала породило тут же другую мысль на месте сгоревшей дотла первой – а что, если написать рецензию на данный справочник в стихах с помощью уже нашумевшего помощника – искусственного интеллекта в лице или в виде чата – ChatGPT. Однако даже несколько попыток создать что-то поэтическое и наполненное смыслом и отзывом о справочнике, не увенчались собственным обывательским поэтическим удовлетворением.

Талантливый человек, талантлив во всем – это выражение, принадлежащее немецкому писателю Лиону Фейхтвангеру, еще раз утвердилось в моей голове после прочтения замечательного справочника, который вы держите в руках. Я всегда знал и знаю автора, как умнейшего и талантливейшего разработчика, программиста, руководителя команды разработки, проекта, но талант, который предстал передо мной на страницах этого справочника, раскрыл совершенно другую сторону автора, я бы сказал – гуманитарное полушарие его гениального мозга. Гениальность тоже можно назвать талантом, т.к. это уникальная особенность человека быть умным, рассудительным и находить простое объяснение сложным вещам… к тому же в стихах!!!

При погружении в чтение стихотворение за стихотворением, мое любопытство и чувство изумления формой и сутью прочитанного постепенно перерастало в практическое удовлетворение. По сути своей нынешней работы мне часто приходится сталкиваться, как оказалось, со многим, что изложено в данном справочнике, например, влёт запомнилось следующее:

Для выполнения проекта
С известным качеством и сроком
Весьма полезным документом
ТЗ является. Во многом
Его задача – однозначность
При понимании системы.
В ТЗ описаны задачи

Проекта так, чтоб были всеми
Они восприняты в едином
Ключе и смысле, и трактовок
Различных не было в картине
И описании разработок.

Это просто гениальное, по мне, стихотворение о сути ТЗ (технического задания), которого часто не хватает в практическом понимании как самому заказчику, так и исполнителю, пишу с полным пониманием и ответственностью. Теперь каждый раз, при обсуждении ТЗ какого-либо проекта или контракта, в голове моей звучат вышеприведенные строки, а для нашего департамента это вообще стало неким девизом, распечатанным и помещенным на видное место для всех. Могу только посоветовать автору развить ТЗ до ЧТЗ (частное техническое задание), более детально прописывающее все необходимые требования к проекту.

Вообще весь раздел справочника, посвященный IT-проектированию, стал бесценным для меня, как человека далекого от программирования, но по сути сегодняшнего дня постоянно занимающегося в Минцифры России развитием и эксплуатацией большого модуля одной из федеральных государственных систем. Материал, в данном разделе помог до конца понять суть некоторых терминов и подходов в проектировании задач развития или эксплуатации любых систем, к тому же стихотворная форма подачи материала во многом облегчает запоминание сути изучаемых определений. Особенно понравилось про UX, Agile, бизнес-логику.

Не менее полезным оказались и другие части справочника: Инструменты программиста, Операционные системы, Обзор основного аппаратного обеспечения ПК. А вот послесловие автора можно с полной уверенностью адресовать любому читателю, даже не обязательно из IT-сферы:

Работа в ИТ – постоянное изучение нового
Область ИТ и разработки –
Весьма тяжёлая дорога.
К ней перейдя без подготовки
Легко остаться понемногу
За бортом новых технологий.
Чтобы успеть за их развитием
И оставаться на вершине,
Все время нужно ряд усилий
Предпринимать по изучению
Подходов разных и методик,
Фреймворков, окружений,
И прочих новых технологий.

Заменив буквы “ИТ" в вышеприведенном стихотворении на любую другую сферу, вы получите, пожалуй, абсолютно актуальный, точный и характерный современной жизни и ритму девиз!

Жизнь задумана как восхождение – остается только порадоваться за нашего автора и моего уважаемого друга за столь уникальное и очень полезное для широкого круга читателей (не только для тех, кто посвятил себя сфере IT) произведение и пожелать дальнейших успехов в подобных начинаниях!

Аркадий Рубикович Эмексузян

Рецензия А. В. Затонского

По жизни нет печальней книги,
Чем си плюс плюса толстый том,
Где Страуструповские сдвиги
Всё ходят по цепи кругом.
Запомнить даже не надейся
#инклюд эстэ… эстэдио.
Вот поработаешь лет десять,
И станешь сишный патриот,
Слова мутируют в аксоны,
Вот ты уже почти сенсэй –
В итоге изучил жаргоны…
Но ты ведь хочешь побыстрей?
В стихах слова, одно к другому,
Влетают в ухо, тормозят,
И, осмотревшись в новом доме,
Обратно шмыгнуть не грозят.
Заснуть, наверно, невозможно
Под лекции ритмичный строй,
Без «э» да «ну» пустопорожних,
Чеканной медью отлитой.
Осмыслишь вмиг лабораторку,
Призывным стихом увлечён,
Проект – рванёт, ПО – на сборку,
Баг – обнаружен, погребён,
Код даже горе-обалдую
Понятен с первых же минут…
Конечно, я рекомендую
К прочтенью этот славный труд!

Андрей Владимирович Затонский

Введение

Есть множество различных знаний,
Для понимания АйТи –
При разработке и создании
Программ их все не обойти.

Этапы разные в процессе
Реализации систем.
Давайте их посмотрим вместе
В обзорном виде и затем

Подробнее в архитектуру
И разработку завернём –
Устройство кода и структуру
Изучим, после перейдём

К обзору ряда инструментов
И окружения программ:
Сред разработки, компонентов,
Фреймворков, диаграмм,

Платформ – до материальных.
Всего того, что нужно знать
В задачах профессиональных,
Чтобы системы создавать.

Часть 1. Проектирование

Глава 1. Системы и их свойства

Подходим к автоматизации с умом
Проектируя систему,
Нужно думать, в основном,
Как она решит проблему,
Для заказчика. Потом,

После выбора задачи,
Нахождения путей,
Как её решить, в придачу,
Надо разобраться в ней.

Да не плохо изучить бы,
Как проходит весь процесс,
Чтоб ему не навредить бы.
Мозг здесь нужен позарез…

По аналогии
Что мы желаем отыскать
В исходных кодах мироздания,
Копаясь снова и опять
В остатках недопонимания?
Как слой за слоем познаём
Глубины общего устройства,
Примерно также создаём
Свои системы и их свойства.

Системный подход
Устройство, логика задачи,
Взаимосвязи и границы –
Всё это надо обозначить,
Учесть, чтобы могла внедриться

Без лишних сложностей программа.
И чтобы были эффективны
При разработке все этапы
Всё спроектировать должны мы.

Для понимания процесса,
Который хочется улучшить
За счёт внедрения системы,
Подход системный будет нужен.

Система
Система – группа элементов,
Что связаны между собой.

Объединяясь с общей целью
И с точки зрения любой.

Подсистема
Подсистемой называют
Часть системы, что, при этом,
Некой целью обладает,
Как отдельная система.

Элемент системы
Элемент – предел членения
Для системы с точки зрения
И аспекта рассмотрения
Цели и ее решения.

Закон необходимого разнообразия
Разнообразие системы,
Коя задачу выполняет,
Должно не меньше быть проблемы,
Которую она решает.

Закономерности систем
Целостность
При целостности элементы,
Объединенные в системе,
Часть свойств теряют перманентно,
С явлением новых в тоже время.

Иерархичность
Иерархичность означает,
Что каждый элемент системы
Собой систему представляет
На новом шаге рассмотрения.

Интегративность
Интегративностью системы
Обычно связи называют
Внутри системы, что сильнее
Тех, кои внешними считают.

Свойства систем
Все свойства всяческих систем
Подразделяют на две группы.
Функциональные – им всем
Присуще выполнение функций –

Задач системы – ради них
Система создана обычно.
Другая группа говорит,

Как выполняться им прилично.

И нефункциональных свойств
Набор проявится во время
Взаимодействия устройств
Системы с ближним окружением.

Верифицируемость
Верифицируемость – это
Способность оценить корректность
Системы. И обычно тесты,
При этом, основное средство.

Безопасность
Суть «безопасности», как свойства,
В том, что система при работе
Ни для людей жизнеустройства
Не угрожает ни природе.

Защищённость
А защищённость нам покажет,
Как много может от воздействий
Извне и всяческих вмешательств
В системе быть противодействий.

Надёжность
Свойство надёжности системы
Даёт возможность оценить,
Что в установленных пределах
Она параметры хранит

Для функций нужные. И часто
Включает свойств набор таких
Как долговечность, безотказность,
И сохраняемость при них.

Производительность
Производительность системы
Есть внешний признак. Обозначит
Период времени примерный
На выполнение задачи.

Эффективность
А эффективность позволяет
Оценку дать внутри системы –
Насколько мало потребляет
Она ресурсов в это время.

Масштабируемость

А масштабируемость может
Нам показать, насколько тут же
Эффект растёт, коль приумножить
Ресурс, что для системы нужен.

Совместимость, интероперабельность, переносимость
Интероперабельность,
Она же совместимость
Как коммуникабельность
Систем. Переносимость –

Есть совместимость та же,
Только вертикальная:
Одна система ляжет
Другой в основание.

Повторная применимость (Reuseability)
Что ж… применимостью повторной,
Зовут способности системы,
Частей её, быть примененной
В создании новых приложений.

Способность к эволюции
Способность к эволюции –
Возможность изменения,
Даёт добавить функции
Уже после внедрения.

Ремонтопригодность
А свойство «ремонтопригодность»
Обозначает, что в системе
Предполагается возможность
Ремонта и обнаружения

Причин отказов, повреждений,
И прочих сведений полезных
В работах по восстановлению
И в тех. обслуживаниях к месту.

Понятность
Понятность – внутреннее свойство,
Показывающее возможность
Понять системное устройство –
Сокрытую в системе сложность.

Удобство
Удобство – свойство у систем
Показывающее наглядность
И дружественность их ко всем –

Для пользователей приятность.

Опыт пользователя, опыт взаимодействия (User experience, UX)
User experience (Юзер экспириенс) системы –
Дизайн пользования ей,
Весьма полезен в применении
Программы будет у людей.

При проектировании надо
Понять, кто пользователь наш,
Вооружиться этим взглядом,
Экранов набросать коллаж,

Определить их переходы,
Сформировать макеты и
Продемонстрировать в итоге
То, что получится, среди

Возможных юзеров системы
И тех, чей важен интерес,
Чтобы понять, насколько в целом
Такой подходит интерфейс.

Глава 2. Виды информационных систем

Классификация по степени автоматизации
Автоматические системы
Автоматической системе
Участие людей не нужно,
А если нужно, то по мере
Эпизодической с ней дружбы.

Автоматизированные системы
Системы, где в работе нужен
Труд человека постоянно,
И делают его получише
Автоматизи…ированно.

Классификация по сфере применения
Система обработки транзакций или данных (СОД, СОТ, OLTP, TPS)
Системы обработки данных –
Транзакционные системы –
Используются регулярно
Для типовых задач, к примеру,

Сбор данных с датчиков, заказы,
Запросы, платежи… как ритмы,
Решения в них однообразны
И по известным алгоритмам.

Информационно-справочная система (ИСС)
Системы для хранения данных,
Поиска, вывода их в виде
Бумажном или же экранном,
Контекстном и иерархичном.

Информационная система управления (ИСУ, УИС, MIS)
ИСУ, системы управления,
Учётные системы – могут
Хранить в БД свои значения,
Искать их, обработать, чтобы

Формировать из них отчёты.
И оставляют человеку
Решения принимать в расчёте
На понимание данных этих.

Система поддержки принятия решений (СППР)
СППР даёт возможность,
Наборы данных обработав,

Повысить для решений точность
За счет проделанных расчетов.

Информационная система мониторинга (ИСМ, ESS)
Производя объединение
Текущих данных из различных
Других систем для наблюдения
За деятельностью динамичной

Всей фирмы и для выявления
Оперативного проблемы,
Руководителям решения
Принять помогут ИэСэМы.

Система электронного документооборота, автоматизации делопроизводства (СЭД, OAS)
Системы делопроизводства
И документооборота
Предоставляют собой способ
Ускорить в офисе работу.

Согласование документов
Удобней будет многократно,
Когда за каждым из акцептов
Шагать на край Земли не надо.

Классификация по масштабу
Персональная ИС, Автоматизированное рабочее место (АРМ)
Для обособленных задач бывает уместно
Автоматизированное рабочее место –
По сути, программа для одного человека –
Без ролей и синхронизаций бывает, при этом.

Групповая ИС
Для коллективного владения
Набором данных применяют
Системы для подразделения –
Их групповыми называют.

Корпоративная ИС, система планирования ресурсов предприятия (ERP)
Система для учета многих,
А, в идеале, всех ресурсов
На предприятии позволит
Убрать ненужную нагрузку

На повторение процесса
Учета общих данных. Силой
Их актуальность обеспечит
И согласованность в единой

Корпоративной базе данных.
Центральное ядро системы
С набором дополнений разных
Бизнес-процессов и отделов

Собой совместно представляют
Систему ERP (Еэрпэ́). Не спешно
Ей все процессы закрывают,
Когда внедрение успешно.

Система интерактивной аналитической обработки (OLAP)
OLAP (ОЛА́П)-системы позволяют
Наборы данных обработать,
По многим измерениям сразу
Собрав суммарные отчеты.

Система управления эффективностью организации (BPM)
Дополнив ERP ОЛАП-ом,
Получим BPM (Бэпээ́м)-систему.
Взяв в ERP наборы данных,
ОЛАП позволит быстро сделать

Их обработку: выявление
И поиск узких мест в процессах,
И способов их улучшения,
Для повсеместного прогресса.

Глава 3. Жизненный цикл систем

Жизненный цикл представляет
Набор этапов – описание,
В коих система пребывает
За всё своё существование.

С момента зарождения мысли
О появлении системы
До вывода её из жизни –
Всё размещают в эти схемы.

Распространённые этапы
В жизненном цикле для программы:
Анализ, разработка плана,
Концепции и тех. задания,

Реализация проекта,
Отладка и объединение,
Ввод в эксплуатацию, не редко
Сопровождение, завершение.

Модель жизненного цикла
Модели жизненного цикла
Описывают ряд процессов,
Их связи и порядок в «жизни»
Систем для большего прогресса

При их создании и развитии.
Известны разные модели,
Рассмотрим виды основные,
Что чаще можно встретить в деле.

Каскадная модель ЖЦ
Каскадные модели часто
Используют для тех процессов,
Где выполнение понятно
И не предвидится эксцессов.

Реализацию системы
В такой модели представляют
В виде простой линейной схемы –
Всё по порядку выполняют.

Она проста и для показа
Заказчикам понятна, в общем.
Но, если не учесть всё сразу,

Затраты резко станут больше.

Спиральная модель ЖЦ
Начав с простого прототипа,
Спиральная модель позволит
Заказчика довольно быстро
Спросить, насколько всё устроит.

И снизит риск проблем в заказе
Того, что может быть не нужно,
За счёт такой обратной связи.
Но есть и минус – перегружен

Процесс создания системы
Может стать, если будет много
Документации и, в целом,
Возможно растяжение сроков.

Гибкая методология разработки (Agile)
Agile (Аджа́йл) – группа направлений –
Набор подходов и методик
Для разработки приложений,
Который, в общем, нынче в моде.

В Agile разработку кода
Проводят в несколько подходов
По две иль три недели, чтобы
Сформировать на каждом что-то.

И по итогу каждой «сдачи»
Продемонстрировать программу
И скорректировать задачи.
Итак, процесс идёт кругами.

Документации здесь мало.
С заказчиком общения много,
Что в планах риски понижало.
Рабочий код – всему итогом.

Среди методик Scrum (Скра́м) известен
И много прочих, чьи находки,
Основаны на манифесте
Гибкой программной разработки.

Быстрая разработка приложений (Rapid application development, RAD)
RAD(Рад)-разработка получила
Широкое распространение,
Поскольку быстроту сулила
При разработке и внедрении.

И позволяла экономить
Бюджет и время. Предлагая
Минимизировать, где можно,
Усилия. Предоставляя

Полученные результаты
Заказчику для регулярной
Обратной связи на этапе
Любом, чтоб уточнять задания.

1

Адаптивная разработка ПО (Adaptive Software Development, ASD)
При адаптивной разработке
Есть три этапа для повтора:
Обдумывание – в подготовке
И выявлении набора

Потребностей и назначения;
Взаимодействие – чтоб вместе
С заказчиком принять решение;
И обучение – здесь тесты,

Анализ и обзор работы
Дают возможность извлечения
Полезных каждому уроков,
Для непрерывных улучшений.

Экстремальное программирование (Extreme Programming, XP)
При экстремальной разработке
Традиционные подходы
Для сроков более коротких
На уровень выходят новый.

Заказчик рядом для вопросов,
Все пишем максимально просто,
Проверка кода – парный кодинг,

Тесты – написаны до кода,

Релизы – частые как можно,
Рефакторинг – все время тоже,
Владение кодом – будет общим,
Стандарт – единый и не сложный.

Глава 4. Исследование предметной области

Предметная область
Предметная область –
Часть реального мира –
Объекты, чьи свойства
С отношениями в силе

Будем мы изучить
Для любых операций –
Для познания и
Для автоматизаций.

Проект
Проект – особый вид работы,
Из не циклических этапов.
Он ограниченный во многом:
По времени и по затратам.

С ограничением ресурсов,
Не повторяется, обычно,
И потому ему план нужен
Чтоб выполнение шло прилично.

А если проект не пошёл?
Как ни крути, а сделать сразу
Проект получится не всякий.
И остаётся лишь к показу
Представить прототип двоякий.

Потом решить, что делать дальше,
Улучшить или же подправить…
В итоге, позже или раньше,
Его куда подальше сплавить.

Начинать или продолжать?
О, сколько можно начинать?
Пора бы завершить хоть что-то.
Ещё есть способ – передать
Кому-то третьему работу…

Как где-то «слышано» не раз:
Два основных движенья в силе:
Один – держать, что есть сейчас,
Другой – менять устройство в мире.

Когда закончится проект –

Процесс начнётся при удаче,
Но вот поддерживать «эффект»
Уже других людей задача.

Гештальт
Финализировать проект,
Отрезав целостный участок,
Так чтобы не было к нему
Ведущих в памяти путей.
Весьма существенный аспект
Для эффективности, и часто
Для жизни, судя по всему.
Чтоб, в целом, было веселей.

Когда проектов очень много
И тянутся из года в год,
Когда не видно их итога,
И результат не настаёт,
Гештальт открытый дольше срока,
Как потемневший небосвод,
Нависший тучей у порога,
Свою погоду создаёт.

Итак, какой-то из финалов
Даёт надежду нам в пути,
На то, что много или мало,
Но можем далее идти.
Без суматохи и авралов
Итог спокойно подвести
Полезно, чтобы легче стало
Работать далее в АйТи.

Бизнес-процесс
Бизнес-процесс – совокупность работ
И задач всех, что продукт создаёт
Или услугу для потребления
Есть на три группы подразделение:
Управляющие – определяют
Кто как системами управляет.
Операционные – основная работа,
Она приносит в итоге доходы.
Поддерживающие процессы – те,
Что обслуживают другие системы все.

Бизнес-логика
Бизнес-логика – это
Группа принципов, правил,
Поведения объектов –
Всё то, что составит

В разработке предметную
Область решения –
Воплощение конкретное
И ограничения.

Способы изучения предметной области
Есть много способов различных,
Для получения знаний о
Бизнес-процессах и типичных
Их протеканиях. Итого:

Интервьюирование
Интервьюирование – метод
Прямой и эффективный, он
Даёт возможность нам изведать
Всю информацию, при том,

Что будем спрашивать системно,
Записывая без помех
«Предметку», или постепенно
Брать информацию у всех.

Рабочий семинар
Рабочий семинар, пожалуй,
Быть может эффективным самым
Вариантом получения знаний,
Однако сложен и затратен.

Анкетирование
Для массового сбора данных
И группового мнения можно
Использовать анкеты. Брать их
Бумажно или электронно…

Документация
Для изучения процессов,
Весьма полезно получить
Документацию, при этом,
Внимание стоит обратить

На все входные-выходные
Формы, уставы, положения,
Регламенты и должностные
Инструкции – всё в рассмотрение.

Обзор аналогов
Для подготовки к разработке
Необходимо изучение,

Обзор аналогов – насколько
Уже готовые решения

Задачу выполняют – может,
Их применение дешевле,
Чем будет разработка новой
Системы, или совершенней.

Для изучения вариантов
Аналогов системы стоит,
Вначале перечень составить
Из требований, что устроит

Заказчика по всем аспектам.
Найти системы, что подходят,
Вооружившись интернетом.
В табличном виде их оформить,

И указать какие будут
Покрыты требования ими,
Поставив минусы и плюсы
В таблице, где их разместили.

Глава 5. Структурное моделирование

Декомпозиция
Декомпозиция нужна,
Для изучения системы.
Ее использование нам
Даёт систему постепенно

Делить на части до тех пор,
Пока любая из частей
Не станет ясной на обзор,
Позволив разобраться в ней.

Нотации моделирования
Есть много всяческих нотаций
Для построения диаграмм,
Чтоб можно было собираться
Как разработчикам программ,

Так и заказчикам, и прочим
Участникам и «налегке»
Всем разъясняться на рабочем
Одном наглядном языке.

IDEF (Integrated DEFinition)
Методологии семейства
IDEF (Айде́ф) дают создать модели
Систем, предоставляя средства
Различных видов построения.

IDEF0 (Айдее́ф ноль) – этап начальный
Анализа систем – их функций.
На этом виде диаграммы
Есть ряд известных всем конструкций.

Процессы – функции системы,
Потоки данных: управления –
Обычно сверху от процессов,
Выходы справа, входы слева.

Такая форма представления
Бизнес-процессов позволяет
Показывать их отношения –
Соподчиненность отражая.

Диаграммы потоков данных (Data flow diagrams, DFD)
Один из нескольких подходов

Для изучения систем
Их функций и границ народу
Известный многим, хоть не всем –

Подход структурный и системный –
На основании DFD (Дээфдэ́).
С разбором функций постепенным
Для составления ТЗ.

Начальный уровень – контекстный –
На нем есть основной процесс
С потоками взаимодействий
С внешними сущностями. Здесь

Определяются границы
Для построения системы
По документам и страницам
Взаимодействующим с нею.

В дальнейшем изучении будем
Декомпозировать процесс мы
На подпроцессы – список функций
Для изучаемой системы.

Элементы DFD-диаграмм
Для построения моделей
Потоков данных применяют
Нотации. Для этих целей
В них элементы выделяют:

Процесс – указывают смело
Для отражения функций, целей,
Обозначают, что ей делать
Как в целом также и отдельно.

Внешняя сущность – для показа
Объектов вне нашей системы
И демонстрации их связи
С системным основным процессом.

Хранилище – оно же база
Тех данных, что хранят в системе.
Его располагают сразу
На первом уровне модели.

Поток – графическое средство
Показа связей диаграммы:
От внешней сущности к процессу
И от процесса к базе данных.

Словарь данных
Словарик данных помогает
Потокам данных описания
Сформировать. Предоставляет
Их в виде текстового знания.

Так, чтобы было всем понятно,
Что именно передаётся
Между процессов. Аккуратно
В итоге всё в БД сведётся.

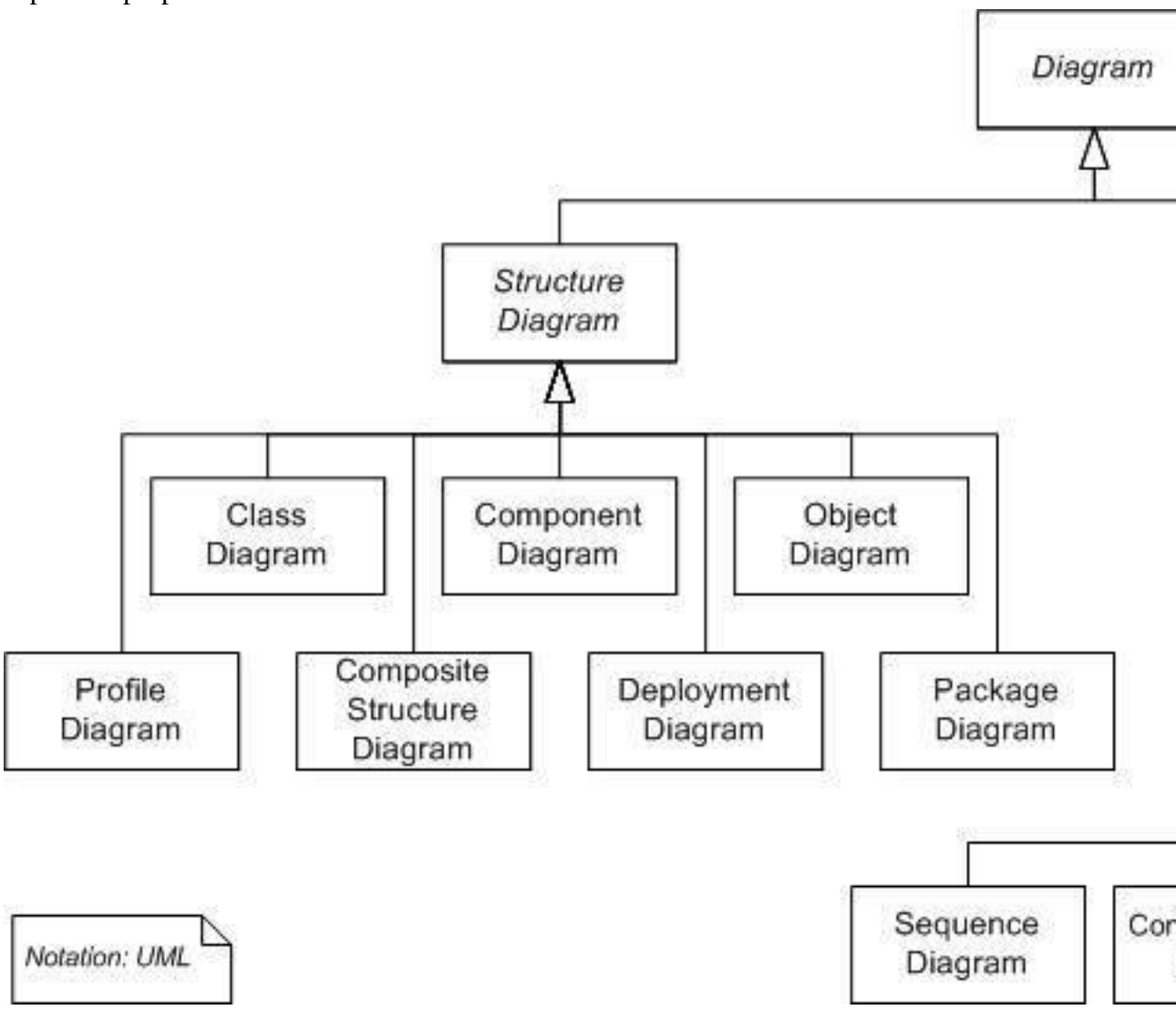
Спецификация процессов
Для описания процессов,
Когда нет смысла в их делении,
Бывает применить полезно
Другие средства в объяснении:

Спецификации, к примеру,
Как описание в виде текста,
Да хоть обычную блок-схему,
Иль флоу-форму – всё уместно.

Глава 6. Объектно-ориентированное моделирование

Унифицированный язык моделирования (Unified Modeling Language, UML)
Для построения диаграмм
В унифицированном виде
При описании программ
Язык объектный примените –

Универсальный – UML (Юмэ́л).
В нём моделируйте процессы
Программных и бизнес-систем
В разных разрезах и контекстах.



Виды диаграмм UML[2]

Диаграмма классов (Class diagram)
Статическая диаграмма
Структуры кода и системы –
Пожалуй, диаграмма классов,

Одна из главных в Юмээле.

На ней показывают классы,
Их методы и атрибуты.
И связи между ними сразу
Здесь тоже есть в их общей сути.

Диаграмма прецедентов (Use case diagram)
На диаграмме прецедентов
Показывают отношения –
Связи от юзеров системы
К ее вариантам выполнения.

Диаграмма последовательности (Sequence diagram)
Взаимодействие объектов
Показывают диаграммой
Последовательности выполнения.
На ней представлены программа

И пользователь, и другие
Участники, как вертикали.
И сообщения между ними
По времени их протекания.

Диаграмма компонентов (Component diagram)
На диаграмме компонентов
Показаны библиотеки,
Модули, файлы и пакеты
И связи между ними всеми.

Диаграмма развёртывания/размещения (Deployment diagram)
На диаграмме размещения
Показывают наложение
Программного обеспечения
На аппаратные решения.

Глава 7. Техническая документация

Техническое задание[3]
Для выполнения проекта
С известным качеством и сроком
Весьма полезным документом
ТЗ является. Во многом

Его задача – однозначность
При понимании системы.
В ТЗ описаны задачи
Проекта так, чтоб были всеми

Они восприняты в едином
Ключе и смысле, и трактовок
Различных не было в картине
И описании разработок.

Частное техническое задание
Когда проект большой ведётся,
И разработчиков в нём много,
На подсистемы создаётся
Задание частное в итоге.

Технический проект[4]
Все описания дальнейших
Проектных принятых решений
Технический проект содержит,
В нём излагают о системе

Устройство, алгоритмы, схемы,
От базы данных до внедрения
И эффективности системы.
На языке для исполнения:

Когда ТЗ для всех понятно,
ТП – уже для программиста –
В нём не столь нужно деликатно
Искоренять все жаргонизмы.

Руководство пользователя
Когда написана система,
Для помощи в работе с нею
Полезна текстовая схема,
Чтоб описать её идею

Для пользователей и просто
Помочь в процессе изучения
Её работы – руководство.
Обычно в нем обозначение

Дается следующим вопросам:
Обзор и ссылки, назначение
Системы, функции и способ
Их применения, и решение

Проблем возможных при работе
И при типичном применении.
Полезный документ для многих,
При изучении приложения.

Руководство администратора
Администратору в работе
Инструкция нужна другая –
В ней описание даёте
Как доступ, роли назначают,

Как заполняют базы данных
И разворачивают сервер,
Как исправлять ошибки надо,
Коль есть известные примерно.

Программа и методика испытаний[5]
Когда проект идёт к внедрению,
Бывает нужен документ,
В котором есть определение,
Как «тестить» каждый элемент.

Программа тестов-испытаний
При разработке под заказ
Даёт возможность понимания,
Что как проверить и, подчас,

Нужна не менее задания
На разработку, ведь по ней
Проводится согласование
С заказчиком системы всей.

В ней нужен список всех условий
Для выполнения работ,
Сценарий, тесты, по которым
Заказчик проверять пойдёт.

Часть 2. Архитектура ПО

Архитектура
Архитектура приложения –
Борьба со сложностями в нём.
И без неё, как наваждение,
Затраты потекут ручьём,

Потом безудержным потоком
На проведение небольших,
Казалось бы, работ. Итогом –
Перерасход сил трудовых.

Программная архитектура –
Есть описание частей
Системы – вся её структура,
Устройство, отношения в ней –

Все те решения, что в дальнейшем
Затратно будет изменять.
Поэтому вопрос важнейший
Их, в целом, правильно принять.

Хорошая архитектура
Даёт возможность широко
Сопровождения процедуру
Вести удобно и легко.

Вся суть здесь в разделении кода
На модули и компоненты.
С таким предположением, чтобы
Ослабить связи элементов.

Охватить проект единой мыслью?
Память нужна программисту для жизни,
Больше, пожалуй, чем многим другим.
Помнить все связи объектов капризных,
Чтоб, ненароком, их не упустить.

Чтобы не вышло неведомых багов,
Нужно, частенько, проект в голове
Как-то держать. Если код одинаков,
Распределен в однотипной канве,

Это становится проще немного.
Если же связи не слишком сильны
Между частями его, то дорога

Эта заметно полегче. Увы

Все ухищрения работают явно,
Пусть хорошо, но эффект всё ж размыт.
Память нужна, чтобы помнить хотя бы
То, для чего код программный открыт.

Глава 8. Парадигмы программирования

Императивное программирование
В императивной парадигме
Код – для процессора команды,
Что будут выполнены в ритме
Последовательном, как ни странно.

Мы пишем то, как надо сделать,
И ожидаем, что так будет.
Здесь можно весь контроль изведать,
Но просто что-то перепутать.

Декларативное программирование
Декларативная программа
Обозначает результаты –
И представляет описание
Того, что нам в итоге надо.

В ней нет значений переменных,
Нет точной логики работы.
Примером может быть отменным
HTML (Аштиэмэ́ль), SQL (Сикьюэ́ль) коды.

Процедурное программирование
При парадигме процедурной
Все операторы и строки –
Команды кода – можно будет
Делить на целостные блоки.

Структурное программирование
В структурной парадигме сутью
Явилось оформление кода
В иерархической структуре –
В формате иерархий блоков.

За счёт того, что между ними
Нет безусловных переходов,
Ограничениями такими
Даёт облегчить тесты кода.

Объектно-ориентированное программирование
При ООП программу строят
Из блоков кода и их данных.
Структуру классами готовят,
И создают их экземпляры,

Как по шаблону, для хранения
И обработки данных, чтобы,
В больших проектах упрощения
Таким путём добыть немного.

Функциональное программирование
В функциональной парадигме
Все вычисления ведутся
В «математическом режиме» –
В формате вычисления функций

Без сохранения состояний.
Используются лишь входные
Их аргументы. Позволяя
Убрать ошибки основные

В многопоточных вычислениях,
Но и цена идёт большая –
Для получения значения
Расчёт сначала повторяя.

Аспектно-ориентированное программирование
В аспектной парадигме можно
Сквозные функции системы,
Которые бывает сложно
Не поместить в другие темы,

К примеру, логи и проверки,
Авторизацию, профайлинг,
Отметить в качестве аспектов.
К примеру, атрибутов в шарпе.

Обобщённое программирование
При обобщённой парадигме
Для обработки разных данных
Используются алгоритмы,
Что пишутся универсально.

Примером могут быть шаблоны –
Дженерик-функции и классы.
Параметрический, в итоге,
Полиморфизм состоялся.

Глава 9. Принципы проектирования

Глобальное проектирование прежде всего (Big Design Up Front)
Прежде всего подумать стоит
И спроектировать систему.
И это, может, сэкономит
Нам кучу времени, проблемы

Предотвратив на ранних сроках.
Так изменить ТЗ – не долго,
А код бессмысленный намного
Дороже написать без толку.

Предметно-ориентированное проектирование (Domain-driven design, DDD)
Для проектирования кода
На основании бизнес-правил –
То бишь проблемного подхода –
Модель предметную составим.

Система, как набор моделей
Предметной области, позволит
Облегчить построение в целом
Её структуры и ускорит

Её развитие в дальнейшем.
Понизит сложность изучения
Частей системы для скорейших
Их написания и внедрения.

Придерживайся простоты (Keep it simple, stupid, KISS)
Усложняя, упрощай –
Избегай ненужных дебрей,
В простоте все сохраняй,
Ищи лучшее решение.

Чем система проще будет,
Тем надёжнее она,
Усложнение всё погубит,
И запутает весьма.

Бритва Оккама (Occam's Razor, OR)
Не нужно сущность создавать,
Когда ей нет особой роли.
Не нужно нового, когда
Подходит то, что есть дотоле.

Не повторяйте себя (Don't Repeat Yourself, DRY)

Не надо повторять себя –
Решай единожды задачи.
Ведь если надо будет взять
И сделать что-либо иначе,

Во всех повторах повторить
Придётся эти изменения
И ничего не пропустить –
Довольно сложно в выполнении.

До написания кодов
Полезно изучить систему:
Вполне возможно, код готов
И кем-то был когда-то сделан.

Вам это не понадобится (You Aren't Gonna Need It, YAGNI)
Пишите только то, что надо,
Прямо сейчас, а не в грядущем.
Это уменьшит вам затраты.
Не нужно делать, что не нужно.

Преждевременная оптимизация (Avoid Premature Optimization)
Не нужно слишком рано код
Оптимизировать упорно.
И лишь когда проект готов,
Оптимизации достойна

Становится программа вся.
И начинать полезно будет
С тех мест, что явно тормозят,
От остального не убудет.

Глава 10. Проектирование слоёв, звеньев и подсистем

Границы и зависимости подсистем
Чтоб нам систему развивать
Удобно было и приятно,
Её полезно разделять
Так, чтобы было не затратно

В дальнейшем части отделить
В процессы, сервисы и службы,
Или обратно совместить
Как монолит – бывает нужно.

Сей выбор можно отложить
И написать систему в целом
Так, чтобы можно было жить
Ей в разных звеньях и разделах.

Как независимую часть
Отдельным модулем, проектом –
В основе – лучше прописать
Всю бизнес-логику. При этом

Взаимодействия вокруг –
Фреймворки, базы данных
И интерфейсы – в виде слуг
И плагинов непостоянных.

Слои абстракции
Для упрощения создания
Больших систем их делят на
Слои, что больше пониманья
Дают для каждого звена.

Слои в себе скрывают сложность,
Давая только интерфейс
Для тех что выше и возможность
Замены внутренностей здесь.

Слои, что ниже, в общем целом,
О верхних знать не должны,
Над ними могут, между делом,
И новые быть введены.

Глобальных данных быть не может,
Всё состояние – внутри.
Конкретных связей, функций – тоже –

Лишь интерфейсы между них.

Благодаря такой структуре
Слои возможно заменять.
Внутри слоёв в архитектуре
Всю сложность лишнюю скрывать.

Звенья
Систему можно разделить
На несколько частей, к примеру,
На сервер и клиент. Решить,
Как будет лучше это сделать,

Обычно, первый из шагов
Архитектуры приложения.
Ряд слабосвязанных узлов
Отдельных называют звенья.

Файл-сервер
Для файл-серверных систем
Предполагается возможность
Хранения данных в файлах. Всем
К ним должен быть открытый доступ.

Двухзвенная архитектура клиент-сервер
Двухзвенная архитектура
Предполагает два звена
И два подхода, как структура,
И логика разделена.

Удаленный доступ к данным (Remote Data Access, RDA)
В модели RDA (Эрдэа́), иначе –
Доступа к удаленным данным –
Вся логика и все задачи
На стороне клиент-программы,

А в базе лишь хранение данных.
При этом больше трафик и
Выше возможность нежеланных
Вмешательств с третьей стороны.

DBS (Database Server)
В модели сервера баз данных,
Иначе – в DBS (Дэбээ́с)-модели –
Клиент – собрание форм экранных,
А логикой владеет сервер.

Но мощь хранимых процедур,

Которыми реализуют
Здесь логику, и их структур,
Пред кода силами пасуют.

Многозвенная архитектура
Сервер приложений (Application Server, AS)
В модели «сервер приложений»
Есть основные три звена:
«Клиент» ведёт отображение,
А серверов, обычно, два.

«Сервер баз данных» для хранения.
И сервер приложения, где
Проходят бизнес-вычисления.
Быть может не один вполне.

Модель подобная сложнее,
Но есть и целесообразность:
Поддержка проще и прямее,
Гораздо выше безопасность.

Многозвенная web-архитектура
Другой вариант трёхзвенки – это
Веб-приложение. Для него
Сервер баз данных будет где-то
И сервер приложения. Но

На сервере и вычисления,
И построение интерфейса.
И лишь одно отображение
В веб-браузере на клиенте.

Микросервисы
Когда систему разделяют
На много маленьких программ,
Их все отдельно запускают
В процессах, службах, тут и там…

По сути, функция – программа –
Отдельный код и разработчик.
Для масштабирования – славно,
Для эффективности – не очень.

Глава 11. Проектирование компонентов

Сложность интеграции изменений единого крупного проекта при командной работе
Когда система велика,
В едином связном коде очень
Бывает сборка не легка,
И каждый новый разработчик

Здесь добавляет свой кусок
При интеграции системы,
На это тратится часок,
А у кого-то и неделя.

Компонент
При разработке код программ
Подразделяют на проекты.
Они потом помогут нам
Разрезать всё на компоненты.

Конец ознакомительного фрагмента.