Министерство образования и науки Российской Федерации
КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. А.Н. ТУПОЛЕВА

Кафедра прикладной математики и информатики им. Ю.В. Кожевникова

А.И. РАХМАТУЛЛИН

ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНЫХ СИСТЕМ

Методические указания по выполнению курсовой работы

Сокращённый вариант для открытой публикации

Казань 2010

Содержание

Введение

Цель курсовой работы закрепление основных понятий и положений дисциплины «Технологии разработки программных систем», приобретение практических навыков анализа и проектирования программных систем.

Задача курсовой работы состоит в анализе и проектировании предметной области (ПрО) реальной организационно-технической системы с использованием объектно-ориентированного анализа и проектирования на основе языка UML в CASE-системе Rational Rose.

Задание по курсовой работе заключается в анализе выбранной (или заданной руководителем) ПрО и построении моделей UML (в том числе диаграмм). Эти модели должны в полной мере раскрывать этап формализации выбранных объектов и процессов для дальнейшей реализации программной системы.

Для построения моделей UML необходимо использовать CASE-систему Rational Rose от IBM Rational. Для генерации кода на Delphi Pascal необходимо использовать средство Rose Delphi Link от фирмы Ensemble Systems, представляющее плагин к системе Rational Rose. Основы работы с системой и плагином рассматриваются в методических указаниях к лабораторным работам.

Содержание курсовой работы ориентировано на проработку студентами умений, полученных во время лабораторных работ, и знаний, приобретённых в течение курса лекций по дисциплине «Технологии разработки программных систем».

Так как RUP является слишком сложным подходом для разработки системы в рамках курсовой работы, в качестве основы выбран подход ICONIX Process.

Курсовая работа выполняется студентами в течение семестра самостоятельно в соответствии с индивидуальными заданиями.

1. Общие сведения

Для выполнения курсовой работы необходимо знать некоторые аспекты языка UML и Процесса ICONIX.

Обзор языка UML

UML – это общецелевой объектно-ориентированный язык для визуализации, специфицирования, анализа, проектирования, конструирования и документирования артефактов программной системы. Использование UML основано на понимании общих принципов моделирования сложных систем, в частности особенностей процесса объектно-ориентированного анализа и проектирования (ООАП).

Принципы моделирования

Рассмотрим кратко принципы моделирования – принципы построения моделей сложных систем.

Первым принципом является *принцип абстрагирования*, который предписывает включать в модель только те аспекты разрабатываемой системы, которые имеют непосредственное отношение к выполнению системой своих функций или своего целевого предназначения. Все второстепенные детали опускаются, чтобы чрезмерно не усложнять процесс анализа и исследования полученной модели.

Вторым принципом является *принцип многомодельности*. Этот принцип представляет собой утверждение о том, что одна модель не может с достаточной степенью адекватности описать различные аспекты сложной системы. Приемлемая модель сложной системы допускает некоторое число взаимосвязанных представлений, каждое из которых адекватно отражает некоторый аспект структуры или поведения системы. Наиболее общими представлениями сложной системы принято считать статическое и динамическое представления, которые в свою очередь могут подразделяться на другие более частные представления.

Третьим принципом является *принцип иерархичности*. Этот принцип предписывает рассматривать процесс построения модели на разных уровнях абстрагирования или детализации в рамках фиксированных представлений.

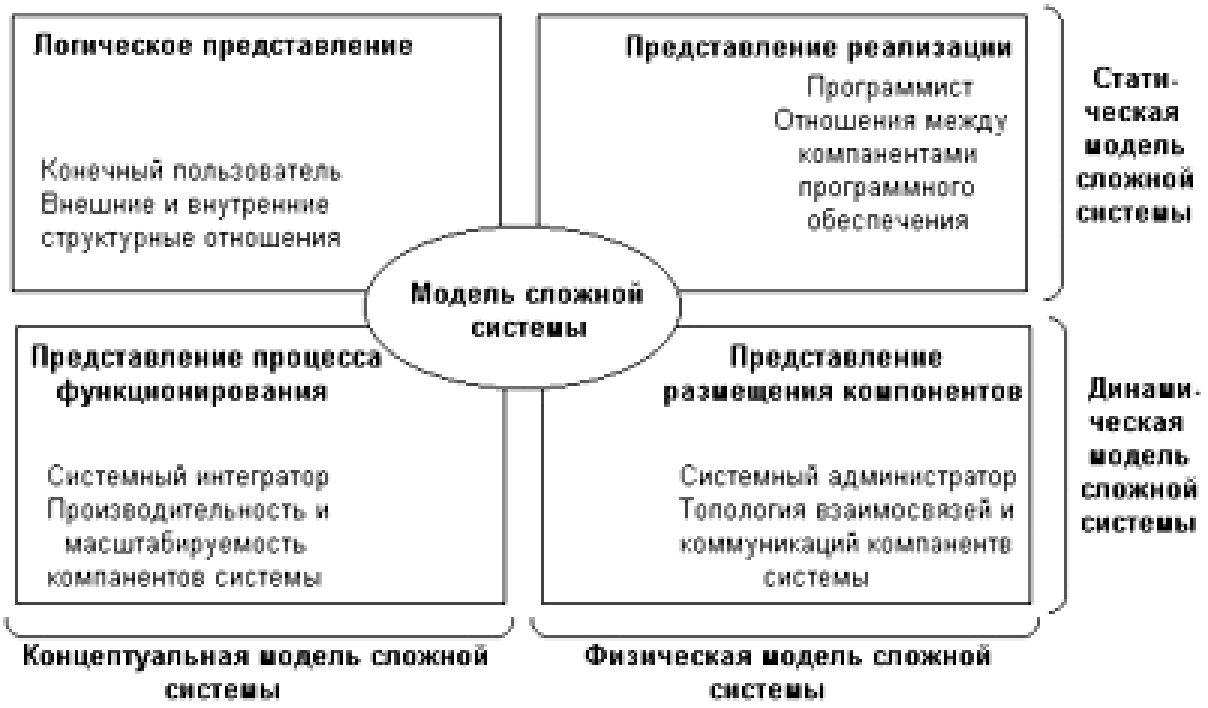


Рис.1. Модели и представления сложной системы

Таким образом, процесс ООАП можно представить как поуровневый спуск от наиболее общих моделей и представлений концептуального уровня к более частным и детальным представлениям логического и физического уровня. При этом на каждом из этапов ООАП эти модели последовательно дополняются всё большим количеством деталей, что позволяет им более адекватно отражать различные аспекты конкретной реализации сложной системы.

Формальное описание

UML состоит из хорошо определённых взаимодействующих частей: семантики и синтаксиса. Семантика представляет собой некоторую метамодель, которая определяет абстрактный синтаксис и семантику понятий объектного моделирования на UML. Синтаксис представляет собой графическую нотацию для визуального представления семантики UML.

Семантика определяется для двух видов моделей: моделей структур и моделей поведения. *Модели структуры*, или *статические модели*, описывают структуру сущностей или частей системы. *Модели поведения*, или *динамические модели*, описывают поведение сущностей системы.

Формальное описание самого UML основывается на некоторой общей иерархической структуре модельных представлений, состоящей из четырех уровней: мета-метамодель, метамодель, модель и объекты пользователя.

Мета-метамодель является основой для всех метамодельных представлений. Главное задача этого уровня состоит в том, чтобы определить язык для спецификации метамодели. Мета-метамодель определяет модель UML на самом высоком уровне абстракции и является наиболее компактным её описанием. С другой стороны, мета-метамодель может специфицировать несколько метамоделей, чем достигается потенциальная гибкость включения дополнительных понятий. Этот уровень тесно связан с теорией формальных языков.

Метамодель является экземпляром или конкретизацией мета-метамодели. Главная задача этого уровня – определить язык для спецификации моделей. Данный уровень является более конструктивным, чем предыдущий, поскольку обладает более развитой семантикой базовых понятий. Все основные понятия UML – это понятия уровня метамодели.

Модель является экземпляром метамодели в том смысле, что любая конкретная модель системы должна использовать только понятия метамодели, конкретизировав их применительно к данному контексту. Это уровень для описания информации о конкретной ПрО. Однако если для построения модели используются понятия UML, то необходима полная согласованность понятий уровня модели с базовыми понятиями UML уровня метамодели.

Объект является экземпляром модели, так как содержит конкретную информацию относительно того, чему в действительности соответствуют те или иные понятия модели.

Метамодель UML описывается на некотором полуформальном языке с использованием трёх представлений: абстрактного синтаксиса, правил правильности и семантики. Поэтому метамодель UML может рассматриваться как комбинация графической нотации, некоторого формального языка и естественного языка.

Представления модели

В рамках UML представления модели сложной системы фиксируются в виде специальных графических конструкций – *диаграмм*.

В UML определены следующие виды диаграмм:

– Диаграммы структуры (Structure diagram):
  - Диаграмма пакетов (Package diagram).
  - Диаграмма классов (Class diagram).
  - Диаграмма композитной структуры (Composite structure diagram):
    ♦Диаграмма кооперации (Collaboration diagram, UML 2.0).
  - Диаграмма объектов (Object diagram).
  - Диаграмма реализации (Implementation diagram).
    ♦Диаграмма компонентов (Component diagram).
    ♦Диаграмма развёртывания (Class diagram).
– Диаграммы поведения (Behavior diagram):
  - Диаграмма прецедентов (Use case diagram).
  - Диаграмма переходов состояний (State machine diagram):
    ♦Диаграмма состояний (Statechart diagram).
    ♦Диаграмма деятельности (Activity diagram).
  - Диаграмма взаимодействия (Interaction diagram).
    ♦Диаграмма последовательности (Sequence diagram).
    ♦Диаграмма синхронизации (Timing diagram, UML 2.0).
    ♦Диаграмма коммуникации (Communication diagram, UML 2.0) / Диаграмма кооперации (Collaboration diagram, UML 1.x).
    ♦Диаграмма обзора взаимодействия (Interaction overview diagram, UML 2.0).

В данных указаниях будет использовать UML версии 1.x.

В ряде подходов используются собственные виды диаграмм, которые тем не менее можно представить как перечисленные выше виды с использованием механизмов расширения.

Диаграмма робастности

Диаграмма робастности (Robustness diagram, тж. диаграмма пригодности) отображает объекты, участвующие в сценарии, и их взаимодействие, в этом смысле она подобна диаграмме кооперации.

Однако диаграмма робастности строится как диаграмма классов с использованием стереотипов. В этой диаграмме используются следующие стереотипы:

- Актёр (actor) – внешний (к системе) субъект/объект, участвующий в сценарии; отображает соответствующего актёра из диаграммы прецедентов.
- Граничный объект (boundary object) – объект на границе системы с внешней средой; обычно является объектом, используемым актёром при взаимодействии с системой; часто отображает связь из диаграммы прецедентов.
- Объект-сущность (entity object) – информационный объект системы; является объектом из модели ПрО; обычно отображает некоторую информацию, необходимую для выполнения прецедента (и не только).
- Объект-управление (control object) – управляющий объект системы; является «соединителем» граничного объекта и объекта-сущности; обычно отображает некоторую функцию, необходимую (только) для выполнения прецедента.

Граничный объект Объект-сущность Объект-управление

Рис.2. Элементы диаграммы робастности

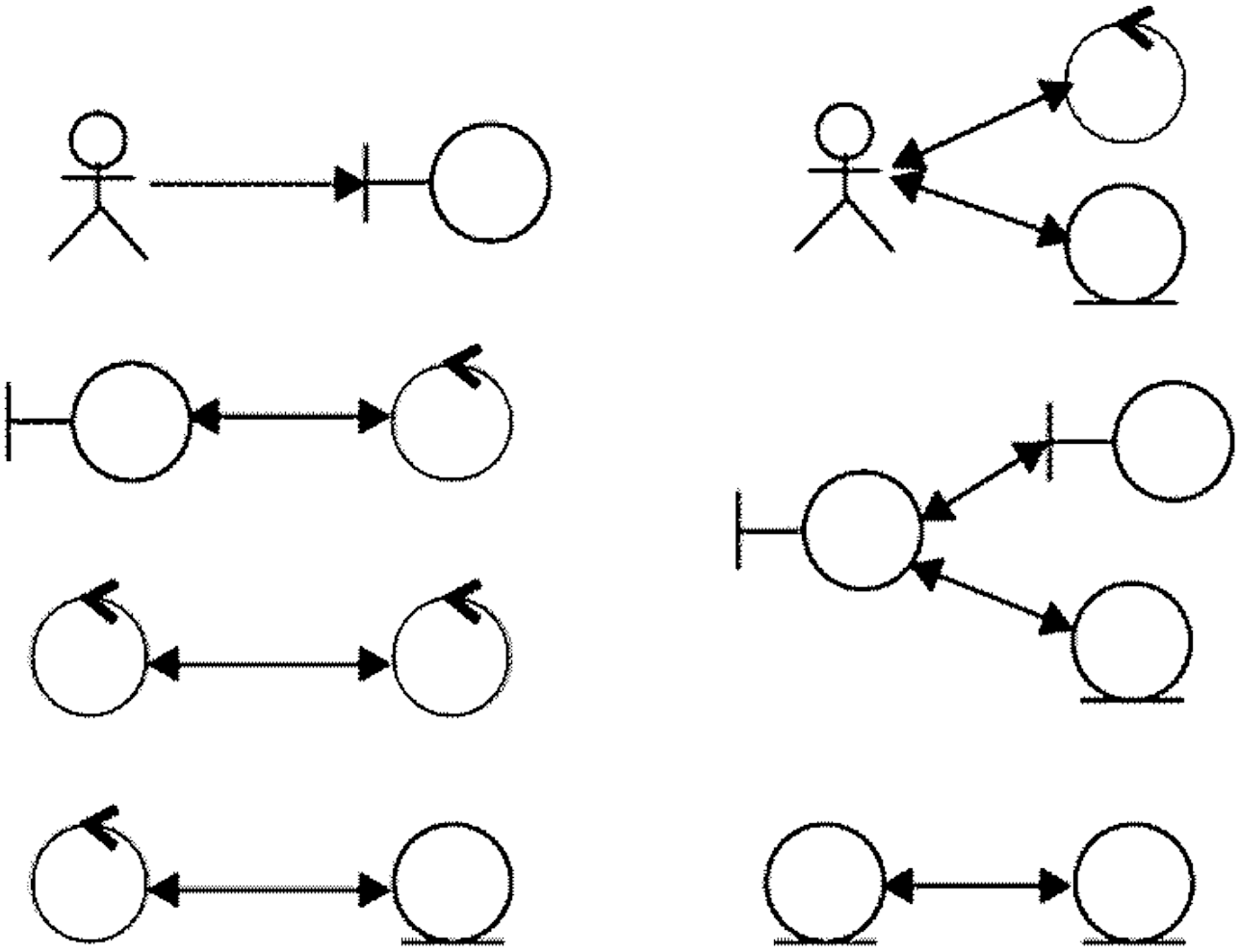
Пример граничных объектов: элементы пользовательского интерфейса (окна, экраны, диалоги, меню). Примеры объектов-сущностей: таблицы базы данных, файлы с информацией длительного хранения, результаты поиска.

Объекты-управления реализуют прикладную логику системы, отделяя её одновременно от граничных объектов и от объектов-сущностей. На практике объекты-управления редко реализуются именно как объекты, они обычно программируются в виде методов классов. Поэтому объект-управление называют также контроллером (controller).

Диаграммы прецедентов обычно отображают основной поток, диаграммы робастности позволяют отобразить одновременно и альтернативные потоки и потому обычно строятся по полным текстовым описаниям прецедентов.

Построение выполняется в соответствии с простыми правилами:

1. Актёры связаны только с граничными объектами.
2. Граничные объекты связаны только с актёрами и контроллерами.
3. Объекты-сущности связаны только с контроллерами.
4. Контроллеры (объекты-управления) связаны с граничными объектами, объектами-сущностями и другими контроллерами, но не с актёрами.



Допустимые связи Недопустимые связи

Рис.3. Правила диаграммы робастности

Использование диаграмм робастности позволяет уточнить диаграмму классов для модели ПрО и более точно построить диаграммы взаимодействия.

2. Порядок выполнения

В ходе выполнения курсовой работы студенты должны разработать архитектуру системы согласно полученным индивидуальным заданиям.

Каждый студент получает индивидуальное задание. Для группы студентов может быть выдано общее задание, если для каждого студента этой группы будет установлена роль и ответственность за выполнение определённой части этого задания, при этом эти части не должны пересекаться и зависеть друг от друга. Эти проблемы решаются студентами группы и во время консультации полученное решение должно быть утверждено руководителем курсовой работы в виде постановки индивидуальной части общего задания.

Определение задания

Постановка индивидуального задания включает в себя следующие части:

1. Тема задания – описание предметной области (ПрО) системы.
2. Функциональные требования к системе – требования заинтересованных лиц, предъявляемые к разрабатываемой системе.
3. Нефункциональные требования к системе – требования, предъявляемые к возможностям системы и/или задающие для неё ограничения.

Постановка задания может быть сформулирована руководителем курсовой работы или самим студентом. Требования к системе должны быть согласованы студентом с руководителем и могут уточняться во время консультаций для прохождения всех этапов выполнения работы.

Этапы выполнения

Для контроля самостоятельной работы студентов в течение семестра и их аттестации, а также проведения консультаций выполнение курсовой работы осуществляется в несколько этапов. Эти этапы во многом соответствуют этапам Процесса ICONIX, а вехи являются моментами проведения консультаций.

Рис.2.4. Модель ЖЦ для Процесса ICONIX

В рамках выполнения курсовой работы необходимо следовать следующему поэтапному порядку построения диаграмм UML для правильной разработки системы по индивидуальному заданию.

Этап 1 «Анализ требований» включает шаги:

- Специфицирование требований. Представление функциональных требований в виде прецедентов. Построение общих диаграмм прецедентов.
- Концептуальное моделирование. Определение (с учётом требований) сущностей ПрО и выявление связей между ними. Построение общей модели ПрО в виде диаграммы классов концептуального уровня.

Веха 1 «Обзор требований» служит для проверки того, что диаграммы этого этапа правильно и достаточно полно совместно отражают все функциональные требования.

Этап 2 «Начальное проектирование» включает шаги:

– Детализация прецедентов. Формирование текстовых описаний прецедентов в виде основного и альтернативных потоков сценариев работы системы.
– Определение с технической архитектурой. Выбор инструментария и платформы с учётом нефункциональных требований.

Веха 2 «Обзор начала дизайна» служит для проверки того, что описания прецедентов соответствуют диаграммам предыдущего этапа, поэтому правильно и достаточно полно совместно отражают все функциональные требования, а выбранная техническая архитектура позволяет разработать систему в соответствии с нефункциональными требованиями.

Этап 3 «Проектирование архитектуры» включает шаги:

– Анализ робастности. Построение диаграммы робастности (как диаграммы классов с использованием стереотипов) для каждого сценария прецедента.
– Логическое моделирование структуры. Обновление диаграммы классов с учётом диаграммы робастности. Получение обновлённой модели ПрО в виде диаграммы классов логического уровня.

Веха 3 «Обзор дизайна архитектуры» служит для проверки того, что диаграммы этапа соответствуют друг другу, правильно и достаточно полно совместно отражают структуру и поведение системы, а также соответствуют диаграммам предыдущего этапа, а значит и функциональным требованиям.

Этап 4 «Проектирование взаимодействия» включает шаги:

– Детализация сценариев. Определение объектов взаимодействия, передаваемых сообщений и используемых методов. Построение диаграмм взаимодействия (диаграмм последовательности и/или кооперации).
– Логическое моделирование поведения. Обновление диаграммы классов с учётом диаграмм взаимодействия. Получение обновлённой модели ПрО в виде диаграммы классов логического уровня.

Веха 4 «Обзор дизайна взаимодействия» аналогична предыдущей вехе.

Этап 5 «Проектирование компонентов» включает шаги:

– Детализация особенностей. Выявление недостаточности построенных моделей при реализации некоторых особенностей системы. Построение диаграмм переходов состояний (диаграмм состояний и/или деятельности).
– Физическое моделирование. Обновление диаграммы классов с учётом диаграмм переходов состояний. Получение детальной модели ПрО в виде диаграммы классов физического уровня.

Веха 5 «Обзор дизайна компонентов» аналогична предыдущей вехе.

Этап 6 «Проектирование реализации» включает шаги:

– Детализация модели. Построение диаграмм реализации (диаграмм компонентов и/или развёртывания).
– Моделирование реализации. Получение окончательной модели ПрО в виде диаграммы классов реализационного уровня.

Веха 6 «Обзор дизайна компонентов» аналогична предыдущей вехе.

Этап 7 «Реализация» включает шаги:

– Генерирование кода. Получение программного кода системы путём автоматической генерации из построенных диаграмм UML с учётом выбранного инструментария и платформы.
– Программирование. Написание оставшегося программного кода системы.
– Верификация. Проверка полученного программного кода системы.

Веха 7 «Обзор дизайна компонентов» служит для проверки того, что программный код соответствует построенным диаграммам, рассматриваемым как руководство к написанию кода.

Содержание отчёта

Результаты выполнения курсовой работы необходимо предоставить в виде отчёта, который должен содержать следующие разделы:

1. Постановка задания.
2. Содержание работы.
3. Введение: описание ПрО.

4. Результаты этапа 1.
   – Спецификация требований: общие диаграммы прецедентов.
   – Общая модель ПрО: диаграмма классов концептуального уровня.
5. Результаты этапа 2.
   – Результаты детализации прецедентов: описания прецедентов.
   – Техническая архитектура: инструментарий и платформа.
6. Результаты этапа 3.
   – Результаты анализа робастности: диаграммы робастности.
   – Логическая модель структуры: диаграмма классов логического уровня.
7. Результаты этапа 4.
   – Результаты детализации сценариев: диаграммы взаимодействия.
   – Логическая модель поведения: диаграмма классов логического уровня.
8. Результаты этапа 5.
   – Результаты детализации особенностей: диаграммы переходов состояний.
   – Физическая модель: диаграмма классов физического уровня.
9. Результаты этапа 6.
   – Результаты детализации модели: диаграммы реализации.
   – Модель реализации: диаграмма классов реализационного уровня.
10. Результаты этапа 7.
   – Результаты генерирования кода: сгенерированный программный код.
   – Результаты программирования: полученный программный код.
   – Результаты верификации (тестирования / инспектирования): исходные данные и результаты проверки, а также программный код системы.
11. Заключение: вывод по результатам выполнения курсовой работы.

3. Типовые задания

Индивидуальное задание формулируются путём выбора либо конкретной ПрО разрабатываемой системы и последующего определения требований к этой системе, либо конкретного варианта задания.

Предметные области

1. Производственное предприятие (завод, комбинат, фабрика).
2. Торговая фирма (магазин, киоск, аптека, сеть магазинов).
3. Учебное заведение (школа, колледж, вуз, специальные курсы).
4. Организация по перевозкам (вокзал, порт, депо, такси, АТП).
5. Общественно-политическая организация (партия, объединение).
6. Фирма «экономического» профиля (банк, ломбард).
7. «Информационное» хранилище (библиотека, музей, НИИ).
8. Средства массовой информации (радио, телевидение, редакция).
9. Отделы предприятия (управление кадров, бухгалтерия, канцелярия).
10. Общественная организация (поликлиника, больница, санаторий).
11. Обслуживающее предприятие (ателье, АТС, АЗС, стоянка).
12. Другие темы по выбору (коллекция объектов, БТИ, бюджет).

Примеры автоматизации

1. Система управления проектами предприятия.
2. Система управления ресурсами предприятия.
3. Система управления документооборотом предприятия.
4. Система управления кадрами предприятия.
5. Система управления взаимодействиями с поставщиками.
6. Система управления взаимодействиями с покупателями.
7. Система предоставления услуг через Интернет.
8. Система автоматизации обслуживания и ремонта.
9. Система сбора и обработки информации предприятия.
10. Система управления продажей и бронированием билетов для мероприятий.

Варианты заданий

1. Система учёта резюме и собеседований для отдела кадров.
2. Система управления распределёнными складами предприятия.
3. Система управления проектами в инвестиционной компании.
4. Система электронного магазина по продаже товаров.
5. Система учёта оплаты услуг для Интернет-провайдера.
6. Система управления процессом транспортировки контейнеров.
7. Система управления процессом обработки корреспонденции.
8. Система автоматизации сервиса для товаров магазина.
9. Система взаимодействия с клиентами для туристической фирмы.
10. Система автоматизации обслуживания гостиничного комплекса.
11. Система управления счетами клиентов для банков.
12. Система управления страховыми взносами для страховых организаций.
13. Система управления страховыми выплатами для страховых организаций.
14. Система управления картами пациентов в медицинских учреждениях.
15. Система учёта правонарушений для отдела правопорядка.
16. Система управления недвижимостью для муниципального органа.
17. Система управления услугами для управляющей компании.
18. Система контроля работы студентов для деканата вуза.
19. Система автоматизации составления расписаний для кафедры вуза.
20. Система организации дистанционного обучения для вуза.
21. Система ведения библиотечных фондов научной организации.

Литература

Основная литература

1. Одинцов И.О. Профессиональное программирование: Системный подход. СПб.: BHV-Санкт-Петербург, 2002. 512 с. (Мастер). (Эл. версия, URL: http://lib.aswl.ru/books/methodology/programming/). См. также: Одинцов И.О. Профессиональное программирование: Системный подход. 2-е изд., доп. СПб.: BHV-Санкт-Петербург, 2004. 624 с. (Мастер).
2. Иванова Г.С. Технология программирования: Учеб. для вузов. М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. 320 с. (Информатика в техн. университете).
3. Орлов С.А. Технологии разработки программного обеспечения: Разработка сложных программных систем: Учеб. пособие. 2-е изд. СПб.: Питер, 2003. 480 с.
4. Воройский Ф.С. Информатика: Новый систематизированный толковый словарь-справочник. (Введение в современные информационные и телекоммуникационные технологии в терминах и фактах). 3-е изд., перераб. и доп. М.: ФИЗМАТЛИТ, 2003. 760 с.
5. Орлик С. Введение в программную инженерию и управление жизненным циклом ПО / При уч. Ю. Булуя. (Эл. изд., URL: http://sorlik.blogspot.com/).
6. Непейвода Н.Н., Скопин И.Н. Основания программирования. М.-Ижевск: Изд-во РХД, 2003. 880 с. (Эл. версия от 11.09.2003. 2+iv+914 с., URL: http://ul-m.udsu.ru/~nnn/fp.zip).
7. Соммервилл И. Инженерия программного обеспечения. 6-е изд. / Пер. с англ. под ред.А.А. Минько. М.: Издат. дом «Вильямс», 2002. 624 с.
8. Калянов Г.Н. CASE-технологии: Консалтинг в автоматизации бизнес-процессов. 3-е изд. М.: Горячая линия – Телеком, 2002. 320 с.
9. Кватрани Т. Rational Rose 2000 и UML: Визуальное моделирование / Пер. с англ. М.: ДМК Пресс, 2001. 176 с. (Объектно-ориентированные технологии в программировании).

Дополнительная литература

10. Системный анализ и принятие решений: Словарь-справочник / Под общ. ред. В.Н. Волковой, В.Н. Козлова. М.: Высш. шк., 2004. 616 с.
11. Непейвода Н.Н. Стили и методы программирования // Интернет-университет информационных технологий, 2005. (Проект Изд-ва «Открытые системы»). (Эл. изд., URL: http://www.intuit.ru/department/se/progstyles/).
12. Технология разработки программного обеспечения: Конспект лекций / Автор неизвестен. (Эл. изд., URL: http://www.solomil.ru/).
13. Берлинский К. Набор серебряных пуль: Справочник удачных проектных решений при разработке ПО. 2004. 99 с. (Эл. изд., v.1.37, 20.06.2004).
14. Марков Е. Архитектура, управляемая моделью // CIT City. 2005. (Эл. изд.: 15.12.2005, URL: http://citcity.ru/).
15. Безуглый Д.Л. Технология разработки программного обеспечения // Корпоративные Информационные Системы. 2001. №2. С. 24–30.
16. IBM Corp. Методология и инструментальные средства IBM Rational для разработки программных систем. 26.04.2007 / Пер. с англ. (Эл. изд., URL: http://1050049.ru/iservices/files.asp?artId=2186&file=0).
17. Microsoft Corp. Microsoft Solutions Framework: Белая книга. Модель процессов MSF. Ver. 3.1, 06.2002 / Пер. с англ. под ред. В. Павлова. (Эл. изд., URL: http://www.microsoft.com/rus/docs/msdn/msf/MSF_process_model_rus.doc).
18. Коуберн А. Четвёртое измерение, или Как обмануть Железный Треугольник / Пер. с англ. К. Максимова, А. Максимова. 08.08.2004. (Эл. изд., оригинал от 04.10.2003, URL: http://www.maxkir.com/).
19. Коуберн А. Создание программного обеспечения как коллективная игра (По материалам статей А. Коуберна за 1997 – 2004 гг.) / Пер. с англ. К. Максимова, А. Максимова. 11.09.2004. (Эл. изд., URL: http://www.maxkir.com/).
20. Фаулер М. Новые методологии программирования / Пер. с англ. К. Максимова, А. Максимова. 13.10.2001. (Эл. изд., URL: http://www.maxkir.com/).

21. Хайсмит Дж. Устаревшие методологии – на пенсию! / Пер. с англ. К. Максимова, А. Максимова. 20.03.2002. (Эл. изд., оригинал от 07 – 08.2000, URL: http://www.maxkir.com/).

22. Бек К. Экстремальное программирование. СПб.: Питер. 216 с. (Б-ка программиста). См. также: Бек К. Экстремальное программирование: Разработка через тестирование. СПб.: Питер, 2003. 224 с. (Б-ка программиста).

23. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приёмы объектно-ориентированного программирования: Паттерны проектирования / Пер. с англ. А. Слинкина. СПб.: Питер, 2001. 368 с. (Б-ка программиста).

24. Дубина О. Обзор паттернов проектирования // CIT Forum. 2005. (Эл. изд., URL: http://citforum.ru/SE/project/pattern/).

25. Ершов А.П. Научные основы доказательного программирования. 1984. 15 с. (Эл. версия, URL: http://www.europrog.ru/paper/ae1984-02r.pdf).

26. Ершов А.П. Отношения методологии и технологии программирования // Технология программирования. Информ. материалы и тез. плен. докл. II Всесоюз. конф. (Киев, 18 – 21.11.1986.) / Ин-т кибернетики им. В.М. Глушкова, АН УССР. Киев, 1986. С. 10 – 13. (Эл. версия, URL: http://ershov.iis.nsk.su/).

27. Дейкстра Э.В. Дисциплина программирования / Пер. с англ. под ред. Э.З. Любимского. М.: Мир, 1978. 290 с. (Математическое обеспечение ЭВМ).

28. Грис Д. Наука программирования / Пер. с англ. под ред. А.П. Ершова. М.: Мир, 1984. 416 с.

29. Вендров А.М. CASE-технологии: Современные методы и средства проектирования информационных систем // CIT Forum. 1997. (Эл. изд., URL: http://citforum.ru/).

30. Норенков И.П. Основы автоматизированного проектирования: Учеб. для вузов. 2-е изд., перераб. и доп. М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. 336 с. (Информатика в техн. университете).

31. Дубова Н. В круге разработки // Открытые системы. 2003. № 9; CIT City. (Эл. изд.: 18.09.2003, URL: http://www.osp.ru/os/).

Документация

Основные стандарты, связанные с жизненным циклом ПО и систем:

32. ISO/IEC 12207:1995 «Information Technology – Software Life Cycle Processes». Рус. версия: ГОСТ Р ИСО/МЭК 12207:1999 «Информационная технология. Процессы жизненного цикла программных средств».
33. ISO/IEC 15288:2002 «Systems Engineering – System Life Cycle Processes». Рус. версия: ГОСТ Р ИСО/МЭК 15288:2005 «Системная инженерия. Процессы жизненного цикла систем».
34. ISO/IEC TR 15271:1998 «Information Technology – Guide for ISO/IEC 12207 (Software Life Cycle Processes)». Рус. версия: ГОСТ Р ИСО/МЭК ТО 15271:2002 «Информационная технология. Руководство по применению ГОСТ Р ИСО/МЭК 12207 (Процессы жизненного цикла программных средств)».
35. ISO/IEC TR 16326:1999 «Software Engineering – Guide for the Application of ISO/IEC 12207 to Project Management». Рус. версия: ГОСТ Р ИСО/МЭК ТО 16326:2002 «Программная инженерия. Руководство по применению ГОСТ Р ИСО/МЭК 12207 при управлении проектом».
36. ISO/IEC TR 19760:2003 «Systems Engineering – A Guide for the Application of ISO/IEC 15288 (System Life Cycle Processes)». Рус. версия: ГОСТ Р ИСО/МЭК ТО 19760:? «Системная инженерия. Руководство по применению ГОСТ Р ИСО/МЭК 15288 (Процессы жизненного цикла систем)».

Другая документация по разработке ПО и систем:

37. IEEE Guide to the Software Engineering Body of Knowledge (SWEBOK). 2004 Version. IEEE Computer Society. Los Alamos, CA. 2004. 204 p.

Интернет – источники

38. Объектно-ориентированный анализ и проектирование. (обозначение сайта – ООА&П / OOA&D, URL: http://ooad.asf.ru/).
39. Клуб разработчиков программных систем. (URL: http://www.caseclub.ru/).

40. Сайт MAXKIR.com – переводы зарубежных публикаций К. Максимова, А. Максимова. (URL: http://www.maxkir.com/)

41. Wikipedia.org – The Free Encyclopedia. (URL: http://en.wikipedia.org/).

42. Европейский центр программирования. (URL: http://www.europrog.ru/).

43. Agile Manifesto – Живой манифест. (URL: http://www.agilemanifesto.org/).

44. PraxOS – Организационная система PraxOS. (URL: http://praxos.ru/).

45. ИНТУИТ – Интернет-университет информационных технологий. (URL: http://www.intuit.ru/).

46. Open Systems – Проект «Открытые системы». (URL: http://www.osp.ru/).

47. CIT – Центр информационных технологий. (URL: http://www.citforum.ru/, http://www.citcity.ru/ и другие сайты).