Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»

Н. Б. ОСИПЕНКО, А. Н. ОСИПЕНКО

CASE-ИНСТРУМЕНТАРИЙ АВТОМАТИЗАЦИИ АНАЛИЗА, ПРОЕКТИРОВАНИЯ И РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И СТАНДАРТЫ ДОКУМЕНТИРОВАНИЯ ПРОГРАММНЫХ СРЕДСТВ

Практическое руководство

для студентов специальности I–40 01 01
«Программное обеспечение информационных технологий»

Гомель
ГГУ им. Ф.Скорины
2015

УДК 004.41-057.4
ББК 32.972я73
О-74

Рецензенты:
кандидат физико-математических наук А. И. Рябченко;
кандидат физико-математических наук Д. С. Кузьменков

Рекомендовано к изданию научно-методическим советом
учреждения образования «Гомельский государственный
университет имени Франциска Скорины»

Осипенко, Н. Б.

О-74 CASE-инструментарий автоматизации анализа, проектирования и разработки программного обеспечения и стандарты документирования программных средств : практическое руководство / Н. Б. Осипенко, А. Н. Осипенко ; М-во образования РБ, Гом. гос. ун-т им. Ф. Скорины. – Гомель: ГГУ им. Ф. Скорины, 2015. – 38 с.

ISBN 978-985-577-055-9

Практическое руководство предназначено для оказания помощи студентам в усвоении знаний по двум разделам курса «Основы стандартизации и сертификации программного обеспечения»: CASE-инструментарий автоматизации анализа, проектирования и разработки программного обеспечения и стандарты документирования программных средств». В него включены также контрольные вопросы по указанным разделам.

Адресовано студентам специальности 1–40 01 01 «Программное обеспечение информационных технологий».

УДК 004.41-057.4
ББК 32.972я73

ISBN 978-985-577-055-9

Оглавление

Предисловие

Целью данного практического руководства является оказание помощи студентам в усвоении знаний и формировании практических навыков при изучении CASE-инструментария автоматизации анализа, проектирования и разработки программного обеспечения, а также стандартов документирования программных средств, приобретаемых в рамках курса «Основы стандартизации и сертификации программного обеспечения».

Одним из важных способов улучшения качества программного обеспечения в соответствии с требованиями пользователей программной продукции является ее стандартизация и аттестация работы программного обеспечения; контроль за внедрением и соблюдением стандартов за счет использования CASE-инструментария автоматизации анализа, проектирования и разработки программного обеспечения. Создание конкурентоспособной программной продукции невозможно без использования стандартов документирования программных средств.

В практическое руководство включены теоретические сведения по двум разделам: «CASE-инструментарий автоматизации анализа, проектирования и разработки программного обеспечения» и «Стандарты документирования программных средств» и четырем темам: «Классификация CASE-инструментария», «Концептуальные основы CASE-технологий», «Общая характеристика проблем и задач документирования программного обеспечения» и «Единая система программной документации», а также вопросы для самоконтроля, усвоение которых необходимо для выполнения заданий на практических занятиях.

Практическое руководство адресовано студентам специальности 1–40 01 01 «Программное обеспечение информационных технологий».

1 CASE-инструментарий автоматизации анализа, проектирования и разработки программного обеспечения

1.1 Классификация CASE-инструментария

1.1.1 Классификация по типам

Традиционно CASE[1)]-инструментарий автоматизации анализа, проектирования и разработки программного обеспечения систематизируют на типы, категории и уровни [2, с. 173]. Классификация ***по типам*** отражает функциональную ориентацию CASE-инструментария в технологическом процессе.

1 ***Анализ и проектирование.*** Средства данной группы используются для создания спецификаций системы и ее проектирования; они поддерживают широко известные методологии проектирования [2, стр.173]. К таким средствам относятся: *The Developer (ASYST Technologies), POSE (Computer Systems Advisers), ProKit*Workbench (McDonnell Douglas), Excelerator (Index Technology), Design–Aid (Nastec), Design Machine (Optima), MicroStep (Mela Systems), vsDesigner (Visual Software), Analist/Designer (Yourdon), Design/IDEF (Meta Software), BPWin (Logic Works), SELECT (Select Software Tools), System Architect (Popkin Software & Systems), Westmount I–CASE Yourdon (Westmount Technology B. V. & CADRE Technologies), CASE/4/0 (microTOOL GmbH); CASE.Аналитик (Эйтэкс)*. Их целью является определение системных требований и свойств, которыми система должна обладать, а также создание проекта системы, удовлетворяющей этим требованиям и обладающей соответствующими свойствами. На выходе продуцируются спецификации компонент системы и интерфейсов, связывающих эти компоненты, а также «калька» архитектуры системы и детальная «калька» проекта, включающая алгоритмы и определения структур данных.

2 ***Проектирование баз данных и файлов.*** Средства данной группы обеспечивают логическое моделирование данных, автоматическое

[1)] Аббревиатура CASE расшифровывается как Computer Aided Software Engineering (программная инженерия с компьютерной поддержкой).

преобразование моделей данных в третью нормальную форму, автоматическую генерацию схем БД и описаний форматов файлов на уровне программного кода: *ERWin (Logic Works), Chen Toolkit (Chen &Asssociates), S-Designor (SDP), Designer2000 (Oracle), Silverrun (Computer Systems Advisers).*

3 ***Программирование.*** Средства этой группы поддерживают этапы программирования и тестирования, а также автоматическую кодогенерацию из спецификаций, получая полностью документированную выполняемую программу: *COBOL 2/Workbench (Mikro Focus), DECASE (DEC), NETRON/CAP (Netron), APS (Sage Software).* Помимо диграммеров различного назначения и средств поддержки работы с репозитарием[1)], в эту группу средств включены и традиционные генераторы кодов, анализаторы кодов (как в статике, так и в динамике), генераторы наборов тестов, анализаторы покрытия тестами, отладчики.

4 ***Сопровождение и реинжиниринг.*** К таким средствам относятся документаторы, анализаторы программ, средства реструктурирования и реинжиниринга: *Adpac CASE Tools (Adpac), Scan/COBOL и Superstructure (Computer Data Systems), Inspector/Recoder (Language Technology).* Их целью является корректировка, изменение, анализ, преобразование и реинжиниринг существующей системы. Средства позволяют осуществлять поддержку всей системной документации, включая коды, спецификации, наборы тестов; контролировать покрытие тестами для оценки полноты тестируемости; управлять функционированием системы и т. п. Особый интерес представляют средства обеспечения мобильности (в CASE они получили название средств миграции) и реинжиниринга. К средствам миграции относятся трансляторы, конверторы, макрогенераторы и другие, позволяющие обеспечить перенос существующей системы в новое операционное или аппаратурное окружение. Средства реинжиниринга включают:

– статические анализаторы для продуцирования схем системы ПО из ее кодов, оценки влияния модификаций (например, «эффекта ряби» – внесение изменений с целью исправления ошибок порождает новые ошибки);

– динамические анализаторы (обычно, компиляторы и интерпретаторы с встроенными отладочными возможностями);

– документаторы, позволяющие автоматически получать обновленную документацию при изменении кода;

– редакторы кодов, автоматически изменяющие при редактировании и все предшествующие коду структуры (например, спецификации);

[1)] То же, что репозиторий; хранилище, место хранения.

– средства доступа к спецификациям, их модификации и генерации нового (модифицированного) кода;

– средства реверсного инжиниринга, транслирующие коды в спецификации.

5 ***Окружение.*** Средства поддержки платформ для интеграции, создания и придания товарного вида CASE-средствам: *Multi/Cam (AGS Management Systems), Design/OA (Meta Software).*

6 ***Управление проектом.*** Средства, поддерживающие планирование, контроль, руководство, взаимодействие, т. е. функции, необходимые в процессе разработки и сопровождения проектов: *Project Workbench (Applied Business Technology).*

1.1.2 Классификация по категориям

Классификация по категориям определяет уровень интегрированности по выполняемым функциям и включает вспомогательные программы (tools), пакеты разработчика (toolkit) и инструментальные средства (workbench). Категория ***tools*** обозначает вспомогательный пакет, решающий небольшую автономную задачу, принадлежащую проблеме более широкого масштаба. Категория ***toolkit*** представляет совокупность интегрированных программных средств, обеспечивающих помощь для одного из классов программных задач; использует репозитарий для всей технической и управляющей информации о проекте, концентрируясь при этом на поддержке, как правило, одной фазы или одного этапа разработки ПО. Категория ***workbench*** представляет собой интеграцию программных средств, которые поддерживают системный анализ, проектирование и разработку ПО; используют репозитарий, содержащий всю техническую и управляющую информацию о проекте; обеспечивают автоматическую передачу системной информации между разработчиками и этапами разработки; организуют поддержку практически полного ЖЦ (от анализа требований и проектирования ПО до получения документированной выполняемой программы). Workbench, по сравнению с toolkit, обладает более высокой степенью интеграции выполняемых функций, большей самостоятельностью и автономностью использования, а также наличием тесной связи с системными и техническими средствами аппаратно-вычислительной среды, на которой workbench функционирует. По существу, workbench может рассматриваться как автоматизированная рабочая станция, используемая как инструментарий для автоматизации всех или отдельных совокупностей работ по созданию ПО.

1.1.3 Классификация по уровням

Классификация по уровням связана с областью действия CASE в пределах жизненного цикла ПО. Однако четкие критерии определения границ между уровнями не установлены, поэтому данная классификация имеет качественный характер.

Верхние (Upper) CASE часто называют средствами компьютерного планирования. Они призваны повышать эффективность деятельности руководителей фирмы и проекта путем сокращения затрат на определение политики фирмы и на создание общего плана проекта. Этот план включает цели и стратегии их достижения, основные действия в свете целей и задач фирмы, установление стандартов на различные виды взаимосвязей и т. д. Использование верхних CASE позволяет построить модель предметной области, отражающую всю существующую специфику. Она направлена на понимание общего и частного механизмов функционирования, имеющихся возможностей, ресурсов, целей проекта в соответствии с назначением фирмы. Эти средства позволяют проводить анализ различных сценариев (в том числе наилучших и наихудших), накапливая информацию для принятия оптимальных решений.

Средние (Middle) CASE считаются средствами поддержки этапов анализа требований и проектирования спецификаций и структуры ПО. Их использование существенно сокращает цикл разработки проекта; при этом важную роль играет возможность накопления и хранения знаний, обычно имеющихся только в голове разработчика–аналитика, что позволит использовать накопленные решения при создании других проектов. Основная выгода от использования среднего CASE состоит в значительном облегчении проектирования систем, проектирование превращается в итеративный процесс, включающий следующие действия: пользователь обсуждает с аналитиком требования к проектируемой системе; аналитик документирует эти требования, используя диаграммы и словари входных данных; пользователь проверяет эти диаграммы и словари, при необходимости модифицируя их; аналитик отвечает на эти модификации, изменяя соответствующие спецификации. Кроме того, средние CASE обеспечивают возможности быстрого документирования требований и быстрого прототипирования.

Нижние (Lower) CASE являются средствами разработки ПО (при этом может использоваться до 30 % спецификаций, созданных средствами среднего CASE). Они содержат системные словари и графические средства, исключающие необходимость разработки физических спецификаций. Имеются системные спецификации, которые

непосредственно переводятся в программные коды разрабатываемой системы (при этом автоматически генерируется до 80–90 % кодов). На эти средства возложены также функции тестирования, управления конфигурацией, формирования документации. Главными преимуществами нижних CASE являются: значительное уменьшение времени на разработку, облегчение модификаций, поддержка возможностей прототипирования (совместно со средними CASE).

Цели и достоинства верхних, средних и нижних CASE приведены в таблице 1.1.

Таблица 1.1 – Цели и достоинства верхних, средних и нижних CASE

Свойства	Верхние CASE	Средние CASE	Нижние CASE
Чем являются?	Средствами компьютерного планирования	Средствами поддержки этапов анализа требований и проектирования спецификаций и структуры ПО	Средствами разработки ПО
Какая цель?	Повысить эффективность деятельности руководителей фирмы и проекта	Существенно сократить цикл разработки проекта, дает возможность накопления и хранения данных о создании других проектов	Непосредственно перевести системные спецификации в программные коды разрабатываемой системы
Какие достоинства?	Позволяют построить модель предметной области, отражающую всю существующую специфику	Проектирование превращается в итеративный процесс взаимодействия аналитика и пользователя; обеспечивают возможности быстрого документирования требований и прототипирования	Значительное уменьшение времени на разработку, облегчение модификаций, поддержка возможностей прототипирования

1.1.4 Эволюция CASE-инструментария

С самого начала CASE-технологии развивались с целью преодоления ограничений ручных применений методологий структурного

анализа и проектирования 60–70-х гг. (сложности понимания, большой трудоемкости и стоимости использования, трудности внесения изменений в проектные спецификации и т. д.) за счет их автоматизации и интеграции поддерживающих средств. Таким образом, CASE-технологии не могут считаться самостоятельными методологиями, они только делают более эффективными пути их применения. CASE – не революция в программотехнике: современные CASE-средства являются естественным продолжением эволюции всей отрасли средств разработки ПО. Традиционно выделяют шесть периодов, качественно отличающихся применяемой техникой и методами разработки ПО, которые характеризуются использованием в качестве инструментальных следующих средств: ассемблеров, дампов памяти, анализаторов; компиляторов, интерпретаторов, трассировщиков; символических отладчиков, пакетов программ; систем анализа и управления исходными текстами; CASE-средств анализа требований, проектирования спецификаций и структуры, редактирования интерфейсов (первая генерация CASE-1); CASE-средств генерации исходных текстов и реализации интегрированного окружения поддержки полного жизненного цикла разработки ПО (2-ая генерация CASE-2). Краткая характеристика возможностей, которые появились на шести периодах эволюции CASE-инструментария приведена в таблице 1.2.

CASE-I является первой технологией, адресованной непосредственно системным аналитикам и проектировщикам, и включающей средства для поддержки графических моделей, проектирования спецификаций, экранных редакторов и словарей данных. Она не предназначена для поддержки полного ЖЦ и концентрирует внимание на функциональных спецификациях и начальных шагах проекта – системном анализе, определении требований, системном проектировании, логическом проектировании БД.

CASE-2 отличается значительно более развитыми возможностями, улучшенными характеристиками и исчерпывающим подходом к полному ЖЦ. В ней в первую очередь используются средства поддержки автоматической кодогенерации, а также обеспечивается полная функциональная поддержка для порождения графических системных требований и спецификаций проектирования; контроля, анализа и связывания системной информации, а также информации по управлению проектированием; построения прототипов и моделей системы; тестирования, верификации и анализа сгенерированных программ; генерации документов по проекту; контроля на соответствие стандартам по всем этапам ЖЦ. CASE-2 может включать свыше 100 функциональных компонент, поддерживающих все этапы ЖЦ,

при этом пользователям предоставляется возможность выбора необходимых средств и их интеграции в нужном составе.

Таблица 1.2 – Характеристика возможностей, которые появились на шести периодах эволюции CASE-инструментария

Этап эволюции	Что появилось	Что позволяют
1	2	3
1	Ассемблеры	Позволяют писать самый быстрый и компактный код, какой вообще возможен для данного процессора
	Дампы памяти	Собрать информацию о причинах фатальных для программного обеспечения сбоев в его собственной работе
	Анализаторы	Позволяют с большой точностью оценить многие параметры работы сети, такие как скорость прохождения сигналов, участки скопления коллизий и т. д. Выявляют атаки на компьютерные сети и оповещают администратора о них на основе анализа трафика
2	Компиляторы	Проводят трансляцию машинной программы с проблемно-ориентированного языка на машинно-ориентированный язык
	Интерпретаторы	Позволяют работать на любой платформе, на которой есть соответствующий интерпретатор; упростить отладку исходного кода программ. Более совершенные и наглядные средства диагностики ошибок в исходных кодах. Имеют маленький размер кода
	Трассировщики	Позволяют пошагово выполнять программы
3	Символические отладчики	Позволяют программисту вести отладку программы непосредственно в исходном тексте, просматривать и изменять при остановах значения переменных, обращаясь к ним по именам
	Пакеты программ	Позволяют решать задачи определённого класса некоторой предметной области
4	Системы анализа и управления исходными текстами	Позволяют установить структурные связи между переменными или элементами исследуемой системы.

Окончание таблицы 1.2

1	2	3
5	CASE-средства анализа требований	Поддерживают работу пользователей при создании и редактировании графического проекта в интерактивном режиме
	CASE-средства проектирования спецификаций и структуры	Позволяют существенно сократить цикл разработки проекта.
	CASE-средства редактирования интерфейсов	Включают средства поддержки графических моделей, проектирования спецификаций, экранных редакторов и словарей данных
6	CASE-средства генерации исходных текстов и реализации интегрированного окружения поддержки полного ЖЦ разработки ПО	Поддерживают автоматическую кодогенерацию. Обеспечивают полную функциональную поддержку: порождения графических системных требований и спецификаций проектирования, контроля, анализа и связывания системной информации, а также информации по управлению проектированием; построения прототипов и моделей системы; тестирования, верификации и анализа сгенерированных программ; генерации документов по проекту

Вопросы и задания для самоконтроля

1 Приведите классификацию CASE-инструментария по типам.

2 Каковы цели CASE-инструментария из группы анализа и проектирования системы?

3 Каковы цели CASE-инструментария из группы проектирования баз данных и файлов?

4 Каковы цели CASE-инструментария из группы программирования?

5 Каковы цели CASE-инструментария из группы сопровождения и реинжиниринга?

6 Каковы цели CASE-инструментария из группы окружение?

7 Каковы цели CASE-инструментария из группы управление проектом?

8 Как классифицируется CASE-инструментарий по категориям?

9 Как классифицируется CASE-инструментарий по уровням?

10 Какую часть ЖЦ разработки ПО поддерживают верхние CASE? Какова их цель?

11 Какую часть ЖЦ разработки ПО поддерживают средние CASE? Какова их цель и выгода от их применения?

12 Какую часть ЖЦ разработки ПО поддерживают нижние CASE? Какова их цель и выгода от их применения?

13 Какая цель, какие достоинства, чем являются верхние (Upper), средние (Middle) и нижние (Lower) CASE? Изобразите схематично в виде таблицы.

14 Изобразите схематично, какие стадии ЖЦ разработки ПО поддерживают верхние, средние и нижние CASE.

15 Назовите этапы эволюции инструментальных средств, которые привели к появлению CASE-инструментария.

16 Каковы цели CASE-инструментария 1 и 2. Примеры CASE-1 и CASE-2?

1.2 Концептуальные основы CASE-технологий

1.2.1 CASE-модель жизненного цикла программного обеспечения

CASE-технологии предлагают новый, основанный на автоматизации, подход к концепции ЖЦ ПО. При использовании CASE изменяются все фазы ЖЦ, при этом наибольшие изменения касаются фаз анализа и проектирования. Простейшая модель ЖЦ реализуется этапами: анализа, проектирования, кодирования, тестирования и сопровождения; в соответствующей ей CASE-модели ЖЦ (прототипирование, проектирование спецификаций, контроль проекта, кодогенерация, системное тестирование, сопровождение) фаза прототипирования заменяет традиционную фазу системного анализа. Необходимо отметить, что наиболее автоматизируемыми фазами являются фазы контроля проекта и кодогенерации (хотя все остальные фазы также поддерживаются CASE-средствами).

В таблице 1.3 приведены оценки трудозатрат по фазам ЖЦ. Первая строка таблицы соответствует традиционной разработке, вторая – разработке с использованием структурных методологий проектирования, третья – разработке с использованием CASE-технологий. В таблицу 1.4 сведены основные изменения в ЖЦ при использовании CASE-технологий по сравнению с традиционной разработкой.

Таблица 1.3 – Оценки трудозатрат по фазам ЖЦ

Способ разработки	Анализ	Проектирование	Кодирование	Тестирование
Традиционная разработка	20 %	15 %	20 %	45 %
Использование структурных методологий проектирования	30 %	30 %	15 %	25 %
Использование CASE-технологий	40 %	40 %	5 %	15 %

Таблица 1.4 – Основные отличия ЖЦ при использовании CASE-технологий по сравнению с традиционной разработкой

	Традиционная разработка	CASE
1	Основные усилия – на кодирование и тестирование	Основные усилия – на анализ и проектирование
2	«Бумажные» спецификации	Быстрое итеративное прототипирование
3	Ручное кодирование	Автоматическая кодогенерация
4	Ручное документирование	Автоматическая генерация документации
5	Тестирование кодов	Автоматический контроль проекта
6	Сопровождение кодов	Сопровождение спецификаций проектирования

На рисунке 1.1 представлены результаты сравнения традиционной разработки программных проектов и разработки с применением CASE-технологий, показывающие уменьшение затрат на проектирование программного обеспечения за счет применения CASE-технологий.

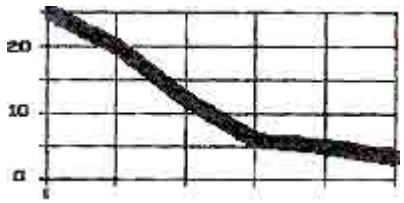

Рисунок 1.1 – Уменьшение затрат на проектирование ПО за счет CASE

1.2.2 Состав и структура и функциональные особенности CASE-инструментария

CASE-инструментарий предназначен для поддержки и усиления методов структурного анализа и проектирования. Эти инструменты поддерживают работу пользователей при создании и редактировании графического проекта в интерактивном режиме, они способствуют организации проекта в виде иерархии уровней абстракции, выполняют проверки соответствия компонентов. Фактически CASE-средства представляют собой новый тип графически–ориентированных инструментов, восходящих к системе поддержки ЖЦ ПО. Обычно к ним относят любое программное средство, обеспечивающее автоматическую помощь при разработке ПО, его сопровождении или деятельности по управлению проектом, и проявляющее следующие дополнительные черты:

– мощная графика для описания и документирования систем ПО, а также для улучшения интерфейса с пользователем, развивающая творческие возможности специалистов и не отвлекающая их от процесса проектирования на решение второстепенных вопросов;

– интеграция, обеспечивающая легкость передачи данных между средствами и позволяющая управлять всем процессом проектирования и разработки ПО непосредственно через процесс планирования проекта;

– использование компьютерного хранилища (репозитария) для всей информации о проекте, которая может разделяться между разработчиками и исполнителями как основа для автоматического продуцирования ПО и повторного его использования в будущих системах.

Помимо перечисленных основополагающих принципов графической ориентации, интеграции и локализации всей проектной информации в репозитарии в основе концептуального построения CASE-инструментария лежат следующие положения:

1) человеческий фактор, определяющий разработку ПО как легкий, удобный и экономичный процесс;

2) широкое использование базовых программных средств, получивших массовое распространение в других приложениях (БД и СУБД, компиляторы с различных языков программирования, отладчики, документаторы, издательские системы, оболочки экспертных систем и базы знаний, языки четвертого поколения и др.);

3) автоматизированная или автоматическая кодогенерация, выполняющая несколько видов генерации кодов: преобразования для получения документации, формирования БД, ввода / модификации данных, получения выполняемых машинных кодов из спецификаций

ПО, автоматической сборки модулей из словарей и моделей данных и повторно используемых программ, автоматической конверсии ранее используемых файлов в форматы новых требований;

4) ограничение сложности, позволяющее получать компоненты, поддающиеся управлению, обозримые и доступные для понимания, а также обладающие простой и ясной структурой;

5) доступность для разных категорий пользователей;

6) рентабельность;

7) сопровождаемость, обеспечивающая способность адаптации применения требований и целей проекта.

Интегрированный CASE-пакет содержит четыре основные компоненты, краткое описание которых приведено ниже.

1 Средства централизованного хранения всей информации о проектируемом ПО в течении всего ЖЦ (репозитарий) являются основой CASE-пакета. Соответствущая БД должна иметь возможность поддерживать большую систему описании и характеристик и предусматривать надежные меры по защите от ошибок и потерь информации. Репозитарий должен обеспечивать:

– инкрементный режим при вводе описаний объектов;

– распространение действия нового или скорректированного описания на информационное пространство всего проекта;

– синхронизацию поступления информации от различных пользователей;

– хранение версий проекта и его отдельных компонент;

– сборку любой запрошенной версии;

– контроль информации на корректность, полноту и состоятельность.

2 Средства ввода предназначены для ввода данных в репозитарий, а также для организации взаимодействия с CASE-пакетом. Эти средства должны поддерживать различные методологии и использоваться на всем ЖЦ разными категориями разработчиков: аналитиками, проектировщиками, инженерами, администраторами и т. д.

3 Средства анализа, проектирования и разработки предназначены для того, чтобы обеспечить планирование и анализ различных описаний, а также их преобразования в процессе разработки.

4 Средства вывода служат для документирования, управления проектом и кодовой генерации.

Все перечисленные компоненты в совокупности должны: поддерживать графические модели; контролировать ошибки; организовывать и поддерживать репозитарий; поддерживать процесс проектирования и разработки.

1.2.3 Поддержка графических моделей

Графическая ориентация CASE заключается в том, что программы являются схематическими проектами и формами, которые много проще в использовании, чем многостраничные описания. Для представления программ применяются структурные диаграммы различных типов, дополнительное достоинство которых заключается в их использовании в качестве наглядной «двумерной» документации по проекту.

Для CASE существенны 4 типа диаграмм: диаграммы функционального проектирования (для этих целей наиболее часто употребляются DFD – диаграммы потоков данных), диаграммы моделирования данных (как правило, ERD – диаграммы «сущность–связь»), диаграммы моделирования поведения (как правило, STD – диаграммы переходов состояний) и структурные диаграммы (карты), применяющиеся на этапе проектирования и описывающие отношения между модулями и внутримодульную структуру. Создание и модификация подобных диаграмм осуществляется с помощью специальных графических редакторов (диаграммеров), являющихся сервисными средствами на этапах анализа требований и проектирования спецификаций. Современные диаграммеры обеспечивают:

– создание иерархически связанных диаграмм, в которых комбинируются графические и текстовые объекты;

– создание и редактирование объектов в любом месте диаграммы;

– создание, перемещение и выравнивание групп объектов, изменение их размеров, масштабирование;

– сохранение связей между объектами при их перемещении и изменении размеров;

– автоматический контроль ошибок и др.

Реализация подобных возможностей позволяет пользователю целиком сосредоточиться на собственно проектировании, не отвлекаясь на решение второстепенных вопросов, связанных с размещением элементов диаграмм, их компоновкой и т. п.

Полученные диаграммы дают ясное понимание и решение проблемы, позволяют проанализировать функционирование создаваемого ПО, фиксируют связи между разработчиками, пользователями и руководителями, обеспечивают стандартизацию представления структуры программы и данных.

Важность **контроля ошибок** на этапах анализа требований и проектирования спецификаций обуславливается возможностью их автоматического обнаружения на ранних этапах ЖЦ. CASE обеспечивает автоматическую верификацию и контроль проекта на полноту и

состоятельность на ранних этапах ЖЦ, что влияет на успех разработки в целом. В подтверждение этого можно привести следующие статистические данные, основанные на отчетах фирмы TRW по анализу 5 крупных проектов [2]:

– при традиционной организации работ ошибки проектирования и кодирования составляют, соответственно, 64% и 32% от общего числа ошибок;

– ошибки проектирования в 100 раз труднее обнаружить на этапе сопровождения ПО, чем на этапах анализа требований и проектирования спецификаций.

В CASE диаграммеры и верификаторы способны осуществлять следующие типы контроля:

1 *Контроль синтаксиса диаграмм и типов их элементов.* Обычно такой контроль осуществляется при вводе и редактировании элементов диаграмм. Примеры контролируемых ситуаций:

– *по синтаксису:* любой функциональный элемент диаграммы должен иметь по крайней мере один входной и один выходной поток; два элемента данных не могут быть непосредственно связаны;

– *по типам:* функциональный элемент должен всегда использоваться для представления процедурной компоненты; поток данных всегда должен быть представлен компонентой данных.

2 *Контроль полноты и состоятельности диаграмм:* все элементы диаграмм должны быть идентифицированы и отражены в репозитарии. Например, для DFD контролируются неименованные или несвязанные потоки данных, процессы и хранилища данных; источники и стоки данных (внешние сущности) вне контекстной диаграммы; хранилища данных на контекстной диаграмме и т. п. При анализе словаря данных необходимо выявлять циклические определения, эквивалентные определения, неопределенные объекты.

3 *Контроль декомпозиции функций* включает оценку качества на основе различных метрик ПО и частичный семантический контроль.

4 Сквозной контроль диаграмм одного или различных типов на предмет их состоятельности по уровням *вертикальное и горизонтальное балансирование диаграмм.* При вертикальном балансировании (диаграммы одного типа) выявляются несбалансированные потоки данных между детализируемой и детализирующей диаграммами. Горизонтальное балансирование определяет некорректности между DFD, ERD, STD, словарями данных и миниспецификациями процессов. Так при балансировании DFD-ERD контролируется соответствие каждого хранилища данных на DFD сущности или отношению на ERD. Контроль DFD-STD осуществляется по следующим правилам:

каждый управляющий процесс на DFD детализируется спецификацией управления STD, и наоборот, каждой STD должен соответствовать управляющий процесс; каждое условие (действие) в STD должно соответствовать входному (выходному) управляющему потоку на DFD, и наоборот, каждому управляющему потоку в зависимости от его направленности должно соответствовать условие / действие на STD. При балансировании DFD-словаря данных и миниспецификаций процессов должно проверяться следующее:

– каждый поток и хранилище данных должны быть определены в словаре данных (контроль неопределенных значений), и наоборот, каждое определение в словаре должно быть отражено на диаграмме, в миниспецификации или другом определении (контроль неиспользуемых значений);

– каждый процесс на DFD должен детализироваться с помощью DFD или миниспецификации (но не тем и другим одновременно), и наоборот, каждая миниспецификация должна соответствовать единственному процессу;

– ссылки к данным в миниспецификациях должны соответствовать объектам на диаграммах и в словаре данных;

– по возможности должна контролироваться семантика миниспецификации: например, если входные и / или выходные потоки связаны с хранилищем данных, то это должно быть отражено в миниспецификации (операторами READ, GET, WRITE, PUT и т. п.).

1.2.4 Поддержка процесса проектирования и разработки

При поддержке процесса проектирования и разработки основную роль играют следующие возможности CASE-пакетов: покрытие ЖЦ, поддержка прототипирования, поддержка структурных методологий, автоматическая кодогенерация.

При *покрытии ЖЦ* наибольшее внимание уделяется его наиболее критичным этапам – анализу требований и проектированию спецификаций. Последние являются основой всего проекта, поэтому их полнота и корректность влияют на успех разработки в целом.

Важную роль при автоматизации ранних этапов ЖЦ играют возможности *поддержки прототипирования.* Соответствующие средства используются для определения системных требований и ответа на вопросы об ожидаемом поведении системы. Такие средства как генераторы меню, экранов и отчетов позволяют быстро построить прототипы пользовательских интерфейсов и снабдить моделью функционирования

системы с позиций конечного пользователя. Использование языков четвертого поколения (4GL) позволяет строить более сложные модели, при этом прототип позволяет промоделировать основные функции системы, но не способен контролировать ее ожидаемое поведение. Исполняемые языки спецификаций преобразуют процесс разработки в следующий итеративный процесс: спецификации определяются и выполняются, затем производится переопределение или корректировка. Созданные таким образом прототипы позволяют определять, является ли проектируемая система полной и корректной.

Поддержка структурных методологий осуществляется за счет средств их автоматизации на следующих двух уровнях:

– подготовка документации, графическая поддержка построения структурных диаграмм различных типов, продуцирование спецификаций для детализации функциональных блоков в диаграммах и структур данных на нижних уровнях (для таких спецификаций введен специальный термин – «миниспецификация»);

– корректное использование шагов обработки в методологиях.

Кодогеперация осуществляется на основе репозитария и позволяет автоматически построить до 80–90% объектных кодов или текстов программ на языках высокого уровня. При этом различными CASE-пакетами поддерживаются практически все известные языки программирования, однако наиболее часто в качестве целевых языков выступают COBOL, C и ADA. Средства кодогенерации по отношению к полноте целевого продукта разделяются на средства генерации каркаса ПО и средства генерации полного продукта. В первом случае автоматически строится откомментированная логика (потоки управления) ПО, а также коды для БД, файлов, экранов, отчетов и т. п., остальные фрагменты ПО кодируются вручную. Во втором случае из проектных спецификаций генерируется полная документированная программа, включая выполняемый код, пользовательскую и программную документацию, наборы тестов и т. д. Все эти компоненты полной программы связываются в единый объект, хранящийся в репозитарии для облегчения доступа и сопровождения.

Идея автоматической кодогенерации на основе модели заключается в следующем. Любая программа схематически может быть представлена в виде тройки: обрабатываемые **данные, логический** каркас обработки, **линейные** участки обработки. Схема базы данных может быть легко сгенерирована на основании модели «сущность–связь», и современные средства информационного моделирования (например, ERWin, Designer/2000 и др.) обеспечивают такую генерацию. Логика обработки генерируется на основе диаграмм потоков данных: известны

алгоритмы автоматического преобразования иерархии DFD в структурные карты, а с задачей получения из структурных карт соответствующих кодов легко справляется теория компиляции. Наконец, линейным участкам соответствуют миниспецификации модели, и именно здесь лежит **ключ** к высокому проценту автоматически сгенерированного кода, существенно зависящему от метода задания миниспецификаций.

Вопросы и задания для самоконтроля

1 Приведите этапы реализации простейшей модели ЖЦ и в соответствующей ей CASE-модели ЖЦ.

2 Назовите наиболее автоматизируемые фазы ЖЦ ПО в CASE-модели ЖЦ.

3 Сравните традиционную разработку программных проектов и разработки с применением CASE-технологий.

4 ПС с какими возможностями относят к CASE-инструментарию?

5 Какие положения лежат в основе концептуального построения CASE-инструментария?

6 Каковы основные компоненты интегрированного CASE-инструментария?

7 Какие наиболее важные типы диаграмм для CASE-инструментария?

8 Что обеспечивают диаграммеры на этапах анализа требований и проектирования спецификаций?

9 Ошибки каких этапов ЖЦ ПО составляют большую долю?

10 Поддержку какого контроля осуществляют диаграммеры и верификаторы CASE-инструментария?

11 Какие возможности CASE-пакетов при поддержке процесса проектирования и разработки играют основную роль?

12 CASE-инструментарий 1 или 2 позволяет выполнить покрытие ЖЦ?

13 Что такое поддержка прототипирования?

14 На каких уровнях за счет CASE-инструментария осуществляется поддержка структурных методологий?

15 В чём заключается идея автоматической кодогенерации? Изобразите схематично.

2 Стандарты документирования программных средств

2.1 Общая характеристика проблем и задач документирования программного обеспечения

2.1.1 Проблемы и задачи создания программной документации

Создание программной документации – важный этап, так как пользователь начинает свое знакомство с программным продуктом именно с документации. Для чего предназначен программный продукт, как его установить, как начать с ним работать – вот одни из первых вопросов, на которые должна отвечать программная документация (Installation Guide, Getting Started). Составлением программной документации обычно занимаются специальные люди – технические писатели (иногда программную документацию пишут сами программисты или аналитики). Этот этап является самым неприятным и тяжелым в программистской работе. К сожалению, обычно этому либо не учат совсем, либо в лучшем случае не обращают на качество получаемых документов должного внимания. Тем не менее владение этим искусством является одним из важнейших факторов, определяющих качество программиста.

Умение создавать программную документацию определяет профессиональный уровень программиста. Заказчик не будет вникать в тонкости и особенности даже самой замечательной программы. Заказчик будет сначала читать документацию. Большую роль играет в этом и психологический фактор.

Грамотно созданный пакет программной документации избавит разработчиков ПО от многих неприятностей, в частности, от назойливых вопросов и необоснованных претензий, отослав пользователя к документации. Это касается прежде всего важнейшего документа – Технического задания. Многомиллионный иск в 70-х гг. 20 в., предъявленный к компании IBM крупным издательством о неудовлетворительном качестве вычислительной техники и программного обеспечения, был в суде выигран благодаря подписанному обеими сторонами Техническому заданию.

Вообще программную документацию можно разделить **по отношению** к пользователю на внутреннюю и внешнюю. К внешней документации относят всевозможные руководства для пользователей, техническое задание, справочники. Внутренняя документация – это документация, которая используется в процессе разработки программного обеспечения и недоступна конечному пользователю (различные внутренние стандарты, комментарии исходного текста, технологии программирования и т. д.).

Когда программист-разработчик получает в той или иной форме задание на программирование, перед ним, перед руководителем проекта и перед всей проектной группой встают вопросы: Что должно быть сделано, кроме собственно программы? Что и как должно быть оформлено в виде документации? Что передавать пользователям, а что – службе сопровождения? Как управлять всем этим процессом? Что должно входить в само задание на программирование?

На эти и другие вопросы когда-то отвечали государственные стандарты на программную документацию – комплекс стандартов 19-й серии ГОСТ ЕСПД. Но уже тогда у программистов была масса претензий к этим стандартам. Что-то требовалось дублировать в документации много раз (как оказалось – неоправданно), а многое не было предусмотрено, как, например, отражение специфики документирования программ, работающих с интегрированной базой данных.

2.1.2 Общая характеристика состояния в области документирования программных средств

Основу отечественной нормативной базы в области документирования ПС составляет комплекс стандартов Единой системы программной документации (ЕСПД). Основная и большая часть комплекса ЕСПД была разработана в 70-е и 80-е годы 20 века. Сейчас этот комплекс представляет собой систему межгосударственных стандартов стран СНГ (ГОСТ), действующих также и на территории Беларуси, на основе межгосударственного соглашения по стандартизации.

Единая система программной документации – это комплекс государственных стандартов, устанавливающих взаимоувязанные правила разработки, оформления и обращения программ и программной документации.

Стандарты ЕСПД в основном охватывают ту часть документации, которая создается в процессе разработки ПС, и связаны, по большей

части, с документированием функциональных характеристик ПС. Следует отметить, что стандарты ЕСПД (ГОСТ 19) носят рекомендательный характер. Впрочем, это относится и ко всем другим стандартам в области ПС (ГОСТ 34, международному стандарту ISO/IEC и др.). Дело в том, что в соответствии с Законом «О стандартизации» эти стандарты становятся обязательными на контрактной основе, т. е. при ссылке на них в договоре на разработку (поставку) ПС.

В состав ЕСПД входят:

– основополагающие и организационно-методические стандарты;

– стандарты, определяющие формы и содержание программных документов, применяемых при обработке данных;

– стандарты, обеспечивающие автоматизацию разработки программных документов.

Вопросы и задания для самоконтроля

1 На какие вопросы даёт ответ программная документация?

2 Кто занимается разработкой программной документации?

3 Что такое внешняя и внутренняя документация и каковы их цели?

4 На какие вопросы даёт ответ комплекс стандартов 19-й серии ГОСТ?

5 Что такое ЕСПД?

6 С документированием каких характеристик ПС связаны в основном стандарты ЕСПД?

7 В каком случае стандарт ЕСПД является регламентом?

8 Каков структурный состав ЕСПД?

2.2 Единая система программной документации

2.2.1 Основные недостатки и сильные стороны ЕСПД

Говоря о состоянии ЕСПД в целом, можно констатировать, что большая часть стандартов ЕСПД морально устарела. К числу основных недостатков ЕСПД можно отнести:

– ориентацию на единственную «каскадную» модель жизненного цикла ПС;

– отсутствие четких рекомендаций по документированию характеристик качества ПС;

– отсутствие системной увязки с другими действующими отечественными системами стандартов по ЖЦ и документированию продукции в целом, например ЕСКД[1)];

– нечетко выраженный подход к документированию ПС как товарной продукции;

– отсутствие рекомендаций по самодокументированию ПС, например, в виде экранных меню и средств оперативной помощи пользователю (хелпов);

– отсутствие рекомендаций по составу, содержанию и оформлению перспективных документов на ПС, согласованных с рекомендациями международных и региональных стандартов.

Из перечисленных выше недостатков ЕСПД очевидно, что она нуждается в полном пересмотре на основе стандарта ИСО/МЭК 12207-95 на процессы жизненного цикла ПС. Тем не менее до пересмотра всего комплекса многие стандарты могут с пользой применяться в практике документирования ПС. Эта позиция основана на следующем:

– стандарты ЕСПД вносят элемент упорядочения в процесс документирования ПС;

– предусмотренный стандартами ЕСПД состав программных документов вовсе не такой «жесткий», как некоторым кажется: стандарты позволяют вносить в комплект документации на ПС дополнительные виды программных документов (ПД), необходимых в конкретных проектах, и исключать многие ПД;

– стандарты ЕСПД позволяют вдобавок мобильно изменять структуры и содержание установленных видов ПД исходя из требований заказчика и пользователя.

При этом стиль применения стандартов может соответствовать современному общему стилю адаптации стандартов к специфике проекта: заказчик и руководитель проекта выбирают уместное в проекте подмножество стандартов и ПД, дополняют выбранные ПД нужными разделами и исключают ненужные, привязывают создание этих документов к той схеме ЖЦ, которая используется в проекте.

Надо сказать, что наряду с комплексом ЕСПД официальная нормативная база СНГ в области документирования ПС и в смежных областях включает ряд перспективных стандартов (отечественного, межгосударственного и международного уровней). Международный стандарт ISO/IEC 12207:1995 на организацию ЖЦ продуктов ПО, казалось бы, весьма неконкретный, но вполне новый и отчасти «модный» стандарт.

[1)] Единая система конструкторской документации

2.2.2 Общая характеристика Единой системы программной документации

Стандарты ЕСПД, как и другие ГОСТы, подразделяют на группы, которые приведены в таблице 2.1.

Обозначение стандарта ЕСПД должно состоять из:

– числа 19 (присвоенных классу стандартов ЕСПД);

– одной цифры (после точки), обозначающей код классификационной группы стандартов, указанной в таблице 2.1;

– двух цифр, обозначающих порядковый номер документа в группе стандартов соответствующего кода;

– двузначного числа (после тире), указывающего год регистрации стандарта.

Таблица 2.1 – Группы стандартов ЕСПД

Код группы	Наименование группы
0	Общие положения
1	Основополагающие стандарты
2	Правила выполнения документации разработки
3	Правила выполнения документации изготовления
4	Правила выполнения документации сопровождения
5	Правила выполнения эксплуатационной документации
6	Правила обращения программной документации
7–8	Резервные группы
9	Прочие стандарты

Вообще перечень документов ЕСПД очень обширен.

Примечание. – В него, в частности, входят следующие ГОСТы:

ГОСТ 19.001-77 ЕСПД. Общие положения.

ГОСТ 19.005-85 ЕСПД. Р–схемы алгоритмов и программ. Обозначения услов-ные графические и правила выполнения.

ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.

ГОСТ 19.102-77 ЕСПД. Стадии разработки.

ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов.

ГОСТ 19.104-78 ЕСПД. Основные надписи.

ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам.

ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом.

ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению.

ГОСТ 19.202-78 ЕСПД. Спецификация. Требования к содержанию и оформлению.

ГОСТ 19.301-79 ЕСПД. Порядок и методика испытаний.

ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению.

ГОСТ 19.402-78 ЕСПД. Описание программы.

ГОСТ 19.403-79 ЕСПД. Ведомость держателей подлинников.

ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.

ГОСТ 19.501-78 ЕСПД. Формуляр. Требования к содержанию и оформлению.

ГОСТ 19.502-78 ЕСПД. Описание применения. Требования к содержанию и оформлению.

ГОСТ 19.503-79 ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению.

ГОСТ 19.504-79 ЕСПД. Руководство программиста. Требования к содержанию и оформлению.

ГОСТ 19.505-79 ЕСПД. Руководство оператора. Требования к содержанию и оформлению.

ГОСТ 19.506-79 ЕСПД. Описание языка. Требования к содержанию и оформлению.

ГОСТ 19.507-79 ЕСПД. Ведомость эксплуатационных документов.

ГОСТ 19.508-79 ЕСПД. Руководство по техническому обслуживанию. Требования к содержанию и оформлению.

ГОСТ 19.601-78 ЕСПД. Общие правила дублирования, учета и хранения.

ГОСТ 19.602-78 ЕСПД. Правила дублирования, учета и хранения программных документов, выполненных печатным образом.

ГОСТ 19.603-78 ЕСПД. Общие правила внесения изменений.

ГОСТ 19.604-78 ЕСПД. Правила внесения изменений в программные доку-менты, выполняемые печатным способом.

ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.

ГОСТ 19781-90. Обеспечение систем обработки информации программное. Термины и определения. ГОСТ 19.104-78 ЕСПД. Основные надписи.

ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам.

ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом.

Прежде чем приступить к рассмотрению правил составления программной документации, необходимо сделать следующее замечание. Каждый документ желательно предварять некоторым введением.

Во введении говорится об актуальности, о необходимости составления и т. п. Цель исполнителя здесь – показать значимость и необходимость выполнения этой работы.

Остановимся на некоторых стандартах ЕСПД, наиболее часто используемых на практике.

Первым укажем стандарт, который устанавливает виды программ и программных документов для вычислительных машин, комплексов и систем независимо от их назначения и области применения.

2.2.3 Виды программ и программных документов (ГОСТ 19.101-77 ЕСПД)

ГОСТ подразделяет программы на следующие виды: компонент и комплекс. Компонент – программа, рассматриваемая как единое целое, выполняющая законченную функцию и применяемая самостоятельно или в составе комплекса. Комплекс – программа, состоящая из двух или более компонентов, выполняющих взаимосвязанные функции, и применяемая самостоятельно или в составе другого комплекса.

Документация, разработанная на программу, может использоваться для реализации и передачи программы на носителях данных, а также для изготовления программного изделия. К числу программных данный ГОСТ относит документы, содержащие сведения, необходимые для разработки, изготовления, сопровождения и эксплуатации программ. Рассмотрим виды программных документов и их содержание.

Спецификация – содержит состав программы и документацию на нее.

Ведомость держателей подлинников – содержит перечень предприятий, на которых хранят подлинники программных документов.

Текст программы – представляет запись программы с необходимыми комментариями.

Описание программы – содержит сведения о логической структуре и функционировании программы.

Программа и методика испытаний – содержит требования, подлежащие проверке при испытании программы, а также порядок и методы их контроля.

Техническое задание – описывает назначение и область применения программы, технические, технико-экономические и специальные

требования, предъявляемые к программе, необходимые стадии и сроки разработки, виды испытаний.

Пояснительная записка – содержит схему алгоритма, общее описание алгоритма и (или) функционирования программы, а также обоснование принятых технических и технико-экономических решений.

Эксплуатационные документы – содержат сведения для обеспечения функционирования и эксплуатации программы.

В зависимости от способа *выполнения* и характера *применения* программные документы подразделяются на подлинник, дубликат и копию (ГОСТ 2.102-68), предназначенные для разработки, сопровождения и эксплуатации программы.

Допускается объединять отдельные виды эксплуатационных документов (за исключением ведомости эксплуатационных документов и формуляра). Необходимость объединения этих документов указывается в техническом задании. Объединенному документу присваивают наименование и обозначение одного из объединяемых документов. В объединенных документах должны быть приведены сведения, которые необходимо включать в каждый объединяемый документ.

2.2.4 Стадии разработки

ГОСТ 19.102-77 ЕСПД устанавливает стадии разработки программ и программной документации для вычислительных машин, комплексов и систем независимо от их назначения и области применения (таблица 2.2).

В соответствии с документами допускается исключать вторую стадию разработки, а в технически обоснованных случаях – вторую и третью стадии. Необходимость проведения этих стадий указывается в техническом задании.

Допускается объединять, исключать этапы работ и (или) их содержание, а также вводить другие этапы работ *по согласованию* с заказчиком.

2.2.5 Краткая характеристика некоторых ГОСТов по программной документации

Общие требования к программным документам (ГОСТ 19.105-78 ЕСПД). Данный стандарт устанавливает общие требования к оформлению программных документов для вычислительных машин,

комплексов и систем независимо от их назначения и области применения и предусмотренных стандартами Единой системы программной документации (ЕСПД) для любого способа выполнения документов на различных носителях данных.

Таблица 2.2 – Стадии разработки, этапы и содержание работ

Стадия разработки	Этап работы	Содержание работ
1	2	3
Техническое задание	Обоснование необходимости разработки программы	Постановка задачи. Сбор исходных материалов. Выбор и обоснование критериев эффективности и качества разрабатываемой программы. Обоснование необходимости проведения научно-исследовательских работ
	Научно-исследовательские работы	Определение структуры входных и выходных данных. Предварительный выбор методов решения задач. Обоснование целесообразности применения ранее разработанных программ. Определение требований к техническим средствам. Обоснование принципиальной возможности решения поставленной задачи
	Разработка и утверждение технического задания	Определение требований к программе. Технико-экономическое обоснование разработки программы. Определение стадий, этапов и сроков разработки программы и документации на нее. Выбор языков программирования. Определение необходимости проведения научно-исследовательских работ на последующих стадиях. Согласование и утверждение технического задания
Эскизный проект	Разработка эскизного проекта	Предварительная разработка структуры входных и выходных данных. Уточнение методов решения задачи. Разработка общего описания алгоритма решения задачи. Разработка технико-экономического обоснования
	Утверждение эскизного проекта	Разработка пояснительной записки. Согласование и утверждение эскизного проекта

Окончание таблицы 2.2

1	2	3
Технический проект	Разработка технического проекта	Уточнение структуры входных и выходных данных. Разработка алгоритма решения задачи. Определение формы представления входных и выходных данных. Определение семантики и синтаксиса языка. Разработка структуры программы. Окончательное определение конфигурации технических средств
	Утверждение технического проекта	Разработка плана мероприятий по разработке и внедрению программ. Разработка пояснительной записки. Согласование и утверждение технического проекта
	Разработка программы	Программирование и отладка программы
	Разработка программной документации	Разработка программных документов в соответствии с требованиями ГОСТ 19.101-77
Рабочий проект	Испытания программы	Разработка, согласование и утверждение программы и методики испытаний. Проведение предварительных государственных, межведомственных, приемо-сдаточных и других видов испытаний. Корректировка программы и программной документации по результатам испытаний
Внедрение	Подготовка и передача программы	Подготовка и передача программы и программной документации для сопровождения и (или) изготовления. Оформление и утверждение акта о передаче программы на сопровождение и (или) изготовление. Передача программы в фонд алгоритмов и программ

Программный документ может быть представлен на различных типах носителей данных и состоит из следующих условных частей:

- титульной;
- информационной;
- основной.

Правила оформления документа и его частей на каждом носителе данных устанавливаются стандартами ЕСПД на правила оформления документов на соответствующих носителях данных. Титульная часть оформляется согласно ГОСТ 19.104-78. Информационная часть должна состоять из аннотации и содержания. В аннотации приводят сведения

о назначении документа и краткое изложение основной части. Содержание включает перечень записей о структурных элементах основной части документа. Состав и структура основной части программного документа устанавливаются стандартами ЕСПД на соответствующие документы.

Техническое задание. Требования к содержанию и оформлению (ГОСТ 19.201-78 ЕСПД). Техническое задание (ТЗ) содержит совокупность требований к ПС и может использоваться как критерий проверки и приемки разработанной программы. Поэтому достаточно полно составленное (с учетом возможности внесения дополнительных разделов) и принятое заказчиком и разработчиком ТЗ является одним из основополагающих документов проекта ПС.

Техническое задание должно содержать следующие разделы:

– введение;
– основания для разработки;
– назначение разработки;
– требования к программе или программному изделию;
– требования к программной документации;
– технико-экономические показатели;
– стадии и этапы разработки;
– порядок контроля и приемки;

В техническое задание допускается включать приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

Рассмотрим требования к программной документации. В данном разделе должны быть указаны предварительный состав программной документации и при необходимости специальные требования к ней.

Описание программы (ГОСТ 19.402-78 ЕСПД). Данный стандарт определяет состав и требования к содержанию программного документа «Описание программы».

Описание программы включает:

1 Общие сведения.
2 Функциональное назначение.
3 Описание логической структуры.
4 Используемые технические средства.
5 Вызов и загрузку.
6 Входные данные.
7 Выходные данные.

В разделе Общие сведения указывают:

– обозначение и наименование программы;

– программное обеспечение, необходимое для функционирования программы;

– языки программирования, на которых написана программа.

Раздел «Функциональное назначение» должен отражать классы решаемых задач и / или назначение программы, сведения о функциональных ограничениях на применение.

При описании логической структуры должны быть отражены:

– алгоритм программы;

– используемые методы;

– структура программы с описанием функций составных частей и связей между ними;

– связи программы с другими программами.

В разделе «Используемые технические средства» указывают типы ЭВМ и устройств, которые используются при работе программы.

При описании раздела «Вызов и загрузка» указывают способ вызова программы с соответствующего носителя данных и входные точки в программу.

Раздел «Входные данные» отражает:

– характер, организацию и предварительную подготовку входных данных;

– формат, описание и способ кодирования входных данных.

Раздел «Выходные данные» отражает:

– характер и организацию выходных данных;

– формат, описание и способ кодирования выходных данных.

Пояснительная записка. (ГОСТ 19.404-79 ЕСПД). Требования к содержанию и оформлению.

Согласно данному стандарту пояснительная записка должна включать следующие разделы:

1 Введение.

2 Назначение и область применения.

3 Технические характеристики.

4 Ожидаемые технико-экономические показатели.

5 Источники, использованные при разработке.

Введение должно содержать наименование программы и / или обозначение темы разработки, а также документы, на основе которых ведется разработка.

При описании назначения и области применения указывают назначение программы, краткую характеристику области применения программы.

В разделе «Технические характеристики» содержатся:

– постановка задачи на разработку программы, описание применяемых математических методов и различных ограничений, связанных с выбранным математическим аппаратом;

– описание алгоритма и / или функционирования программы с обоснованием выбора схемы алгоритма решения задачи, возможного взаимодействия программы с другими программами;

– описание и обоснование выбора метода организации входных и выходных данных;

– описание и обоснование выбора состава технических и программных средств на основе проведенных расчетов и анализов, распределение носителей данных, которые использует программа.

В качестве ожидаемых технико-экономических показателей указывают показатели, обосновывающие преимущество выбранного варианта технического решения, а также при необходимости ожидаемые оперативные показатели.

При описании источников, использованных при разработке, необходимо привести перечень научно-технических публикаций, нормативно-технических документов и других научно-технических материалов, на которые есть ссылки в основном тексте.

Руководство системного программиста. Требования к содержанию и оформлению (ГОСТ 19.503-79 ЕСПД). Руководство системного программиста должно содержать следующие разделы:

1 Общие сведения о программе.
2 Структура программы.
3 Настройка программы.
4 Проверка программы.
5 Дополнительные возможности.
6 Сообщения системному программисту.

При необходимости допускается опускать раздел, описывающий дополнительные возможности.

При описании общих сведений о программе необходимо указать назначение и функции программы и сведения о технических и программных средствах, обеспечивающих выполнение данной программы.

В разделе «Структура программы» приводятся сведения о структуре программы, ее составных частях и связях с другими программами.

Раздел «Настройка программы» должен содержать описание действий по настройке программы на условия конкретного применения.

При описании проверки программы необходимо привести и описать способы проверки, позволяющие дать общее заключение о работоспособности программы (контрольные примеры, методы прогона, результаты).

Раздел «Дополнительные возможности» должен содержать описание дополнительных разделов функциональных возможностей программы и способов их выбора.

В разделе «Сообщения системному программисту» необходимо указать тексты сообщений, выдаваемых в ходе выполнения программы, описание содержания и действий, которые необходимо предпринять по этим сообщениям.

Руководство программиста. Требования к содержанию и оформлению (ГОСТ 19.504-79 ЕСПД).

Руководство программиста должно содержать разделы:

1 Назначение и условия применения программы.
2 Характеристики программы.
3 Обращение к программе.
4 Входные и выходные данные.
5 Сообщения.

При описании назначения и условий применения программы необходимо указать назначение и функции, выполняемые программой; условия, необходимые для выполнения программы: объем оперативной памяти, требования к составу и параметрам периферийных устройств; требования к ПО и т. д.

В разделе «Характеристики программы» необходимо привести описание основных характеристик и особенностей программы: временных характеристик, режима работы, средств контроля правильности выполнения и самовосстанавливаемости программы и т. д.

Раздел «Обращение к программе» представляет собой описание процедур вызова программы (способов передачи управления и параметров данных и др.).

Раздел «Входные и выходные данные» должен содержать описание организации используемой входной и выходной информации и при необходимости ее кодирования.

При описании сообщений необходимо привести тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы, описание их содержания и действия, которые необходимо предпринять по этим сообщениям.

2.2.6 Документация пользователя

Документация пользователя (user documentation): полный комплект документов, поставляемых в печатном или другом виде, который обеспечивает применение продукта, а также является его неотъемлемой частью.

Документация пользователя должна отвечать следующим характеристикам.

Полнота (completeness). Документация пользователя должна содержать информацию, необходимую для использования продукта. В ней должны быть полностью описаны все функции, установленные в описании продукта, и все вызываемые пользователем функции из программы. Кроме того, граничные значения, заданные в описании продукта, должны быть продублированы в документации пользователя. Если установка (инсталляция) продукта может быть проведена пользователем, то в документацию пользователя должно быть включено руководство по установке продукта, содержащее всю необходимую информацию. Если сопровождение продукта может проводиться пользователем, то в документацию пользователя должно быть включено руководство по сопровождению программы, содержащее всю информацию, которая необходима для обеспечения данного вида сопровождения.

Правильность (correctness). Документация пользователя не должна содержать неоднозначных толкований и ошибок.

Непротиворечивость (consistency). Документы, входящие в комплект документации пользователя, не должны противоречить сами себе, друг другу и описанию продукта. Каждый термин должен иметь один и тот же смысл во всех смежных документах и пунктах.

Понятность (understandability). Документация пользователя должна быть понятной для сообщества пользователей, выполняющих указанную рабочую задачу, например, посредством использования в ней соответствующим образом подобранных терминов, графических вставок, уточняющих пояснений и путем ссылок на полезные источники информации.

Простота обозрения (ease of overview). Документация пользователя должна быть достаточно проста для изучения пользователем, чтобы он мог выявить все описываемые в ней взаимосвязи компонентов продукта. В каждый документ могут быть включены оглавление и предметный указатель.

Качество характеристик документации пользователя можно оценить. Например, если программный продукт позволяет выполнить десять операций или функций и две из них описаны непонятно, то оценкой понятности может быть число 0.8, полученное исходя из следующих соображений: в числителе дроби число 8 (число функций, описанных в документации понятно), а в знаменателе – число 10 (общее число функций).

Резюмируя, скажем, что возникла настоятельная потребность во введении в отечественные стандарты на документирование ПС тех

норм, правил, требований и рекомендаций, которые установлены на международном и передовом зарубежном уровнях. Но при проведении этих работ нельзя ограничиваться прямым переводом отдельных международных стандартов. Нужно, чтобы новые стандарты правильно стыковались со всем имеющимся и будущим множеством нормативных документов.

Вопросы и задания для самоконтроля

1 Каковы плюсы и минусы ЕСПД?
2 Что такое ЕСКД?
3 Как принято обозначать стандарт ЕСПД?
4 Какие виды программ выделяют?
5 Какие виды программных документов существуют?
6 Каково содержание программных документов?
7 Наряду с видами программ и программных документов по ГОСТ 19.101-77 ЕСПД какие ещё существуют подходы к систематизации программных документов и соответствующие им разновидности?
8 Перечислите стадии разработки программ.
9 Каковы обязательные стадии разработки программ?
10 В каких случаях этапы не являются обязательными при разработке программ?
11 Каковы структурные части программных документов?
12 Что содержит и как может использоваться техническое задание?
13 Каковы состав и требования к содержанию программного документа «Описание программы»?
14 Каковы состав и требования к содержанию программного документа «Пояснительная записка»?
15 Каковы состав и требования к содержанию программных документов «Руководство системного программиста» и «Руководство программиста»? В чем их отличие?
16 Что такое «Документация пользователя»? Каким характеристикам она должна отвечать.
17 Приведите список ключевых слов-синонимов для характеристик «Документации пользователя».
18 Как можно измерить характеристики «Документации пользователя»?

Литература

1 Благодатских, В. А. Стандартизация разработки программных средств : учебное пособие / В. А. Благодатских, В. А. Волнин, К. Ф. Поскакалов; под ред. О. С. Разумова. – М.: Финансы и статистика, 2005. – 284 с.

2 Калянов, Г. Н. Case-технологии. Консалтинг при автоматизации бизнес-процессов / Г. Н. Калянов. – 3-е изд. – М.: Горячая линия – Телеком. 2000. – 239 с.

3 Крылова, Г. Д. Основы стандартизации, сертификации, метрологии : учебник для вузов / Г. Д. Крылова. – 2-е изд.. – М.: ЮНИ-ТИ-ДАНА, 2001. – 356 с.

4 Осипенко, Н. Б. Основы стандартизации и сертификации программного обеспечения: тексты лекций для студентов математических специальностей / Н. Б. Осипенко. – Гомель: ГГУ им. Ф. Скорины, 2007. – 137 с.

Производственно-практическое издание

Осипенко Наталья Борисовна,
Осипенко Александр Николаевич

CASE-ИНСТРУМЕНТАРИЙ АВТОМАТИЗАЦИИ АНАЛИЗА, ПРОЕКТИРОВАНИЯ И РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И СТАНДАРТЫ ДОКУМЕНТИРОВАНИЯ ПРОГРАММНЫХ СРЕДСТВ

Практическое руководство

для студентов специальности I–40 01 01
«Программное обеспечение информационных технологий»

Редактор *В. И. Шкредова*
Корректор *В. В. Калугина*

Подписано в печать 15.10.2015. Формат 60x84 1/16.
Бумага офсетная. Ризография. Усл. печ. л. 2,3.
Уч.-изд. л. 2,5. Тираж 25 экз. Заказ 605.

Издатель и полиграфическое исполнение:
учреждение образования
«Гомельский государственный университет имени Франциска Скорины».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/87 от 18.11.2013.
Специальное разрешение (лицензия) № 02330 / 450 от 18.12.2013.
Ул. Советская, 104, 246019, Гомель.

Н. Б. ОСИПЕНКО, А. Н. ОСИПЕНКО

CASE-ИНСТРУМЕНТАРИЙ АВТОМАТИЗАЦИИ АНАЛИЗА, ПРОЕКТИРОВАНИЯ И РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И СТАНДАРТЫ ДОКУМЕНТИРОВАНИЯ ПРОГРАММНЫХ СРЕДСТВ

Гомель
2015