Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра программного обеспечения информационных технологий

В. В. Бахтизин, Л. А. Глухова, С. Н. Неборский

МЕТРОЛОГИЯ, СТАНДАРТИЗАЦИЯ И СЕРТИФИКАЦИЯ В ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЯХ

*Рекомендовано УМО по образованию
в области информатики и радиоэлектроники для специальностей
1-40 01 01 «Программное обеспечение информационных технологий»,
1-40 04 01 «Информатика и технологии программирования»,
направлений специальности
1-40 05 01-02 «Информационные системы и технологии (в экономике)» и
1-40 05 01-08 «Информационные системы и технологии (в логистике)»
в качестве учебно-методического пособия*

Библиотека БГУИР

Минск БГУИР 2013

УДК [006.91+006.1]:004(076)
ББК 30.10я73+30ця73+32.973.26-018.2я73
Б30

Р е ц е н з е н т ы:

кафедра программного обеспечения вычислительной техники
и автоматизированных систем
Белорусского национального технического университета
(протокол №9 от 18.04.2013 г.);

заведующий кафедрой информатики учреждения образования
«Минский государственный высший радиотехнический колледж»,
кандидат технических наук, доцент Ю. А. Скудняков

Бахтизин, В. В.

Б30 Метрология, стандартизация и сертификация в информационных технологиях : учеб.-метод. пособие / В. В. Бахтизин, Л. А. Глухова, С. Н. Неборский. – Минск : БГУИР, 2013. – 60 с. : ил.

ISBN 978-985-488-977-1.

В пособии рассмотрены основные положения действующих национальных и международных стандартов в области оценки качества программных средств. Приведены модели качества систем и программных средств, а также меры качества, регламентированные названными стандартами. Описаны меры сложности программных средств. Даны варианты индивидуальных заданий для выполнения лабораторных работ по оценке сложности программных средств.

УДК [006.91+006.1]:004(076)
ББК 30.10я73+30ця73+32.973.26-018.2я73

ISBN 978-985-488-977-1

© Бахтизин В. В., Глухова Л. А.,
Неборский С. Н., 2013
© УО «Белорусский государственный
университет информатики
и радиоэлектроники», 2013

Библиотека БГУИР

СОДЕРЖАНИЕ

Библиотека БГУИР

1. КАЧЕСТВО ПРОГРАММНЫХ СРЕДСТВ

1.1. Основные понятия и определения в области качества программных средств и систем

Атрибут (attribute) – внутренне присущее объекту свойство, которое может быть распознано количественно или качественно человеком или автоматизированными средствами. В стандартах серии *SQuaRE* понятие атрибута используется аналогично понятию свойства.

Базовая мера (base measure) – мера, определенная в терминах атрибута и метода для ее количественной оценки. Базовая мера функционально не зависит от других мер. В стандартах серии *SQuaRE* понятие базовой меры соответствует понятию элемента меры качества.

Внешняя мера качества программного средства (external measure of software quality) – мера обеспечения программным продуктом поведения системы (включая программное обеспечение (ПО)), удовлетворяющего заданным и подразумеваемым потребностям, при использовании программного продукта в заданных условиях. Например, количество отказов, обнаруженных при тестировании – это внешняя мера качества программного средства (ПС), связанная с количеством ошибок в нем. Две меры количества отказов, измеренные в разное время, могут отличаться друг от друга, поскольку тестирование может не найти всех ошибок, а ошибка может вызвать другие отказы в других условиях.

Внутренняя мера качества программного средства (internal measure of software quality) – мера удовлетворения набором статических атрибутов программного продукта заданных и подразумеваемых потребностей в данном продукте при его использовании в заданных условиях. Статические атрибуты – это атрибуты, связанные с архитектурой, структурой ПС и его компонентами.

Например, меры сложности, а также количество, серьезность и частота найденных при проверке ошибок являются внутренними мерами качества, применяемыми по отношению к самому программному продукту.

Качество в использовании (quality in use) – степень применимости продукта или системы заданными пользователями для удовлетворения их потребностей в достижении заданных целей с результативностью, эффективностью, свободой от риска и удовлетворенностью в заданных контекстах использования.

Качество программного средства (software quality) – степень удовлетворения программным продуктом заданных и подразумеваемых потребностей при его использовании в заданных условиях.

Контекст использования (context in use) – пользователи, задачи, среда (аппаратная, программная, материалы), а также физическое и социальное окружение, в которых продукт используется.

Мера (measure) – переменная, которой присваивается значение в результате измерений. Термин «мера» используется собирательно для указания базовых мер, производных мер и указателей.

Библиотека БГУИР

Мера качества (quality measure) – производная мера, определяемая как результат функции измерения двух или более величин элементов меры качества (ЭМК).

Метод измерения (measurement method) – логическая последовательность действий, описанная с общих позиций и используемая при измерении.

Модель качества (quality model) – определенный набор характеристик и связей между ними, который обеспечивает схему для задания требований к качеству и оценки качества.

Подразумеваемые потребности (implied needs) – потребности, которые не могут быть заданы, но являются реально существующими потребностями. Подразумеваемые потребности включают потребности не заданные, но подразумеваемые другими заданными потребностями, и те незаданные потребности, которые рассматриваются как очевидные. Некоторые потребности становятся очевидными при использовании продукта в конкретных условиях.

Подхарактеристика качества программного средства (software quality subcharacteristic) – характеристика качества программного средства, входящая в состав другой характеристики качества.

Правообладатель (stakeholder) – лицо или организация, имеющие права, долю, претензии или интерес к системе или ее характеристикам, обеспечивающим их потребности и ожидания.

Программный продукт (software product) – набор компьютерных программ, процедур, возможно связанных с ними документации и данных. В стандартах серии *SQuaRE* качество программных средств (программного обеспечения) имеет то же значение, что и качество программного продукта.

Производная мера (derived measure) – мера, определяемая как функция двух или более значений базовых мер. Базовая мера, преобразованная с помощью математической функции, также может рассматриваться как производная мера.

Процедура измерения (measurement procedure) – логическая организация действий, применяемая специфически, используемая при выполнении конкретных измерений в соответствии с данным методом измерений. Процедура измерений обычно оформляется в виде документа, в котором она должна быть описана достаточно подробно, чтобы позволить выполнять измерение без дополнительной информации.

Свойство для количественной оценки (property to quantify) – свойство целевого объекта, связанное с элементом меры качества, которое может быть оценено количественно с помощью метода измерений. В стандартах серии *SQuaRE* понятие свойства используется аналогично понятию атрибута.

Свойство качества (quality property) – измеримый компонент качества.

Система (system) – совокупность взаимодействующих элементов, организованная для достижения одной или более заданных целей.

Функция измерения (measurement function) – алгоритм или вычисления, выполняемые для объединения двух или более элементов мер качества.

Библиотека БГУИР

Характеристика качества программного средства (software quality characteristic) – категория свойств программного средства, с помощью которой выражается его качество. Характеристики качества программных средств могут быть определены с помощью подхарактеристик и, в конечном итоге, атрибутов качества ПС.

Целевой объект (target entity) – фундаментальный объект интереса для пользователя, о котором собирается информация и который должен быть измерен. Целевым объектом могут быть разрабатываемые рабочие продукты, а также поведение системы, программного средства или правообладателей, таких как пользователи, операторы, разработчики, тестировщики или персонал сопровождения.

Элемент меры качества (quality measure element) – мера, определяемая в терминах свойства и метода измерения для ее количественной оценки, возможно включая преобразование с помощью математической функции. Элемент меры качества может являться как базовой, так и производной мерой.

1.2. Общие сведения о стандартах в области оценки качества, действующих на территории Республики Беларусь

В настоящее время в области оценки качества ПС на территории Республики Беларусь действуют следующие основные стандарты:

– межгосударственный стандарт стран СНГ ***ГОСТ 28806–90. Качество программных средств. Термины и определения*** [3];

– межгосударственный стандарт стран СНГ ***ГОСТ 28195–99. Оценка качества программных средств. Общие положения*** [2];

– национальный стандарт Беларуси ***СТБ ИСО/МЭК 9126-2003. Информационные технологии. Оценка программной продукции. Характеристики качества и руководства по их применению*** [4];

– национальный стандарт Беларуси ***СТБ ISO/IEC 25000-2009. Разработка программного обеспечения. Требования к качеству и оценка программного продукта (SQuaRE). Руководство по SQuaRE*** [5];

– национальный стандарт Беларуси ***СТБ ISO/IEC 25001-2009. Разработка программного обеспечения. Требования к качеству и оценка программного продукта (SQuaRE). Планирование и управление*** [6].

СТБ ИСО/МЭК 9126-2003 представляет собой аутентичный перевод международного стандарта *ISO/IEC 9126:1991* [20].

В *ГОСТ 28806–90* приведены основные термины и определения, принятые в области обеспечения качества ПС, определена модель качества ПС.

В ГОСТ 28195–99 представлены модель качества ПС, метод оценки качества ПС, классификация методов измерений свойств ПС.

СТБ ISO/IEC 25000-2009 и *СТБ ISO/IEC 25001-2009* представляют собой аутентичные переводы международных стандартов *ISO/IEC 25000:2005* и *ISO/IEC 25001:2007*. Данные стандарты относятся к серии стандартов *SQuaRE*. Информация о данной серии приведена в подразд. 1.6.

Библиотека БГУИР

1.3. Иерархическая модель оценки качества программных средств, регламентированная в СТБ ИСО/МЭК 9126-2003

ГОСТ 28806–90, ГОСТ 28195–99 и *СТБ ИСО/МЭК 9126-2003* [2–4] регламентируют выполнение оценки качества ПС и систем на основе ***иерархической модели качества***. В соответствии с данной моделью совокупность свойств, отражающих качество программного средства, представляется в виде многоуровневой структуры.

ГОСТ 28806–90 и *СТБ ИСО/МЭК 9126-2003* определяют первые *два уровня* иерархической модели качества. На первом (верхнем) уровне модели находятся характеристики. Характеристики соответствуют основным свойствам ПС. Характеристики оцениваются посредством подхарактеристик, находящихся на втором уровне модели. При этом номенклатура характеристик является *обязательной*, а номенклатура подхарактеристик – *рекомендуемой*.

ГОСТ 28195–99 определяет *четырехуровневую* иерархическую модель оценки качества ПС. На первом уровне модели находятся факторы качества (соответствуют характеристикам качества), на втором уровне – критерии качества (соответствуют подхарактеристикам качества), на третьем уровне – метрики, на четвертом – оценочные элементы. Номенклатура факторов и критериев является *обязательной*, а номенклатура метрик и оценочных элементов – *рекомендуемой.*

В *ГОСТ 28806–90* и *СТБ ИСО/МЭК 9126-2003* определены ***шесть характеристик качества*** ПС:

1. ***Функциональность (Functionality)*** – совокупность свойств ПС, определяемая наличием и конкретными особенностями набора функций, способных удовлетворять заданные или подразумеваемые потребности.

2. ***Надежность (Reliability)*** – совокупность свойств, характеризующая способность ПС сохранять заданный уровень пригодности в заданных условиях в течение заданного интервала времени.

3. ***Удобство использования (практичность, Usability)*** – совокупность свойств программного средства, характеризующая усилия, необходимые для его использования, и индивидуальную оценку результатов его использования заданным или подразумеваемым кругом пользователей.

4. ***Эффективность (Efficiency)*** – совокупность свойств программного средства, характеризующая те аспекты его уровня пригодности, которые связаны с характером и временем использования ресурсов, необходимых при заданных условиях функционирования.

5. ***Сопровождаемость (Maintainability)*** – совокупность свойств программного средства, характеризующая усилия, которые необходимы для его модификации.

6. ***Мобильность (Portability)*** – совокупность свойств программного средства, характеризующая приспособленность для переноса из одной среды функционирования в другие.

Библиотека БГУИР

На рис. 1.1 приведены два верхних уровня иерархической модели качества, определенной в *ГОСТ 28806–90* и *СТБ ИСО/МЭК 9126-2003.*

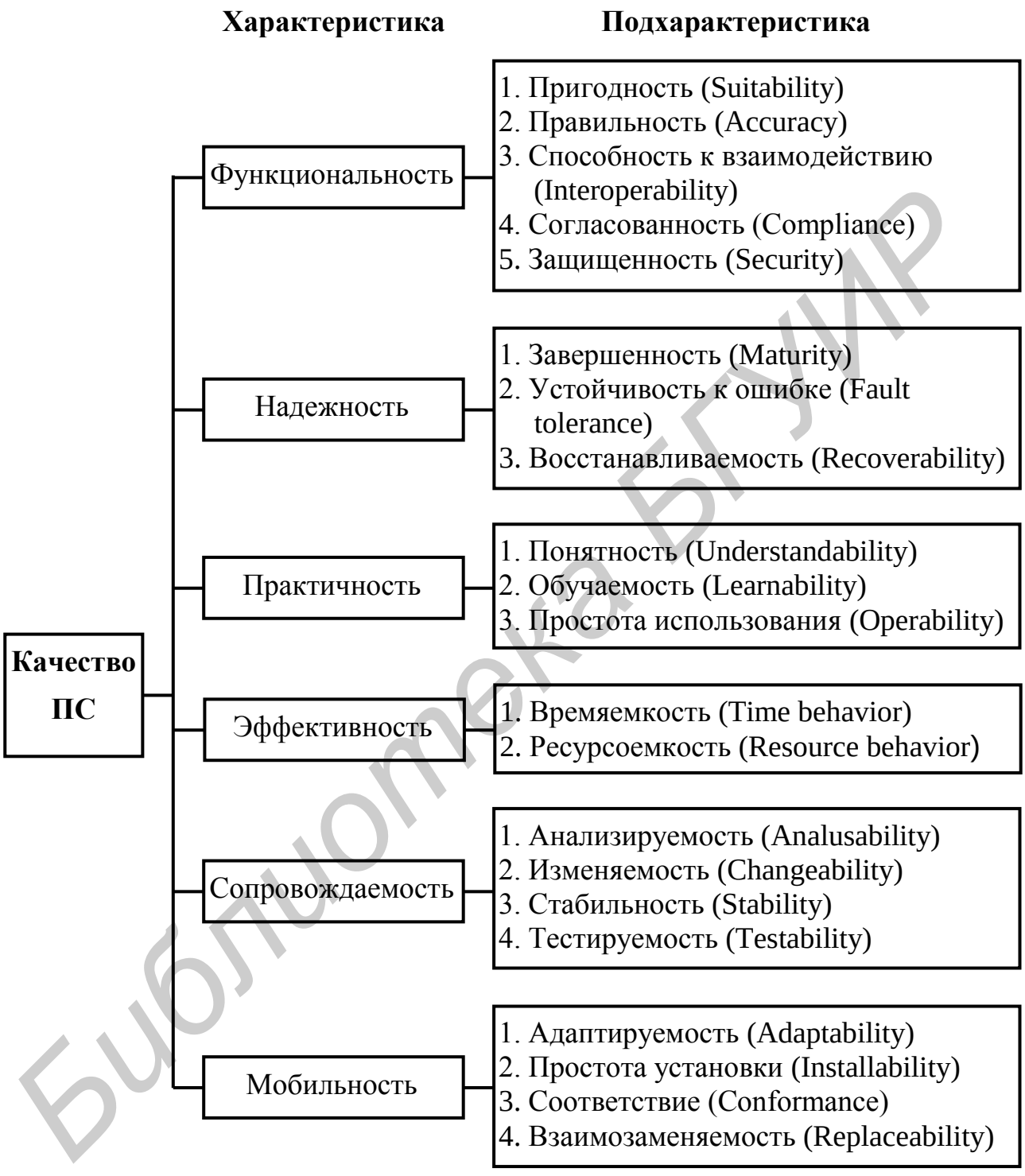


Рис. 1.1. Модель качества по СТБ ИСО/МЭК 9126-2003 (ISO/IEC 9126:1991) и ГОСТ 28806–90

Библиотека БГУИР

1.4. Развитие стандартизации качества программных средств за рубежом

В течение десяти лет (1991–2001 гг.) основой регламентирования характеристик качества ПС за рубежом являлся международный стандарт ***ISO/IEC 9126:1991 – Информационная технология – Оценка программного продукта – Характеристики качества и руководства по их применению*** [20]. В подразд. 1.3 описана модель качества ПС, приведенная в стандарте *СТБ ИСО/МЭК 9126-2003* [4], который является аутентичным переводом вышеназванного стандарта.

В 2001–2004 гг. стандарт *ISO/IEC 9126:1991* был заменен на две взаимосвязанные серии стандартов: *ISO/IEC 9126–1–4:2001–2004* и *ISO/IEC 14598–1–6:1998–2001* [21–24, 7–12].

Как и в *ISO/IEC 9126:1991*, в серии стандартов ***ISO/IEC 9126–1–4:2001–2004*** регламентирована *иерархическая модель качества программных средств*. На верхнем уровне модели находятся *характеристики*. Характеристики разделяются на *подхарактеристики*. Подхарактеристики определяются *метриками*. Метрики измеряют атрибуты (свойства) ПС.

Данная серия стандартов состоит из четырех частей под общим названием ***Программная инженерия – Качество продукта*** [21–24]:

– *ISO/IEC 9126–1:2001* – Часть 1: Модель качества;

– *ISO/IEC TR 9126–2:2003* – Часть 2: Внешние метрики;

– *ISO/IEC TR 9126–3:2003* – Часть 3: Внутренние метрики;

– *ISO/IEC TR 9126–4:2004* – Часть 4: Метрики качества в использовании.

Первая часть ***ISO/IEC 9126–1:2001*** является пересмотренной редакцией *ISO/IEC 9126:1991*. В данной части определены два верхних уровня (характеристики и подхарактеристики) иерархической модели качества, приведены общие требования к метрикам качества. В отличие от *ISO/IEC 9126:1991* подхарактеристики второго уровня стали нормативными, а не рекомендуемыми, определены две части модели качества (модель внутреннего и внешнего качества и модель качества в использовании) и исключен процесс оценки качества (он определен в стандарте *ISO/IEC 14598–1:1999*).

Модель внутреннего и внешнего качества близка к модели качества, определенной в *ISO/IEC 9126:1991*. В ней сохранена та же номенклатура из шести базовых характеристик качества ПС (см. рис. 1.1), к подхарактеристикам качества добавлено несколько новых подхарактеристик. Модель внутреннего и внешнего качества из стандарта *ISO/IEC 9126–1:2001* приведена на рис. 1.2. Курсивом на данном рисунке выделены новые подхарактеристики.

Модель качества в использовании, определенная в *ISO/IEC 9126–1:2001*, представляет собой двухуровневую модель, на верхнем уровне которой находятся характеристики, на втором – метрики. В соответствии с данной моделью качество в использовании разделяется на *четыре характеристики*: Результативность, Продуктивность, Безопасность, Удовлетворенность.

Библиотека БГУИР

Рис. 1.2. Модель внешнего и внутреннего качества по ISO/IEC 9126–1:2001

Библиотека БГУИР

В 2011 г. *ISO/IEC 9126–1*:*2001* заменен на стандарт *ISO/IEC 25010*:*2011* серии *SQuaRE* (см. п. 1.6.3 – 1.6.5).

Вторая – четвертая части *ISO/IEC TR 9126–2–4*:*2003–2004* опубликованы в виде технических отчетов (*TR*). Совокупности метрик, перечисленные в данных частях, являются рекомендуемыми, их набор не является исчерпывающим. Метрики могут модифицироваться. Возможно применение метрик, не включенных в данные части. В этих частях стандарта содержатся примеры метрик для каждой подхарактеристики, примеры применения метрик на протяжении жизненного цикла ПС.

Части *ISO/IEC TR 9126–2–4*:*2003–2004* в настоящее время являются действующими. На их основе организациями *ISO* и *IEC* разрабатываются стандарты *ISO/IEC 25022* и *ISO/IEC 25023* серии *SQuaRE* (см. п. 1.6.6).

Во второй части ***ISO/IEC TR 9126–2:2003*** определены метрики внешнего качества ПС. ***Внешние метрики*** – метрики, предназначенные для измерения качества программного продукта путем измерения поведения системы, частью которой является данный продукт. Внешние метрики могут использоваться в процессе эксплуатации и на стадиях тестирования или испытаний в процессах разработки и сопровождения ПС, когда уже созданы исполнимые коды программного продукта.

В третьей части ***ISO/IEC TR 9126–3:2003*** определяются метрики внутреннего качества ПС. ***Внутренние метрики*** – метрики, измеряющие собственные свойства ПС. Они измеряются в процессе разработки ПС на основе спецификации требований, результатов проектирования, исходного кода или другой документации ПС. Внутренние метрики дают возможность оценить качество промежуточных программных продуктов разработки, предсказывая качество конечного ПС.

В четвертой части ***ISO/IEC TR 9126–4:2004*** определяются метрики качества в использовании. ***Метрики качества в использовании*** – метрики, измеряющие соответствие продукта потребностям заданных пользователей в достижении заданных целей с результативностью, продуктивностью, безопасностью и удовлетворением в заданных контекстах использования. Данные метрики могут использоваться только в процессе эксплуатации ПС в реальной среде окружения. Метрики качества в использовании основаны на измерении поведения типичных пользователей и системы, содержащей данное ПС.

В серии стандартов ***ISO/IEC 14598–1–6:1998–2001*** определены процессы оценки качества программного продукта, содержатся руководство и требования к оценке. Данная серия предназначена для применения при разработке, приобретении и независимой оценке программного средства. Серия состоит из шести частей [7–12]:

– *ISO/IEC 14598–1*:*1999* – Информационная технология – Оценка программного продукта – Часть 1: Общий обзор;

– *ISO/IEC 14598–2*:*2000* – Программная инженерия – Оценка продукта – Часть 2: Планирование и управление;

Библиотека БГУИР

– *ISO/IEC 14598–3*:*2000* – Программная инженерия – Оценка продукта – Часть 3: Процесс для разработчиков;

– *ISO/IEC 14598–4*:*1999* – Программная инженерия – Оценка продукта – Часть 4: Процесс для заказчиков;

– *ISO/IEC 14598–5*:*1998* – Информационная технология – Оценка программного продукта – Часть 5: Процесс для оценщиков;

– *ISO/IEC 14598–6*:*2001* – Программная инженерия – Оценка продукта – Часть 6: Документация модулей оценки.

В первой части ***ISO/IEC 14598–1*:*1999*** приведен обзор остальных частей, определена связь серии *ISO/IEC 14598* с серией *ISO/IEC 9126* и стандартом *ISO/IEC 12207*:*1995*. В данной части представлены общие требования к спецификации и оценке качества, разъяснены концепции оценки. Установлены требования к методам измерений и оценки программных продуктов. Определен общий процесс оценки качества программного продукта. Основой для данного процесса оценки явился процесс оценки качества ПС из *ISO/IEC 9126*:*1991*.

В 2011 г. *ISO/IEC 14598–1*:*1999* заменен на стандарт *ISO/IEC 25040*:*2011* серии *SQuaRE*.

Во *второй части* ***ISO/IEC 14598–2*:*2000*** приводятся концепции планирования и управления процессом оценки качества программного продукта, рассматривается содержание плана количественной оценки качества.

Третья – пятая части ***ISO/IEC 14598–3–5*:*1998–2000*** предназначены для организаций-разработчиков, организаций-заказчиков и оценщиков ПС соответственно. В данных частях рассмотрены особенности процесса оценки качества ПС при его выполнении вышеприведенными сторонами.

Шестая часть стандарта ***ISO/IEC 14598–6*:*2001*** содержит руководство по документированию модулей оценки. ***Модуль оценки*** представляет собой полностью укомплектованную информацию, необходимую для проведения процесса оценки некоторой характеристики или подхарактеристики качества. Модуль содержит спецификацию соответствующей модели качества (характеристика, подхарактеристики, метрики качества), процедуры оценки, входные данные для оценки, структуру типового отчета о результатах выполненной оценки. Рассмотрены примеры модулей оценки.

1.5. Метрики качества программных средств

1.5.1. Свойства и критерии обоснованности метрик

Метрики качества ПС классифицируются на внутренние, внешние и метрики качества в использовании (см. подразд. 1.5). Применение конкретного вида метрик определяется стадией жизненного цикла программного средства.

Вторая – четвертая части стандарта *ISO/IEC TR 9126–2–4*:*2003–2004* [22–24] посвящены детальному рассмотрению соответственно внешних и внутренних метрик качества программных средств и метрик качества в использовании.

В данных частях стандарта определены следующие ***желательные свойства метрик***:

Библиотека БГУИР

1) *надежность*; свойство метрики связано со случайной ошибкой; метрика свободна от случайной ошибки, если случайные изменения не влияют на результаты метрики;

2) *повторяемость*; повторное использование метрики для того же продукта теми же специалистами по оценке, используя ту же спецификацию оценки (включая ту же окружающую среду) и тот же тип пользователей, должно привести к тем же результатам с соответствующими допусками;

3) *воспроизводимость*; применение метрики для того же продукта различными специалистами по оценке, используя ту же спецификацию оценки (включая ту же окружающую среду) и тот же тип пользователей, должно привести к тем же результатам с соответствующими допусками;

4) *доступность*; метрика должна четко указывать условия (например, наличие определенных атрибутов), которые ограничивают ее употребление;

5) *показательность*; данное свойство определяет способность метрики идентифицировать части или элементы программы, которые должны быть улучшены, на основании сравнения измеренных и ожидаемых результатов;

6) *корректность*; метрика должна обладать следующими свойствами:

– *объективность*; результаты метрики и ее входные данные должны быть основаны на фактах и не подвластны чувствам или мнениям специалистов по оценке или тестированию (исключая метрики удовлетворенности или привлекательности, измеряющие чувства и мнения пользователя);

– *беспристрастность*; измерение не должно быть направлено на получение какого-либо специфического результата;

– *адекватность точности*; точность определяется при проектировании метрики и при выборе описаний фактов, используемых как основа для метрики; разработчик метрики должен описать точность и чувствительность метрики;

7) *значимость*; измерение должно давать значимые результаты, касающиеся поведения программы или характеристик качества.

Метрика должна также быть *эффективной по отношению к стоимости*. Это значит, что более дорогие метрики должны обеспечивать лучшие результаты оценки.

Разработчик метрики должен доказать ее обоснованность. Метрика должна удовлетворять хотя бы одному из следующих ***критериев обоснованности метрики***:

1) *корреляция*; изменение в значениях характеристик качества (оперативно определенных по результатам измерения основных метрик), обусловленное изменением в значениях метрики, должно определяться линейной зависимостью;

2) *трассировка*; если метрика M непосредственно связана с величиной характеристики качества Q (оперативно определенной по результатам измерения основных метрик), то изменение величины $Q(T1)$, имеющейся в момент времени $T1$, к величине $Q(T2)$, полученной в момент времени $T2$, должно сопровождаться изменением значения метрики от $M(T1)$ до $M(T2)$ в том же направлении (например, если увеличивается Q, то M тоже увеличивается);

Библиотека БГУИР

3) *непротиворечивость*; если значения характеристик качества (оперативно полученные по результатам измерения основных метрик) ***Q*1**, ***Q*2**,…, ***Qn***, связанные с продуктами или процессами 1, 2**,...,** ***n***, определяются соотношением ***Q*1** > ***Q*2** > ... > ***Qn***, то соответствующие значения метрики должны удовлетворять соотношению ***M*1** > ***M*2** > ... > ***Mn***;

4) *предсказуемость*; если метрика используется в момент времени ***T*1** для прогноза значения (оперативно полученного по результатам измерения основных метрик) характеристики качества ***Q*** в момент времени ***T*2**, то ошибка прогнозирования, определяемая выражением

(прогнозное *Q* (*T*2) – фактическое *Q* (*T*2)) / (фактическое *Q*(*T*2)),

должна попадать в допустимый диапазон ошибок прогнозирования;

5) *селективность*; метрика должна быть способной различать высокое и низкое качество программного средства.

В стандартах *ISO/IEC 9126–2–4:2003–2004* для каждой подхарактеристики внешнего и внутреннего качества и характеристики качества в использовании приведены таблицы, в которых даны примеры метрик качества.

Таблицы имеют следующую структуру:

1) название метрики;
2) назначение метрики;
3) метод применения;
4) способ измерения, формула, исходные и вычисляемые данные;
5) интерпретация измеренного значения (диапазон и предпочтительные значения);
6) тип шкалы, используемой при измерении метрики (номинальная, порядковая, интервальная, шкала отношений или абсолютная);
7) тип измеренного значения; используются следующие ***типы измеренных значений***: *тип размера* (например, функциональный размер, размер исходного текста); *тип времени* (например, затраченное время, необходимое пользователю время); *тип количества* (например, количество изменений, количество отказов);
8) источники входных данных для измерения;
9) ссылка на *ISO/IEC 12207:1995* (процессы жизненного цикла программных средств, при выполнении которых применима метрика);
10) целевая аудитория.

Для обеспечения возможности совместного использования различных метрик (независимо от их физического смысла, единиц измерения и диапазонов значений) при интегральной оценке качества программных продуктов метрики *по возможности* должны быть представлены в относительных единицах в виде

$$X = A / B \qquad (1.1)$$

или

$$X = 1 - A / B, \qquad (1.2)$$

где X – значение метрики; A и B – значения некоторых свойств оцениваемого продукта или документации.

Библиотека БГУИР

В пп. 1.5.2 – 1.5.4 приведены конкретные примеры свойств ***A*** и ***B***. В стандартах серии *SQuaRE* элементы ***A*** и ***B***, участвующие в вычислении метрики, называются элементами меры качества (см. п. 1.6.6)

Из формул (1.1) и (1.2) для конкретной метрики выбирается та, которая соответствует *критериям трассировки и непротиворечивости*: с увеличением относительного значения метрики значение подхарактеристики и характеристики качества должно увеличиваться.

Вычисление метрик по формуле (1.1) или (1.2) при использовании для ***A*** и ***B*** одинаковых единиц измерения позволяет привести их относительные значения в диапазон

$$\mathbf{0} \le \boldsymbol{X} \le \mathbf{1}, \qquad (1.3)$$

что упрощает их совместное использование при интегральной оценке качества программных средств.

В пп. 1.5.2 – 1.5.4 приведены примеры метрик (по одной на каждую подхарактеристику или характеристику качества) из рекомендуемых в стандартах *ISO/IEC TR 9126–2–4:2003–2004* [22–24].

Следует отметить, что не все метрики, приведенные в данных стандартах, удовлетворяют вышеприведенным свойствам, критериям, оцениваются с помощью выражений (1.1), (1.2) или попадают в диапазон (1.3).

1.5.2. Внутренние метрики качества программных средств

Примеры внутренних метрик качества ПС, содержащихся в стандарте *ISO/IEC TR 9126–3*:*2003* [23], содержит табл. 1.1. Во втором столбце таблицы по каждой подхарактеристике приведено название одной метрики, уникальная формула или номер формулы (1.1) или (1.2) из п. 1.5.1 для оценки данной метрики. Исходные данные в третьем столбце – это данные, используемые в соответствующей формуле для вычисления значения представленной метрики.

Таблица 1.1

Внутренние метрики качества программных средств

Название подхарактеристики	Название метрики, формула для ее оценки	Исходные данные для вычисления метрики по соответствующей формуле
1	2	3
Функциональность		
Пригодность	Полнота функциональной реализации (1.2)	***A*** – число нереализованных функций, обнаруженных при оценке; ***B*** – число функций, описанных в спецификации требований

Библиотека БГУИР

Продолжение табл. 1.1

1	2	3
Правильность	Точность (1.1)	***A*** – количество элементов данных, реализованных с заданными уровнями точности, подтвержденное при оценке; ***B*** – количество элементов данных, для которых в спецификации заданы уровни точности
Способность к взаимодействию	Соответствие интерфейсов (протоколов) (1.1)	***A*** – количество интерфейсных протоколов, реализующих заданный в спецификации формат, подтвержденных при проверке; ***B*** – количество интерфейсных протоколов, которые должны быть реализованы в соответствии со спецификацией
Защищенность	Предотвращение разрушения данных (1.1)	***A*** – количество реализованных случаев предотвращения разрушения данных из заданных в спецификации, подтвержденное при проверке; ***B*** – количество случаев доступа, которые определены в спецификации как способные разрушить данные
Соответствие функциональности	Соответствие функциональности (1.1)	***A*** – количество корректно реализованных элементов, связанных с соответствием функциональности, подтвержденное при оценке; ***B*** – общее количество элементов соответствия
Надежность		
Завершенность	Полнота тестирования (1.1)	***A*** – количество тестовых комбинаций, спроектированных в плане тестирования и подтвержденных при проверке; ***B*** – количество требуемых тестовых комбинаций
Устойчивость к ошибке	Предотвращение некорректных действий (1.1)	***A*** – количество функций, реализованных с предотвращением некорректных действий; ***B*** – количество типичных некорректных действий, которое должно быть учтено
Восстанавливаемость	Способность к восстановлению (1.1)	***A*** – количество реализованных требований к восстановлению, подтвержденное при проверке; ***B*** – общее количество требований к восстановлению, определенных в спецификации
Соответствие надежности	Соответствие надежности (1.1)	***A*** – количество корректно реализованных элементов, связанных с соответствием надежности, подтвержденное при оценке; ***B*** – общее количество элементов соответствия

Библиотека БГУИР

Продолжение табл. 1.1

1	2	3
Практичность		
Понятность	Способность к демонстрации (1.1)	***A*** – количество демонстрируемых функций, подтвержденное при проверке; ***B*** – общее количество функций, которые должны обладать способностью к демонстрации
Обучаемость	Полнота документации пользователя и/или возможности электронной справки help (1.1)	***A*** – количество описанных функций; ***B*** – общее количество предоставляемых функций
Простота использования	Отменяемость действий пользователя (1.1)	***A*** – количество реализованных функций, которые могут быть отменены пользователем с восстановлением предыдущих данных; ***B*** – общее количество функций
Привлекательность	Настраиваемость вида интерфейса пользователя (1.1)	***A*** – количество типов элементов интерфейса, которые могут быть настроены; ***B*** – общее количество типов элементов интерфейса
Соответствие практичности	Соответствие практичности (1.1)	***A*** – количество корректно реализованных элементов, связанных с соответствием практичности, подтвержденное при оценке; ***B*** – общее количество элементов соответствия
Эффективность		
Поведение во времени	Пропускная способность $X = A$	***A*** – число задач в единицу времени, подтвержденное при проверке
Использование ресурсов	Использование памяти $X = A$	***A*** – размер памяти в байтах (вычисленный или смоделированный)
Соответствие эффективности	Соответствие эффективности (1.1)	***A*** – количество корректно реализованных элементов, связанных с соответствием эффективности, подтвержденное при проверке; ***B*** – общее количество элементов соответствия
Сопровождаемость		
Анализируемость	Готовность диагностических функций (1.1)	***A*** – количество реализованных диагностических функций из заданных в спецификации, подтвержденное при проверке; ***B*** – требуемое количество диагностических функций

Библиотека БГУИР

Продолжение табл. 1.1

1	2	3
Изменяемость	Регистрируемость изменений (1.1)	***A*** – количество изменений в функциях/модулях, отраженных в комментариях, подтвержденное при проверке; ***B*** – общее количество изменений в функциях/модулях относительно оригинального кода
Стабильность	Влияние изменений (1.2)	***A*** – количество обнаруженных вредных влияний после модификаций; ***B*** – количество сделанных модификаций
Тестируемость	Полнота встроенных функций тестирования (1.1)	***A*** – количество реализованных встроенных функций тестирования из заданных в спецификации, подтвержденное при проверке; ***B*** – требуемое количество встроенных функций тестирования
Соответствие сопровождаемости	Соответствие сопровождаемости (1.1)	***A*** – количество корректно реализованных элементов, связанных с соответствием сопровожаемости, подтвержденное при оценке; ***B*** – общее количество элементов соответствия
Мобильность		
Адаптируемость	Адаптируемость структур данных (1.1)	***A*** – количество структур данных, работоспособность которых не нарушена после адаптации, подтвержденное при проверке; ***B*** – общее количество структур данных, которые должны обладать способностью к адаптации
Настраиваемость	Объем работ по установке (1.1)	***A*** – количество автоматических шагов инсталляции, подтвержденное при проверке; ***B*** – требуемое количество шагов инсталляции
Совместимость	Доступная совместимость (1.1)	***A*** – количество объектов, с которыми продукт может сосуществовать, из заданных в спецификации; ***B*** – количество объектов в окружающей среде, с которыми продукт должен сосуществовать
Взаимозаменяемость	Преемственность данных (1.1)	***A*** – количество элементов данных ПС, которые продолжают использоваться после замещения (из заданных в спецификации), подтвержденное при проверке; ***B*** – количество элементов старых данных, которые должны использоваться из старого ПС

Библиотека БГУИР

Окончание табл. 1.1

1	2	3
Соответствие мобильности	Соответствие мобильности (1.1)	***A*** – количество корректно реализованных элементов, связанных с соответствием мобильности, подтвержденное при проверке; ***B*** – общее количество элементов соответствия

1.5.3. Внешние метрики качества программных средств

Примеры внешних метрик качества программных средств из стандарта *ISO/IEC TR 9126–2*:*2003* [22] содержит табл. 1.2. Во втором столбце таблицы по каждой подхарактеристике приведено название одной метрики, уникальная формула или номер формулы (1.1) или (1.2) из п. 1.5.1 для оценки данной метрики. Исходные данные в третьем столбце – это данные, используемые в соответствующей формуле для вычисления значения представленной метрики.

Таблица 1.2

Внешние метрики качества программных средств

Название подхарактеристики	Название метрики, формула для ее оценки	Исходные данные для вычисления метрики по соответствующей формуле
1	2	3
Функциональность		
Пригодность	Полнота функциональной реализации (1.2)	***A*** – количество отсутствующих функций, обнаруженное при проверке; ***B*** – количество функций, описанных в спецификации
Правильность	Точность ***X = A / T***	***A*** – количество результатов, подсчитанное пользователями, с уровнем точности, отличающимся от требуемого; ***T*** – продолжительность работы
Способность к взаимодействию	Способность к обмену данными (основанная на успешных попытках пользователя) (1.2)	***A*** – количество случаев, в которых пользователю не удалось обменяться данными с другими ПС или системами; ***B*** – количество случаев, в которых пользователь пытался обмениваться данными
Защищенность	Предотвращение разрушения данных (1.2)	***A*** – количество произошедших случаев разрушения важных данных; ***B*** – количество тестовых случаев, направленных на разрушение данных

Библиотека БГУИР

Продолжение табл. 1.2

1	2	3
Соответствие функциональности	Соответствие функциональности (1.2)	***A*** – количество заданных элементов соответствия функциональности, не подтвержденных при тестировании; ***B*** – общее количество заданных элементов соответствия функциональности
Надежность		
Завершенность	Плотность ошибок ***X = A / Z***	***A*** – количество ошибок, обнаруженных в течение определенного испытательного срока; ***Z*** – размер продукта
Устойчивость к ошибке	Предотвращение некорректных действий (1.1)	***A*** – количество предотвращенных критических и серьезных отказов; ***B*** – количество выполненных при тестировании тестовых случаев, направленных на проверку типовых некорректных действий, которые могут привести к отказу
Восстанавливаемость	Способность к восстановлению (1.1)	***A*** – количество случаев успешного восстановления; ***B*** – количество случаев восстановления, протестированных согласно требованиям
Соответствие надежности	Соответствие надежности (1.2)	***A*** – количество заданных элементов соответствия надежности, не подтвержденных при тестировании; ***B*** – общее количество заданных элементов соответствия надежности
Практичность		
Понятность	Полнота описания (1.1)	***A*** – количество функций (или классов функций), понятных после прочтения документации на программный продукт; ***B*** – общее количество функций (или классов функций), реализуемых программным продуктом
Обучаемость	Эффективность документации пользователя и/или справочной системы (help) (1.1)	***A*** – количество задач, успешно выполненных после получения оперативной справки и/или чтения документации; ***B*** – общее количество протестированных задач

Библиотека БГУИР

Продолжение табл. 1.2

1	2	3
Простота использования	Исправление ошибок при использовании (1.1)	***A*** – число экранов или форм, где входные данные были успешно модифицированы или изменены (восстановлены) перед очередной обработкой; ***B*** – число экранов или форм, где пользователь пытался модифицировать или изменить (восстановить) входные данные в течение испытательного срока использования
Привлекательность	Изменяемость вида интерфейса (1.1)	***A*** – количество элементов интерфейса, измененных внешне для удовлетворения пользователя; ***B*** – количество элементов интерфейса, которые пользователь хотел изменить
Соответствие практичности	Соответствие практичности (1.2)	***A*** – количество заданных элементов соответствия практичности, не подтвержденных при тестировании; ***B*** – общее количество заданных элементов соответствия практичности
Эффективность		
Поведение во времени	Время отклика $X = A - B$	***A*** – момент времени получения результата; ***B*** – момент времени завершения ввода команды
Использование ресурсов	Использование устройств ввода/вывода (1.1)	***A*** – время занятости устройств ввода/вывода; ***B*** – заданное время, предназначенное для использования устройств ввода/вывода
Соответствие эффективности	Соответствие эффективности (1.2)	***A*** – количество заданных элементов соответствия эффективности, не подтвержденных при тестировании; ***B*** – общее количество заданных элементов соответствия эффективности
Сопровождаемость		
Анализируемость	Поддержка диагностическими функциями (1.1)	***A*** – количество отказов, при которых персонал сопровождения с помощью диагностических функций может диагностировать причину; ***B*** – общее число зарегистрированных отказов
Изменяемость	Возможность управления изменением ПС (1.1)	***A*** – количество фактически записанных данных регистрации изменений; ***B*** – запланированное количество данных регистрации изменений, достаточное для отслеживания изменений ПС
Стабильность	Возникновение отказа после изменения $X = A / N$	***A*** – количество отказов, возникших в течение заданного периода после устранения отказа; ***N*** – количество отказов, устраненных путем изменения ПС

Библиотека БГУИР

Окончание табл. 1.2

1	2	3
Тестируемость	Доступность встроенных функций тестирования (1.1)	***A*** – количество случаев, в которых персонал сопровождения может использовать встроенные функции тестирования; ***B*** – количество подходящих случаев, в которых можно было бы использовать встроенные тесты
Соответствие сопровождаемости	Соответствие сопровождаемости (1.2)	***A*** – количество заданных элементов соответствия сопровождаемости, не подтвержденных при тестировании; ***B*** – общее количество заданных элементов соответствия сопровождаемости
Мобильность		
Адаптируемость	Адаптируемость структур данных (1.1)	***A*** – количество работоспособных данных, которые не требуют сопровождения при адаптации; ***B*** – ожидаемое число работоспособных данных в окружающей среде, к которой ПС адаптировано
Настраиваемость	Простота установки (1.1)	***A*** – число успешных случаев приспосабливания пользователем операции инсталляции к среде эксплуатации; ***B*** – общее число попыток пользователя приспособить операцию инсталляции к среде окружения
Совместимость	Доступная совместимость $X = A / T$	***A*** – число любых ограничений или непредусмотренных отказов, с которыми пользователь сталкивается во время одновременной работы с другими ПС; ***T*** – продолжительность одновременной работы с другими ПС
Взаимозаменяемость	Преемственность данных (1.1)	***A*** – число данных замещаемого ПС, которые могут продолжать использоваться после его замещения; ***B*** – число данных замещаемого ПС, которые по плану должны продолжать использоваться после его замещения
Соответствие мобильности	Соответствие мобильности (1.2)	***A*** – количество заданных элементов соответствия мобильности, не подтвержденных при тестировании; ***B*** – общее количество заданных элементов соответствия мобильности

1.5.4. Метрики качества программных средств в использовании

Примеры метрик качества в использовании из стандарта *ISO/IEC TR 9126–4:2004* [24] содержит табл. 1.3. Во втором столбце таблицы по каждой подха-

Библиотека БГУИР

рактеристике приведено название одной метрики, номер формулы (1.1) или (1.2) из п. 1.5.1 для оценки данной метрики. Исходные данные в третьем столбце – данные, используемые в соответствующей формуле для вычисления значения представленной метрики.

Таблица 1.3

Метрики качества ПС в использовании

Название характеристики	Название метрики, формула для ее оценки	Исходные данные для вычисления метрики по соответствующей формуле
1	2	3
Результативность	Завершение задачи (1.1)	***A*** – количество завершенных задач; ***B*** – общее количество попыток выполнения задач
Продуктивность	Коэффициент продуктивности (1.1)	***A*** – продуктивное время; ***A = B – (B1 + B2 + + B3)***, где ***B*** – продолжительность выполнения задачи; ***B1*** – продолжительность помощи; ***B2*** – продолжительность обработки ошибок; ***B3*** – продолжительность поиска
Безопасность	Экономический ущерб (1.2)	***A*** – число случаев экономического ущерба; ***B*** – общее число случаев использования
Удовлетворенность	Использование по собственному усмотрению (1.1)	***A*** – количество случаев использования заданных в спецификации функций программного средства / приложений / систем; ***B*** – количество случаев их запланированного использования

1.6. Серия стандартов SQuaRE

1.6.1. Структура серии стандартов SQuaRE

В настоящее время организациями ISO и IEC активно ведутся работы по разработке серии стандартов под общим названием ***Системная и программная инженерия – Требования к качеству и оценка программного продукта*** (***Systems and software engineering – Software product Quality Requirements and Evaluation (SQuaRE)***). Данная серия призвана заменить собой серии стандартов *ISO/IEC 9126–1–4*:*2001–2004* и *ISO/IEC 14598–1–6*:*1998–2001* [21–24, 7–12].

Основными преимуществами серии стандартов *SQuaRE* являются:

– координация руководств по измерению и оценке качества программных продуктов;

– наличие руководства по спецификации требований к качеству программного продукта;

– гармонизация со стандартом *ISO/IEC 15939*:*2007* [13] в форме эталонной модели измерений качества.

Библиотека БГУИР

На рис. 1.3 приведена организация серии стандартов *SQuaRE* [5, 14].

Серия стандартов *SQuaRE* разделена на следующие группы (разделы):

– **ISO/IEC 2500n – группа управления качеством**. Стандарты из данной группы определяют общие модели, термины и определения, которые используются в остальных стандартах серии *SQuaRE*. Данная группа стандартов содержит также руководство по использованию стандартов серии *SQuaRE*;

– **ISO/IEC 2501n – группа модели качества**. В стандартах данной группы представлены подробные модели качества для компьютерных систем и программных продуктов, качества в использовании и качества данных. Данная группа стандартов содержит также практическое руководство по использованию представленных моделей качества;

– **ISO/IEC 2502n – группа измерения качества**. Стандарты данной группы включают эталонную модель измерений качества программного продукта, математические определения мер качества и практическое руководство по их применению. Даются примеры внутренних и внешних мер качества программных продуктов и систем, а также мер качества в использовании. Определены и представлены элементы мер качества, являющиеся основой этих мер;

– **ISO/IEC 2503n – группа требований к качеству**. Стандарты данной группы помогают определить требования к качеству, основываясь на моделях и мерах качества. Эти требования к качеству могут использоваться в процессе выявления требований к качеству разрабатываемого программного продукта или как входные данные для процесса оценки;

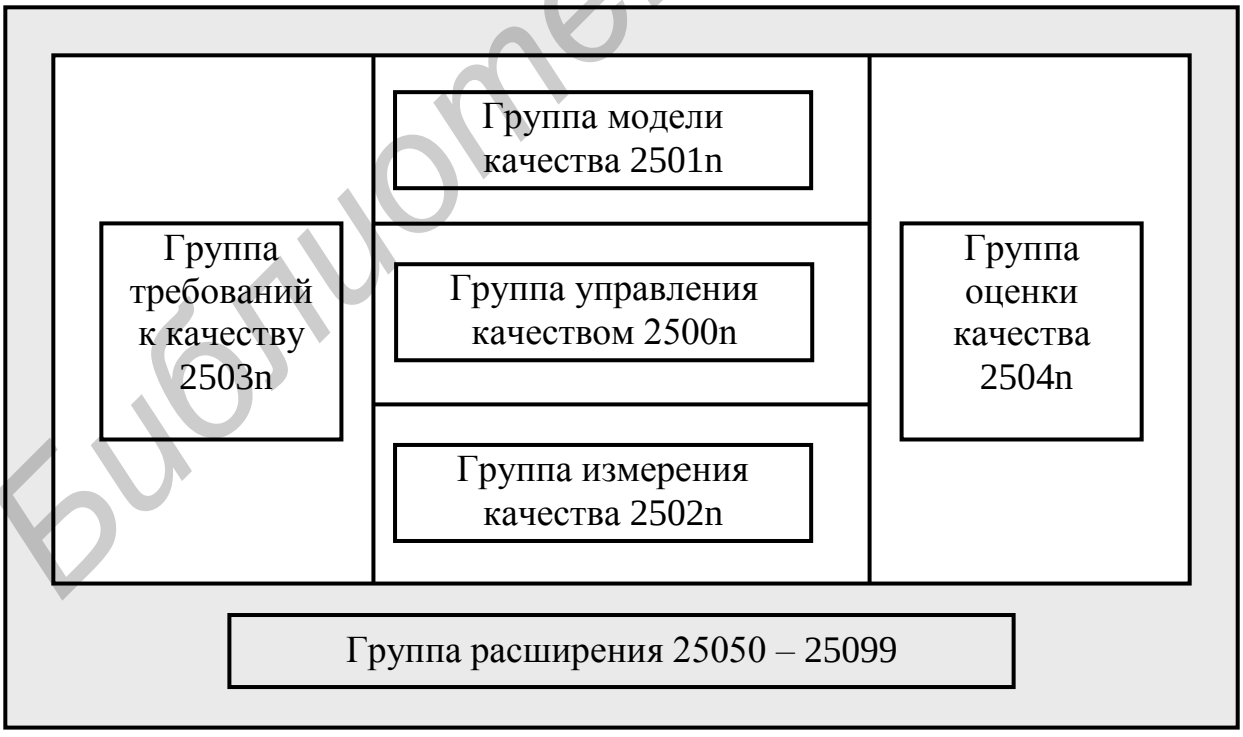


Рис. 1.3. Организация серии международных стандартов SQuaRE

Библиотека БГУИР

– **ISO/IEC 2504n – группа оценки качества**. Стандарты данной группы содержат требования, рекомендации и руководства по оценке программного продукта оценщиками, заказчиками или разработчиками. Здесь так же представлены правила документирования мер в виде модуля оценки;

– **группа расширения ISO/IEC 25050 – 25099**. Стандарты данной группы включают требования к качеству коммерческих готовых программных продуктов (Commercial Off-The-Shelf Software – COTS) и общие промышленные форматы для отчетов по практичности.

1.6.2. Группа стандартов ISO/IEC 2500n

В данную группу стандартов входят следующие стандарты:

– ***ISO/IEC 25000:2005. Программная инженерия – Требования к качеству и оценка программного продукта* (*SQuaRE*) – *Руководство по SQuaRE*** [14];

– ***ISO/IEC 25001:2007. Программная инженерия – Требования к качеству и оценка программного продукта* (*SQuaRE*) – *Планирование и управление*** [15].

Данные стандарты приняты в качестве национальных стандартов Республики Беларусь *СТБ ISO/IEC 25000-2009* и *СТБ ISO/IEC 25001-2009* [5, 6].

Стандарт *ISO/IEC 25000*:*2005* содержит основные термины и определения в области оценки качества ПС и систем, описание структуры серии стандартов *SQuaRE*, описание взаимосвязи стандартов серий *SQuaRE*, *ISO/IEC 9126* и *ISO/IEC 14598*.

В стандарте *ISO/IEC 25001*:*2007* представлены концепции управления оценкой, рекомендации по определению требований к качеству и оценке качества, приведен образец плана оценки качества.

1.6.3. Группа стандартов ISO/IEC 2501n

В данную группу стандартов входят следующие стандарты:

– ***ISO/IEC 25010:2011 – Системная и программная инженерия – Требования к качеству и оценка программного продукта* (*SQuaRE*) – *Модели качества систем и программных средств*** [16];

– ***ISO/IEC 25012:2008 – Программная инженерия – Требования к качеству и оценка программного продукта* (*SQuaRE*) – *Модель качества данных*** [17].

Стандарт *ISO/IEC 25010*:*2011* введен взамен стандарта *ISO/IEC 9126–1*:*2001* [21].

Качество системы – степень удовлетворения системой заданных и подразумеваемых потребностей своих правообладателей. Эти потребности представляются моделями качества, приведенными в стандарте *ISO/IEC 25010*:*2011*. Данные модели имеют иерархическую структуру. На верхнем уровне моделей находятся характеристики. Большинство из характеристик подразделяются на подхарактеристики.

Измеряемые свойства, связанные с качеством системы или продукта, называются *свойствами качества*. Свойства качества ассоциируются с мерами качества. Чтобы получить меры характеристики или подхарактеристики каче-

Библиотека БГУИР

ства без их непосредственного измерения, необходимо определить совокупность свойств, которые вместе покрывают характеристику или подхарактеристику, получить меры качества для каждого из свойств и объединить их с помощью некоторых вычислений для получения результирующей меры качества, связанной с характеристикой или подхарактеристикой качества.

В стандарте *ISO/IEC 25010*:*2011* определены *две модели качества*:

– модель качества в использовании, состоящая из пяти характеристик, имеющих отношение к результатам взаимодействий продукта при его применении в заданном контексте использования. Данная модель представляет собой модель системы, применимую к полным человеко-компьютерным системам, включая как используемые компьютерные системы, так и используемые программные продукты;

– модель качества продукта, состоящая из восьми характеристик, имеющих отношение к статическим свойствам программного средства и динамическим свойствам компьютерной системы. Данная модель применима как к компьютерным системам, так и к программным продуктам.

1.6.4. Модель качества в использовании по стандарту ISO/IEC 25010:2011

Качество в использовании – степень применимости продукта или системы заданными пользователями для удовлетворения их потребностей в достижении заданных целей с результативностью, эффективностью, свободой от риска и удовлетворенностью в заданных контекстах использования. Понятие *контекста использования* свойственно как для качества в использовании, так и для качества продукта. В последнем случае оно определяется как *заданные условия*.

Качество в использовании характеризует влияние, которое продукт (система или программный продукт) оказывает на правообладателей. Оно определяется качеством программного обеспечения, аппаратных средств и эксплуатационной среды, а также характеристиками пользователей, задач и социального окружения.

Модель качества в использовании приведена на рис. 1.4. Данная модель состоит из пяти характеристик, связанных с результатами взаимодействия с системой: результативности, эффективности, удовлетворенности, свободы от рисков, покрытия контекста. Курсивом на рисунке выделены отличия данной модели от предыдущей версии модели качества в использовании, определенной в *ISO/IEC 9126–1*:*2001* (см. подразд. 1.4).

Результативность (Effectiveness) – точность и полнота, с которой пользователи достигают заданных целей.

***Эффективность* (*Efficiency*)** – ресурсы, затрачиваемые в зависимости от точности и полноты, с которыми пользователь достигает целей.

Удовлетворенность (Satisfaction) – степень удовлетворения потребностей пользователя при применении продукта или системы в заданном контексте использования. Подхарактеристиками удовлетворенности являются применимость, доверие, удовольствие, комфорт.

Библиотека БГУИР

Свобода от риска (Freedom from risk) – степень уменьшения продуктом или системой потенциального риска для экономического статуса, человеческой жизни, здоровья или окружающей среды. Подхарактеристиками свободы от риска являются уменьшение экономического риска, уменьшение риска для здоровья и безопасности, уменьшение риска для окружающей среды.

Покрытие контекста (Context coverage) – степень применимости продукта или системы с результативностью, эффективностью, свободой от риска и удовлетворенностью как в заданных контекстах использования, так и вне их. Подхарактеристиками покрытия контекста являются полнота контекста, гибкость.

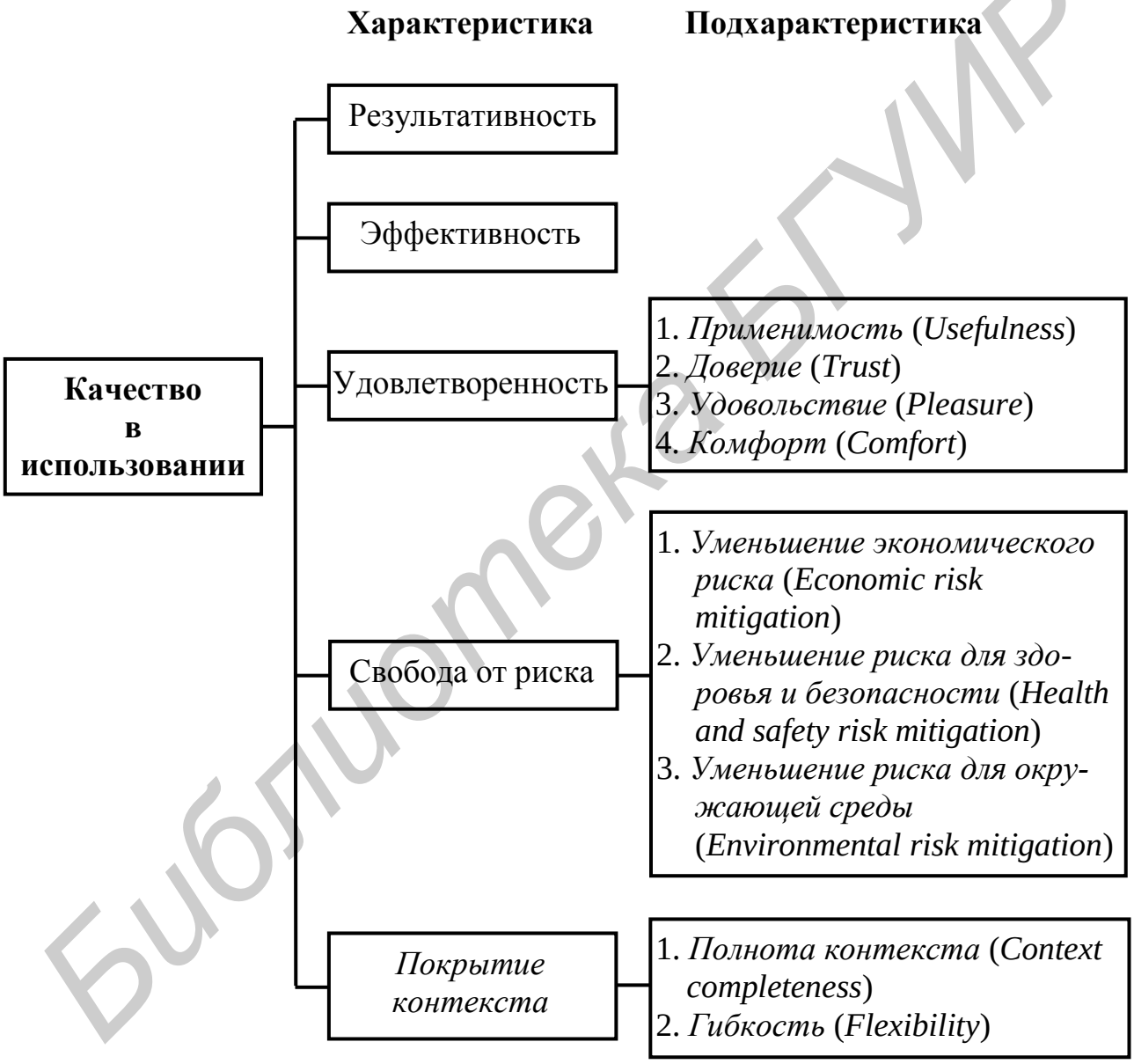


Рис. 1.4. Модель качества в использовании по стандарту ISO/IEC 25010:2011

1.6.5. Модель качества продукта по стандарту ISO/IEC 25010:2011

На рис. 1.5 приведена модель качества продукта. Данная модель подразделяет свойства качества системы или программного продукта на восемь характе-

Библиотека БГУИР

ристик: функциональное соответствие, эффективность функционирования, совместимость, практичность, надежность, защищенность, сопровождаемость, мобильность. Курсивом на рисунке выделены отличия данной модели от модели внутреннего и внешнего качества, определенной в *ISO/IEC 9126–1*:*2001* (см. подразд. 1.4).

Функциональное соответствие (Functional suitability) – степень обеспечения программным продуктом или системой функций, отвечающих заданным или подразумеваемым потребностям при использовании в заданных условиях.

Библиотека БГУИР

Рис. 1.5. Модель качества продукта по стандарту ISO/IEC 25010:2011

Библиотека БГУИР

Подхарактеристиками функционального соответствия являются функциональная полнота (functional completeness), функциональная правильность (functional correctness), функциональная пригодность (functional appropriateness).

Эффективность функционирования (Performance efficiency) – зависимость функционирования от количества ресурсов, используемых в заданных условиях. Ресурсы могут включать другие программные продукты, программную и аппаратную конфигурацию системы, материалы.

Подхарактеристиками эффективности функционирования являются поведение во времени (time behaviour), использование ресурсов (resource utilization), емкость (capacity).

Совместимость (Compatibility) – степень возможностей программного продукта, системы или компонента обмениваться информацией с другими продуктами, системами или компонентами и/или выполнять свои заданные функции при совместном использовании одной и той же аппаратной или программной среды.

Подхарактеристиками совместимости являются сосуществование (coexistence) и способность к взаимодействию (interoperability).

Практичность (Usability) – степень применимости программного продукта или системы заданными пользователями для достижения заданных целей с результативностью, эффективностью и удовлетворенностью в заданном контексте использования.

Подхарактеристиками практичности являются распознаваемость пригодности (appropriateness recognizability), обучаемость (learnability), простота использования (operability), защита от ошибок пользователей (user error protection), эстетичность пользовательского интерфейса (user interface aesthetics), доступность (accessibility).

Надежность (Reliability) – степень выполнения системой, программным продуктом или компонентом заданных функций в заданных условиях в течение заданного периода времени. Для программных средств ограничения надежности связаны с ошибками в требованиях, проекте и реализации или с изменениями контекста.

Подхарактеристиками надежности являются завершенность (maturity), готовность (availability), устойчивость к ошибке (fault tolerance), восстанавливаемость (recoverability).

Защищенность (Security) – степень защиты программным продуктом или системой информации и данных так, чтобы люди, другие продукты или системы имели степень доступа к данным, соответствующую типам и уровням их авторизации.

Подхарактеристиками защищенности являются конфиденциальность (confidentiality), целостность (integrity), неопровержимость (non-repudiation), идентифицируемость (accountability), аутентичность (authenticity).

Сопровождаемость (Maintainability) – степень результативности и продуктивности модификаций программного продукта или системы запланированным персоналом сопровождения. Модификации могут включать исправления, улучшения или адаптацию программного средства к изменениям в окружающей среде, требованиях и функциональных спецификациях.

Библиотека БГУИР

Подхарактеристиками сопровождаемости являются модульность (modularity), повторная используемость (reusability), анализируемость (analysability), модифицируемость (modifiability), тестируемость (testability).

Мобильность (Portability) – степень результативности и эффективности переноса системы, программного продукта или компонента из одной аппаратной, программной или иной эксплуатационной или используемой среды в другую.

Подхарактеристиками мобильности являются адаптируемость (adaptability), простота установки (installability), взаимозаменяемость (replaceability).

1.6.6. Группа стандартов ISO/IEC 2502n

Численные значения свойств качества системы или программного продукта, находящиеся на нижнем уровне иерархической структуры моделей качества, определяются с помощью мер качества. Измерению свойств качества посвящена группа стандартов измерения качества ISO/IEC 2502n.

В данную группу входят следующие стандарты:

– ***ISO/IEC 25020:2007 – Программная инженерия – Требования к качеству и оценка программного продукта (SQuaRE) – Эталонная модель измерений*** [18];

– ***ISO/IEC 25021:2012 – Системная и программная инженерия – Требования к качеству и оценка программного продукта (SQuaRE) – Элементы мер качества*** [19];

– ***ISO/IEC 25022 – Системная и программная инженерия – Требования к качеству и оценка программного продукта (SQuaRE) – Измерение качества в использовании*** (данный стандарт является ревизией стандарта *ISO/IEC TR 9126–4:2004*; в настоящее время находится в разработке);

– ***ISO/IEC 25023 – Системная и программная инженерия – Требования к качеству и оценка программного продукта (SQuaRE) – Измерение качества систем и программных продуктов*** (данный стандарт является ревизией стандартов *ISO/IEC TR 9126–2:2003 ISO/IEC TR 9126–3:2003*; в настоящее время находится в разработке);

– ***ISO/IEC 25024 – Системная и программная инженерия – Требования к качеству и оценка программного продукта (SQuaRE) – Измерение качества данных*** (стандарт находится в разработке).

На рис. 1.6 приведена структура группы стандартов измерения качества.

В стандартах *ISO/IEC 25020:2007* и *ISO/IEC 25021:2012* определено, что основой для нахождения значений мер качества являются элементы мер качества (ЭМК), определяющие значения соответствующих свойств продукта.

Значения ЭМК участвуют в вычислении выражения (называемого в *SQuaRE* функцией измерения), в результате чего определяется значение соответствующей меры качества. Примерами функций измерения являются выражения (1.1) и (1.2) в п. 1.5.1 и в табл. 1.1 – 1.3.

Библиотека БГУИР

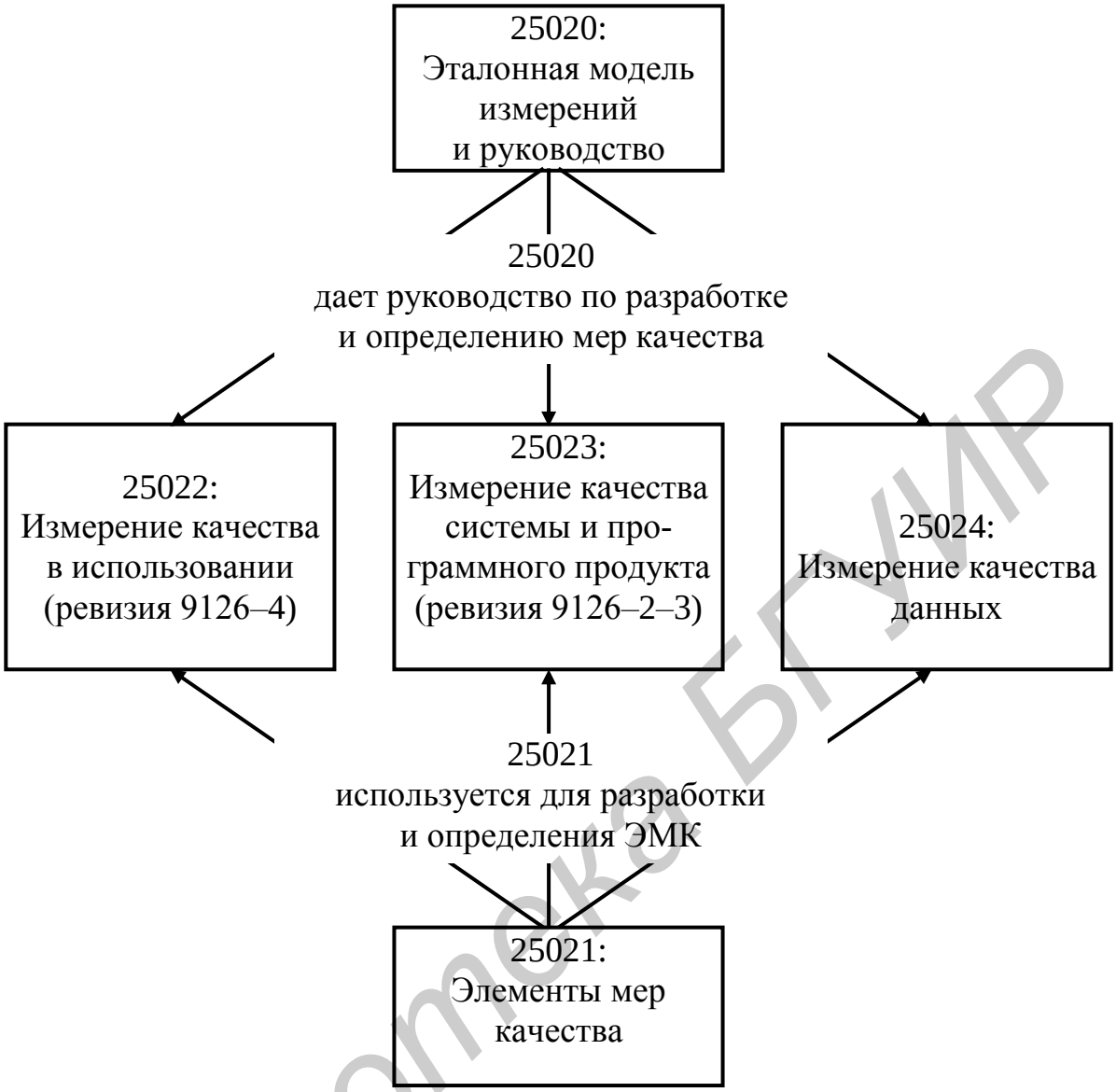


Рис. 1.6. Структура группы измерения качества

В стандарте *ISO/IEC 25021*:*2012* определен начальный набор ЭМК для использования в течение жизненного цикла продукта, приведены правила для проектирования элементов мер качества и верификации существующих ЭМК. Содержимое стандарта устанавливает связь между сериями стандартов *ISO/IEC 9126–1–4*:*2001–2004* и *SQuaRE*. Данную связь отражает рис. 1.7.

Приведенный в стандарте *ISO/IEC 25021*:*2012* набор ЭМК использован при проектировании мер качества, определенных в стандартах *ISO/IEC 9126–2–4*:*2001–2004*. Каждая из мер качества данных стандартов состоит как минимум из двух ЭМК.

Следует отметить, что понятие «мера качества», используемое в стандартах серии *SQuaRE*, эквивалентно понятию «метрика качества» в стандартах *ISO/IEC 9126–1–4*:*2001–2004*. Термин «мера» введен в стандарты *SQuaRE* для обеспечения совместимости со стандартом *ISO/IEC 15939*:*2007* [13].

Библиотека БГУИР

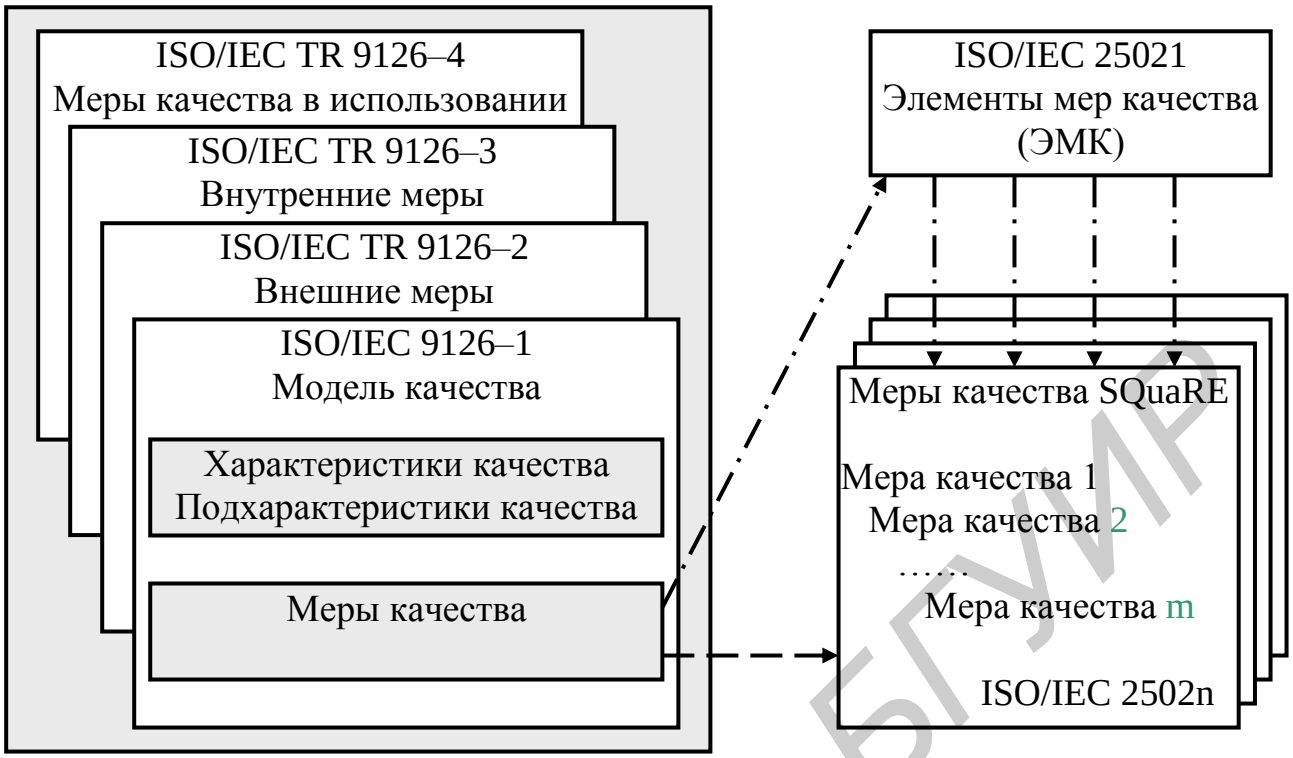


Рис. 1.7. Организация связи между сериями стандартов 9126 и SQuaRE с помощью стандарта ISO/IEC 25021:2012

1.7. Вопросы и задания для самоконтроля

1. Дайте определение качества ПС.
2. Что такое качество в использовании?
3. Что такое внутренняя мера (метрика) качества ПС?
4. Что такое внешняя мера (метрика) качества ПС?
5. Что такое характеристика качества ПС?
6. Назовите стандарты в области оценки качества ПС, действующие на территории Республики Беларусь.
7. Какой международный стандарт послужил основой при разработке *СТБ ИСО/МЭК 9126-2003*?
8. Назовите характеристики и подхарактеристики модели качества ПС по стандарту *СТБ ИСО/МЭК 9126-2003.*
9. Какие серии международных стандартов были разработаны взамен *ISO/IEC 9126*:*1991*?
10. Опишите структуру серии стандартов *ISO/IEC 9126*.
11. Опишите структуру серии стандартов *ISO/IEC 14598*.
12. Чем структура модели внешнего и внутреннего качества, определенная в стандарте *ISO/IEC 9126–1*:*2001*, отличается от структуры модели качества ПС, регламентированной в *ISO/IEC 9126*:*1991*?
13. Опишите структуру модели качества в использовании, определенной в *ISO/IEC 9126–1*:*2001*.

Библиотека БГУИР

14. Какой стандарт заменил собой стандарт *ISO/IEC 9126–1*:*2001*?

15. Назовите желательные свойства и критерии обоснованности метрик (мер).

16. Приведите примеры внутренних метрик качества ПС.

17. Приведите примеры внешних метрик качества ПС.

18. Приведите примеры метрик качества ПС в использовании.

19. Опишите организацию серии стандартов *SQuaRE*.

20. Перечислите стандарты *SQuaRE*, входящие в группу управления качеством, опишите их назначение.

21. Перечислите стандарты *SQuaRE*, входящие в группу модели качества, опишите их назначение.

22. Перечислите стандарты *SQuaRE*, входящие в группу измерения качества, опишите их назначение.

23. Опишите структуру моделей качества по стандарту *ISO/IEC 25010*:*2011*.

24. Какие виды моделей качества определены в стандарте *ISO/IEC 25010*:*2011*?

25. Назовите характеристики и подхарактеристики модели качества в использовании по стандарту *ISO/IEC 25010*:*2011*.

26. Назовите характеристики и подхарактеристики модели качества продукта по стандарту *ISO/IEC 25010*:*2011*.

27. Опишите структуру стандартов измерения качества.

28. Что такое элемент меры качества?

29. Как связаны элементы мер качества с мерами качества?

2. СЛОЖНОСТЬ ПРОГРАММНЫХ СРЕДСТВ

Качество программных средств во многом зависит от их сложности. Например, чем сложнее программа, тем ниже ее надежность и сопровождаемость. Поэтому при оценке качества программ обычно оценивается и их сложность.

Сложность программных средств определяется рядом факторов. Основными из них являются сложность архитектуры и технического проекта ПС, а также сложность кодов ПС.

Сложность программных средств принято оценивать с помощью различных мер (в литературе широко используется как термин «мера сложности», так и термин «метрика сложности»).

Метрики (меры) сложности программ обычно подразделяются на ***три основные группы*** [27, 29]:

− метрики размера программ;

− метрики сложности потока управления программ;

− метрики сложности потока данных программ.

В общем случае метрики сложности могут рассматриваться как элементы мер качества ПС. В конкретных случаях метрики сложности могут использоваться как внутренние меры качества программных средств.

Библиотека БГУИР

2.1. Метрики размера программ

Метрики этой группы основаны на анализе исходных текстов программ.

Существуют различные метрики, с помощью которых может быть оценен размер программы.

К наиболее простым метрикам размера программы относятся ***количество строк исходного текста программы*** и ***количество операторов программы***.

Из метрик размера программ широкое распространение получили ***метрики Холстеда*** [28].

Основу метрик Холстеда составляют *шесть базовых метрик* программы:

– η_1 – словарь операторов (число уникальных операторов программы);

– η_2 – словарь операндов (число уникальных операндов программы);

– N_1 – общее число операторов в программе;

– N_2 – общее число операндов в программе;

– f_{1j} – число вхождений j-го оператора, $j = 1, 2, \ldots, \eta_1$;

– f_{2i} – число вхождений i-го операнда, $i = 1, 2, \ldots, \eta_2$.

Справедливы следующие соотношения:

$$N_1 = \sum_{j=1}^{\eta_1} f_{1j} \qquad N_2 = \sum_{i=1}^{\eta_2} f_{2i}$$

Базовые метрики определяются непосредственно при анализе исходных текстов программ. На основе базовых метрик Холстед предложил рассчитывать ряд производных метрик программы. Наиболее широко из них используются следующие метрики:

– словарь программы (общее число уникальных операторов и операндов программы):

$$\eta = \eta_1 + \eta_2; \qquad (2.1)$$

– длина программы (общее количество операторов и операндов программы):

$$N = N_1 + N_2; \qquad (2.2)$$

– объем программы (число логических единиц информации, необходимых для записи программы):

$$V = N \log_2 \eta. \qquad (2.3)$$

Операнды программы представляют собой используемые в ней переменные и константы.

Под операторами программы Холстед подразумевает входящие в ее состав символы-ограничители (в том числе символы операций, символы скобок и символы-разделители), управляющие операторы, а также имена подпрограмм (процедур и функций). При этом парные символы (например пара из открывающей и закрывающей скобок) считаются одним оператором. Несколько служебных

Библиотека БГУИР

слов, входящих в состав одного оператора (например, If…Then…Else), также считаются одним оператором.

Метки не относятся ни к операторам, ни к операндам.

При подсчете операторов Холстеда объявления и инициализацию элементов программы (например, типов, констант, переменных) вместе с соответствующими символами-ограничителями принято не учитывать.

Очевидно, что совокупность операторов программы и их количество зависят от языка программирования, на котором написана программа.

В табл. 2.1 приведены основные операторы процедурно-ориентированного подмножества языка Delphi в интерпретации Холстеда. При подсчете количества операторов и операндов в программе, написанной на языке Delphi, обычно анализируется только ее раздел операторов, а также разделы операторов процедур и функций пользователя.

Таблица 2.1

Операторы языка Delphi в интерпретации Холстеда

Обозначение оператора	Назначение оператора
1	2
+	плюс (сложение, объединение множеств, сцепление строк)
–	минус (изменение знака, вычитание, разность множеств)
*	звездочка (умножение, пересечение множеств)
/	наклонная черта, слэш (деление)
<	меньше
>	больше
=	равно
.	точка (разделитель полей при обращении к элементам записи)
,	запятая (разделитель в перечислениях)
:	двоеточие (отделяет константы выбора в операторе Case)
;	точка с запятой (разделитель операторов программы)
( )	левая и правая скобки при выделении подвыражений
[ ]	левая и правая квадратные скобки (выделяет индексы элементов массивов)
<=	меньше или равно
>=	больше или равно
<>	неравно
:=	операция присваивания
^	знак карата (обращение к динамической переменной)
@	коммерческое ‘at’ (операция взятия адреса элемента)
And	операция поразрядного логического сложения (И)
Not	операция поразрядного дополнения (НЕ)

Библиотека БГУИР

Окончание табл. 2.1

1	2
Or	операция поразрядного логического сложения (ИЛИ)
Xor	операция поразрядного логического исключающего ИЛИ
Div	целочисленное деление
Mod	остаток от целочисленного деления
Shl	операция сдвига влево
Shr	операция сдвига вправо
In	операция проверки вхождения элемента в множество
Begin…End	составной оператор
Break	оператор безусловного выхода из цикла
Continue	оператор передачи управления на конец тела цикла
Goto <Метка>	оператор безусловного перехода
Case…Of… Else…End	оператор варианта
If…Then…Else	оператор условного перехода
Repeat…Until	оператор цикла с постусловием
While…Do	оператор цикла с предусловием
For…To…Do	оператор цикла с параметром (с увеличением параметра)
For…Downto… Do	оператор цикла с параметром (с уменьшением параметра)
With…Do	оператор присоединения

В табл. 2.2 приведены основные операторы языка *C* в интерпретации Холстеда.

Таблица 2.2

Операторы языка *C* в интерпретации Холстеда

Обозначение оператора	Назначение оператора
1	2
=	операция присваивания
+	сложение
–	вычитание
*****	звездочка (умножение, обращение к динамической переменной)
/	деление
%	остаток от целочисленного деления
++	инкремент
--	декремент
==	равно
!=	неравно

Библиотека БГУИР

Продолжение табл. 2.2

1	2
>	больше
<	меньше
>=	больше или равно
<=	меньше или равно
!	логическое отрицание (НЕ)
&&	логическое И
||	логическое ИЛИ
~	побитовая инверсия
&	побитовое И, ссылка
|	побитовое ИЛИ
^	побитовое исключающее ИЛИ
<<	побитовый сдвиг влево (в сторону старших разрядов)
>>	побитовый сдвиг вправо (в сторону младших разрядов)
+=	присваивание с суммированием
-=	присваивание с вычитанием
*=	присваивание с умножением
/=	присваивание с делением
%=	присваивание по модулю
&=	присваивание с побитовым И
|=	присваивание с побитовым ИЛИ
^=	присваивание с побитовым исключающим ИЛИ
<<=	присваивание с побитовым сдвигом влево
>>=	присваивание с побитовым сдвигом вправо
( )	левая и правая скобки при выделении подвыражений
[]	обращение к элементу массива
->	динамическое обращение к элементу структуры
.	статическое обращение к элементу структуры
,	запятая (операция и разделитель)
;	точка с запятой (разделитель операторов программы)
:	двоеточие (отделяет константы выбора в операторе switch)
?…:	условный оператор
sizeof	размер
(type)	преобразование типа
{}	составной оператор
if…else	оператор выбора
switch()…case… default	оператор множественного выбора
do…while( )	оператор цикла с постусловием
while( )	оператор цикла с предусловием

Библиотека БГУИР

Окончание табл. 2.2

1	2
for( )	оператор цикла с параметром
goto <метка>	оператор безусловного перехода
continue	оператор перехода к следующему шагу цикла
break	оператор выхода из цикла

Пример 1

Расчет метрик Холстеда для программы, вычисляющей значение функции $Y = \sin X$ через разложение функции в бесконечный ряд

$$Y = \sin X = X - X^3 / 3! + X^5 / 5! - X^7 / 7! + \ldots$$

с точностью $Eps = 0{,}0001$.

Текст программы на языке Delphi, реализующей вычисление функции Y, приведен ниже.

```
Program  Sin1;
   Const
      eps = 0.0001;
   Var
      y, x: real;  n: integer;  vs: real;
   Begin
      Readln (x);
      y := x; {Начальные установки}
      n := 2;
      vs := x;
      Repeat
        vs := –vs * x * x / (2 * n – 1) / (2 * n – 2); {Формирование слагаемого}
        n := n + 1;
        y := y + vs
      Until  abs(vs) < eps;  {Выход из цикла по выполнению условия}
      Writeln (x, y, eps)
   End.
```

Расчет базовых метрик Холстеда для данной программы приведен в табл. 2.3. По формулам (2.1) – (2.3) рассчитываются словарь, длина и объем программы.

Словарь программы: $\eta = 14 + 7 = 21$.

Длина программы: $N = 34 + 28 = 62$.

Объем программы: $V = 62 \log_2 21 \approx 272$.

Библиотека БГУИР

Таблица 2.3

Расчет базовых метрик Холстеда для программы, вычисляющей значение функции *Y* = sin *X*

j	Оператор	f_{1j}	***i***	Операнд	f_{2i}
1.	;	7	1.	x	6
2.	:=	6	2.	n	5
3.	*	4	3.	vs	5
4.	–	3	4.	y	4
5.	/	2	5.	2	4
6.	( )	2	6.	1	2
7.	+	2	7.	eps	2
8.	,	2			
9.	Begin…End	1			
10.	Readln ( )	1			
11.	Repeat…Until	1			
12.	abs( )	1			
13.	<	1			
14.	Writeln ( )	1			
η_1 = 14		**N_1 = 34**	**η_2 = 7**		**N_2 = 28**

Пример 2

Расчет метрик Холстеда для программы на языке *C*. Программа реализует вычисление той же функции ***Y* = sin *X***, что и программа из примера 1.

```
#include <stdio.h>
#include <stdlib.h>
main()
{
   const double eps = 0.0001;
   double y, x;
   int n;
   double vs;
   scanf("%f", &x);
   y = x; // Начальные установки
   n = 2;
   vs = x;
   do
   {
      vs = –vs * x * x / (2 * n – 1) / (2 * n – 2);  // Формирование слагаемого
      n++;
      y += vs;
   }
```


Библиотека БГУИР

```
    while (abs(vs) >= eps); // Выход из цикла по выполнению условия
    printf("%f %f %f\n", x, y, eps);
}
```

Расчет базовых метрик Холстеда для данной программы приведен в табл. 2.4. По формулам (2.1) – (2.3) рассчитываются словарь, длина и объем программы.

Словарь программы: $\eta = 16 + 6 = 22$.

Длина программы: $N = 38 + 24 = 62$.

Объем программы: $V = 62 \log_2 22 \approx 276$.

Таблица 2.4

Расчет базовых метрик Холстеда для программы, вычисляющей значение функции *Y* = sin *X* на языке *C*

j	Оператор	f_{1j}	i	Операнд	f_{2i}
1.	;	9	1.	x	6
2.	=	4	2.	vs	5
3.	*	4	3.	n	4
4.	,	4	4.	2	4
5.	–	3	5.	y	3
6.	/	2	6.	eps	2
7.	( )	2			
8.	{…}	2			
9.	scanf ( )	1			
10.	do…while ( )	1			
11.	abs( )	1			
12.	>=	1			
13.	printf ( )	1			
14.	++	1			
15.	+=	1			
16.	&	1			
$\eta_1 = 16$		$N_1 = 38$	$\eta_2 = 6$		$N_2 = 24$

2.2. Метрики сложности потока управления программ

Метрики сложности потока управления программ принято определять на основе представления программ в виде управляющего ориентированного графа $G = (V, E)$, где V – вершины, соответствующие операторам, а E – дуги, соответствующие переходам между операторами [27, 29]. В дуге (v, u) вершина v является исходной, а u – конечной. При этом u непосредственно следует за v, а v непосредственно предшествует u. Если путь от v до u состоит более чем из одной дуги, тогда u следует за v, а v предшествует u.

Библиотека БГУИР

Частным случаем представления ориентированного графа программы можно считать построенную в соответствии с положениями стандарта *ГОСТ 19.701–90* [1] детализированную схему алгоритма, в которой каждому блоку соответствует один оператор программы. Аналогами вершин графа являются блоки алгоритма, причем данные блоки имеют разное графическое представление в зависимости от их назначения. Дугам графа соответствуют линии передачи управления между блоками алгоритма.

Ниже рассмотрены наиболее распространенные метрики сложности потока управления программ.

Метрика Маккейба (цикломатическая сложность графа программы, цикломатическое число Маккейба) предназначена для оценки трудоемкости тестирования программы. Данная метрика определяется по формуле

$$\boldsymbol{Z(G) = e - \upsilon + 2p},$$

где $\boldsymbol{e}$ – число дуг ориентированного графа $\boldsymbol{G}$; υ – число вершин; $\boldsymbol{p}$ – число компонентов связности графа.

Число компонентов связности графа – количество дуг, которые необходимо добавить для преобразования графа в сильносвязный. Сильносвязным графом называется граф, любые две вершины которого взаимно достижимы. Для корректных программ, не имеющих недостижимых от начала программы участков и «висячих» точек входа и выхода, сильносвязный граф получается путем соединения дугой вершины, обозначающей конец программы, с вершиной, обозначающей начало этой программы.

Метрика Маккейба определяет минимальное количество тестовых прогонов программы, необходимых для тестирования всех ее ветвей.

Рассчитаем метрику Маккейба для программы, схема алгоритма которой приведена на рис. 2.1 (для простоты понимания ввод/вывод в данной схеме алгоритма не показан, хотя в реальных вычислениях метрики блоки ввода/вывода следует учитывать). Компонент связности графа обозначен штриховой дугой. Число дуг $\boldsymbol{e} = 8$, число вершин $\upsilon = 7$, $\boldsymbol{p} = 1$. Цикломатическое число Маккейба равно $\boldsymbol{Z(G)} = 8 - 7 + 2 = 3$.

Следует обратить внимание, что при расчете метрики Маккейба начальную и конечную вершины алгоритма (блоки «Начало» и «Конец») необходимо учитывать.

Библиотека БГУИР

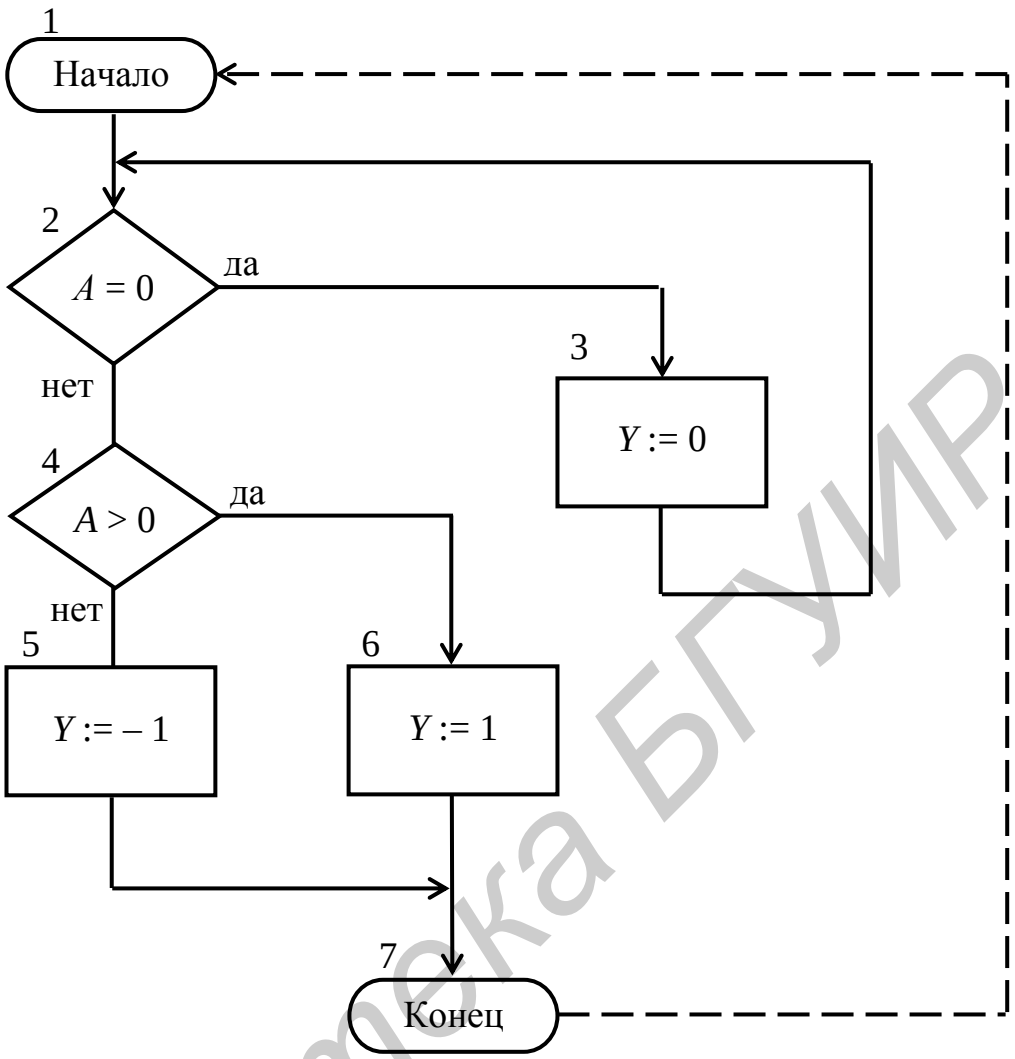


Рис. 2.1. Пример схемы алгоритма программы

Значение метрики Маккейба показывает, что в схеме алгоритма (см. рис. 2.1) можно выделить *три базовых независимых пути* (называемых также линейно независимыми контурами):

1) 1 – 2 (нет) – 4 (нет) – 5 – 7;
2) 1 – 2 (да) – 3 – 2 (нет) – 4 (нет) – 5 – 7;
3) 1 – 2 (нет) – 4 (да) – 6 – 7.

Вторым возможным вариантом совокупности базовых независимых путей является:

1) 1 – 2 (да) – 3 – 2 (нет) – 4 (нет) – 5 – 7;
2) 1 – 2 (нет) – 4 (нет) – 5 – 7;
3) 1 – 2 (да) – 3 – 2 (нет) – 4 (да) – 6 – 7.

Цифры в обозначении путей представляют собой номера блоков в схеме алгоритма программы.

Таким образом, для тестирования совокупности базовых независимых путей исследуемой программы необходимо выполнить минимально три тестовых прогона.

Библиотека БГУИР

Метрика Джилба определяет логическую сложность программы как насыщенность программы условными операторами IF–THEN–ELSE и операторами цикла. Обычно используются два вида метрики Джилба: ***CL*** – количество условных и циклических операторов, характеризующее абсолютную сложность программы; ***cl*** – насыщенность программы условными и циклическими операторами, характеризующая относительную сложность программы; ***cl*** определяется как отношение ***CL*** к общему количеству операторов программы (здесь под оператором подразумевается оператор конкретного языка программирования в классическом представлении, а не в интерпретации Холстеда).

Расширением метрики Джилба является *максимальный уровень вложенности условного и циклического оператора* ***CLI***.

Использование в программе оператора выбора (например, CASE в языке Delphi) с ***n*** разветвлениями эквивалентно применению ***n*** – 1 оператора IF–THEN–ELSE с глубиной вложенности ***n*** – 2.

Например, на рис. 2.2 приведена схема алгоритма вычисления некоторой функции ***Y***. В данной схеме используется выбор, обозначаемый символом «Решение» (ромб) с пятью разветвлениями (***n*** = 5).

Эквивалентный алгоритм вычисления той же функции ***Y***, использующий несколько операторов IF–THEN–ELSE, представлен на рис. 2.3.

На данном рисунке количество операторов IF–THEN–ELSE равно четырем (***n*** – 1), максимальный уровень вложенности оператора IF–THEN–ELSE равен трем (***n*** – 2).

Таким образом, для схем алгоритмов, приведенных на рис. 2.2 и 2.3, ***CL*** = 4, ***cl*** = 0,36 (количество операторов программы равно 11; блоки «Начало» и «Конец» в метрике Джилба не учитываются), ***CLI*** = 3.

Значения метрики Маккейба для данных алгоритмов также совпадают.

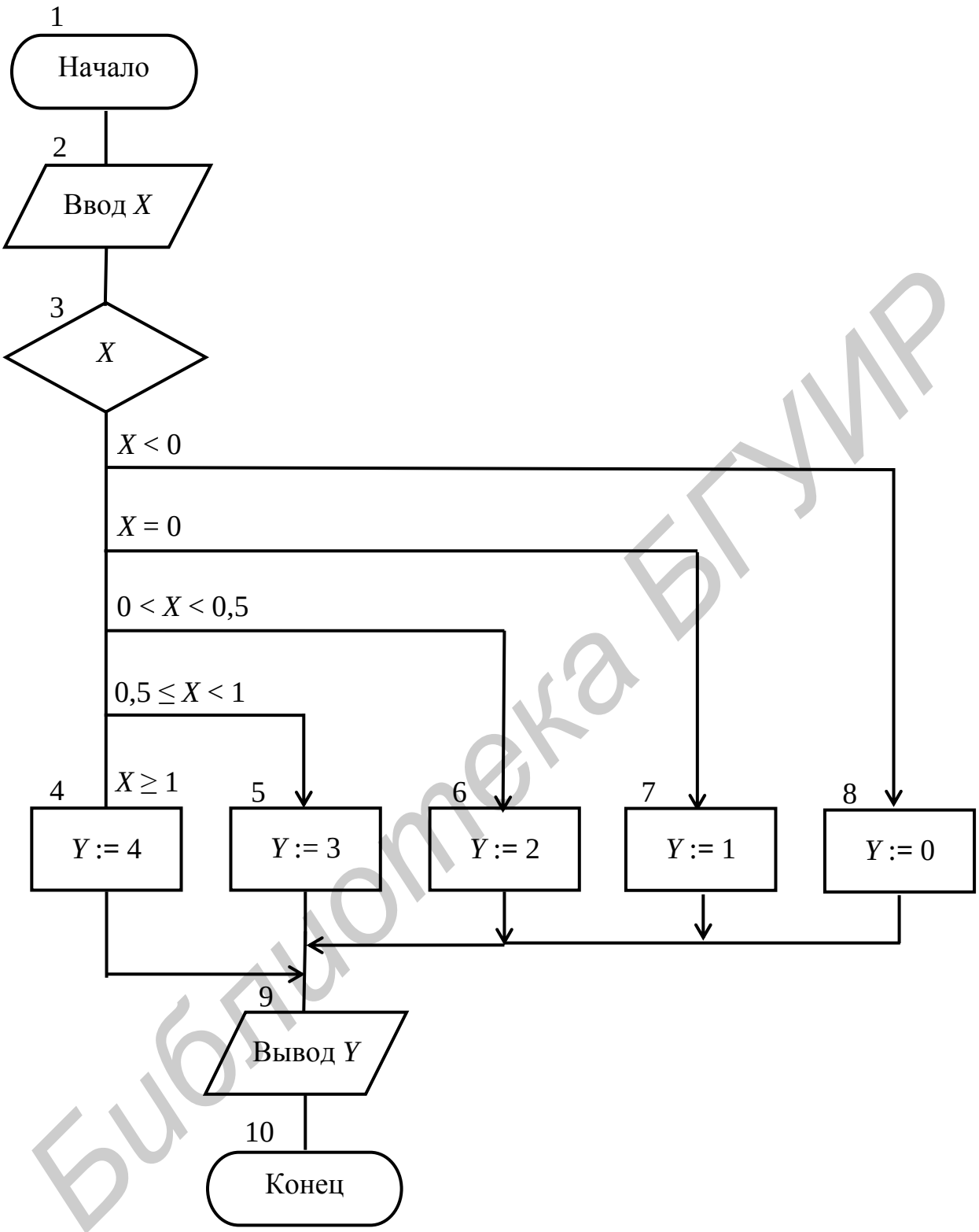
Для схемы алгоритма, представленной на рис. 2.2, ***Z*(*G*)** = 13 – 10 + 2 = 5.

Для схемы алгоритма, приведенной на рис. 2.3, ***Z*(*G*)** = 16 – 13 + 2 = 5.

Для схемы алгоритма, приведенной на рис. 2.1, значение метрики Джилба ***CL*** = 2, ***cl*** = 0,4 (количество операторов программы равно 5), ***CLI*** = 0.

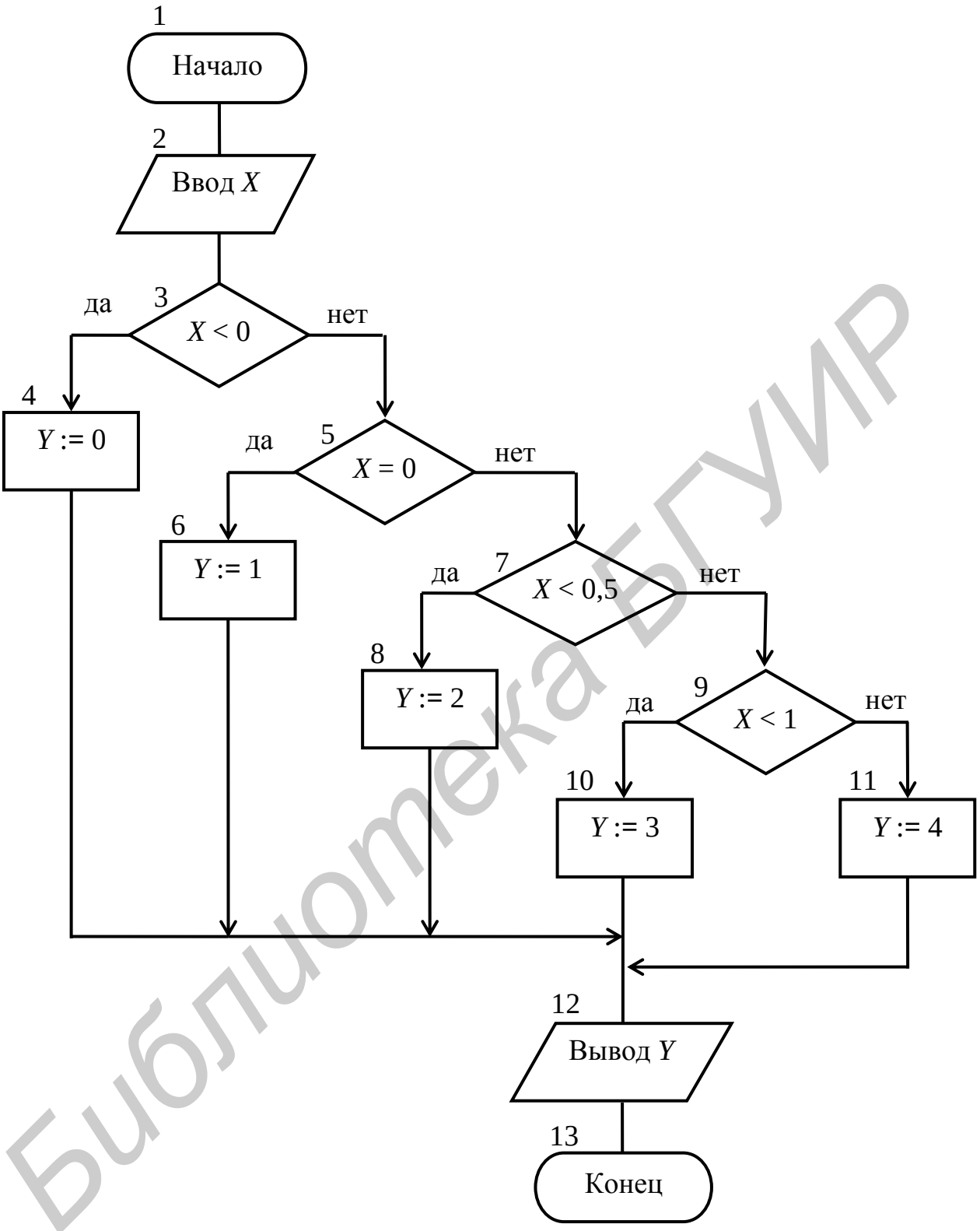
Следует отметить, что сложность программы с помощью метрики Джилба не всегда возможно посчитать на основе схемы алгоритма, т. к. схема алгоритма может быть представлена укрупнено. Поэтому значения метрики Джилба в программе в общем случае следует определять на основе ее исходного текста или детализированной схемы алгоритма, каждый блок которой содержит один оператор программы (как это реализовано в алгоритмах, представленных на рис. 2.1 – 2.3).

Библиотека БГУИР



Библиотека БГУИР

Рис. 2.2. Схема разветвляющегося алгоритма вычисления функции ***Y*** (используется символ «Решение» со многими выходами)



Библиотека БГУИР

Рис. 2.3. Схема разветвляющегося алгоритма вычисления функции ***Y*** (используется символ «Решение» с двумя выходами)

Метрика граничных значений базируется на определении скорректированной сложности вершин графа программы [27].

Пусть $G = (V, E)$ – ориентированный граф программы с единственной начальной и единственной конечной вершинами. В этом графе число входящих в вершину дуг называется *отрицательной степенью вершины*, а число исходящих из вершины дуг – *положительной степенью вершины*. С учетом этого набор вершин графа можно разбить на две группы: вершины, у которых положительная степень меньше или равна 1; вершины, у которых положительная степень больше или равна 2. Вершины первой группы называются *принимающими вершинами*, вершины второй группы – *вершинами выбора* (или предикатными вершинами, условными вершинами, вершинами отбора).

Для оценки сложности программы с использованием метрики граничных значений граф G разбивается на максимальное число подграфов, удовлетворяющих следующим условиям: вход в подграф осуществляется через вершину выбора; каждый подграф включает вершину (нижнюю границу подграфа), в которую можно попасть из любой другой вершины подграфа.

Каждая вершина выбора имеет скорректированную сложность, равную числу вершин, образующих связанный с ней подграф. Каждая принимающая вершина имеет скорректированную сложность, равную 1. Конечная вершина графа имеет скорректированную сложность, равную 0.

Следует обратить внимание, что если подграф представляет собой обычное разветвление, то вершина выбора, через которую осуществляется вход в такой подграф, не является элементом подграфа. Поэтому при расчете скорректированной сложности такой вершины выбора сама эта вершина не учитывается.

Если подграф представляет собой цикл, то по одной из исходящих дуг вершины выбора осуществляется вход в тело цикла, а по другой – выход из цикла. В этом случае нижней границей подграфа является вершина перехода, находящаяся после вершины выбора на дуге выхода из цикла. После входа в подграф через вершину выбора и выполнения тела цикла для достижения нижней границы подграфа следует снова возвращаться к данной вершине выбора. Таким образом, вершина выбора, через которую осуществляется вход в циклический подграф, является элементом данного подграфа. Поэтому данную вершину выбора следует учитывать при подсчете ее скорректированной сложности.

Абсолютная граничная сложность программы S_a определяется как сумма скорректированных сложностей всех вершин графа G.

Относительная граничная сложность программы S_o определяется по формуле

$$S_o = 1 - \frac{\upsilon - 1}{S_a},$$

где υ – общее число вершин графа программы.

В табл. 2.5 представлены свойства подграфов программы, схема алгоритма которой приведена на рис. 2.3. Скорректированные сложности вершин графа данной программы содержит табл. 2.6. Номера вершин графа соответствуют номерам соответствующих блоков на схеме алгоритма.

Библиотека БГУИР

В рассматриваемой схеме алгоритма все подграфы представляют собой разветвления. Поэтому вершины выбора в данные подграфы не входят и при расчете своей скорректированной сложности не учитываются (см. табл. 2.5).

Таблица 2.5

Свойства подграфов программы*

Свойства подграфов программы	Номер вершины выбора			
	3	5	7	9
Номера вершин перехода	4, 5	6, 7	8, 9	10, 11
Номера вершин подграфа	4, 5, 6, 7, 8, 9, 10, 11	6, 7, 8, 9, 10, 11	8, 9, 10, 11	10, 11
Номер нижней границы подграфа	12	12	12	12
Скорректированная сложность вершины выбора	9	7	5	3

* Схема алгоритма программы приведена на рис. 2.3.

Таблица 2.6

Скорректированные сложности вершин графа программы**

Номер вершины графа программы	1	2	3	4	5	6	7	8	9	10	11	12	13	S_a
Скорректированная сложность вершины графа	1	1	9	1	7	1	5	1	3	1	1	1	0	**32**

** Схема алгоритма программы представлена на рис. 2.3.

Таким образом, абсолютная граничная сложность S_a программы, схема алгоритма которой приведена на рис. 2.3, равна 32. Относительная граничная сложность данной программы равна

$$S_o = l - (13 - 1)/32 = 0{,}625.$$

В табл. 2.7 представлены свойства подграфов программы, схема алгоритма которой приведена на рис. 2.1. Скорректированные сложности вершин графа данной программы содержит табл. 2.8.

В рассматриваемой схеме алгоритма подграф с вершиной выбора 2 содержит цикл. Поэтому данная вершина выбора является элементом подграфа и участвует при подсчете своей скорректированной сложности (см. табл. 2.7).

Абсолютная граничная сложность S_a программы, схема алгоритма которой приведена на рис. 2.1, равна 10. Относительная граничная сложность данной программы равна

$$S_o = l - (7 - 1)/10 = 0{,}4.$$

Библиотека БГУИР

Таблица 2.7

Свойства подграфов программы*

Свойства подграфов программы	Номер вершины выбора	
	2	4
Номера вершин перехода	3, 4	5, 6
Номера вершин подграфа	2, 3	5, 6
Номер нижней границы подграфа	4	7
Скорректированная сложность вершины выбора	3	3

* Схема алгоритма программы представлена на рис. 2.1.

Таблица 2.8

Скорректированные сложности вершин графа программы**

Номер вершины графа программы	1	2	3	4	5	6	7	S_a
Скорректированная сложность вершины графа	1	3	1	3	1	1	0	**10**

** Схема алгоритма программы представлена на рис. 2.1.

Метрики сложности потока управления для программ, схемы алгоритмов которых приведены на рис. 2.1 – 2.3, содержит табл. 2.9.

Таблица 2.9

Метрики сложности потока управления программ

Метрики сложности потока управления	Схемы алгоритмов		
	Рис. 2.1	Рис. 2.2	Рис. 2.3
Метрика Маккейба ***Z*(*G*)**	3	5	5
Абсолютная сложность программы ***CL*** по метрике Джилба	2	4	4
Относительная сложность программы ***cl*** по метрике Джилба	0,4	0,36	0,36
Максимальный уровень вложенности условного оператора ***CLI*** по метрике Джилба	0	3	3
Метрика граничных значений S_a (абсолютная граничная сложность программы)	10	14	32
Метрика граничных значений S_o (относительная граничная сложность программы)	0,4	0,357	0,625

Библиотека БГУИР

2.3. Метрики сложности потока данных

Метрики сложности потока данных связывают сложность программы с использованием и размещением данных в этой программе. Метрики данной группы основаны на анализе исходных текстов программ.

К наиболее известным в рассматриваемой группе метрик можно отнести спен и метрику Чепина [27, 29].

Спен идентификатора – число повторных появлений идентификатора (число появлений после его первого появления) в тексте программы. Идентификатор, встречающийся в тексте программы n раз, имеет спен, равный $n - 1$.

Величина спена связана со сложностью тестирования и отладки программы. Например, если спен идентификатора равен 10, то при трассировании программы по этому идентификатору следует ввести в текст программы 10 контрольных точек, что усложняет тестирование и отладку программы.

Метрика Чепина базируется на анализе характера использования переменных в программе.

Существуют различные варианты метрики Чепина. Ниже рассмотрен вариант (назовем данный вариант полной метрикой Чепина), в котором все множество переменных программы разбивается на четыре функциональные группы:

1. P – вводимые переменные, содержащие исходную информацию, но не модифицируемые в программе и не являющиеся управляющими переменными;
2. M – вводимые модифицируемые переменные и создаваемые внутри программы константы и переменные, не являющиеся управляющими переменными;
3. C – переменные, участвующие в управлении работой программы (управляющие переменные);
4. T – не используемые в программе («паразитные») переменные, например, вычисленные переменные, значения которых не выводятся и не участвуют в дальнейших вычислениях.

Значение метрики Чепина определяется по формуле

$$Q = a_1 p + a_2 m + a_3 c + a_4 t ,$$

где a_1, a_2, a_3, a_4 – весовые коэффициенты; p, m, c, t – количество переменных в группах P, M, C, T соответственно.

Весовые коэффициенты позволяют учитывать различное влияние на сложность программы каждой функциональной группы. Наиболее часто применяются следующие значения весовых коэффициентов: $a_1 = 1$, $a_2 = 2$, $a_3 = 3$, $a_4 = 0{,}5$. С учетом данных значений формула для определения метрики Чепина принимает вид

$$Q = p + 2m + 3c + 0{,}5t .$$

Помимо полной метрики Чепина распространен ее вариант, при котором анализу и разбиению на четыре группы подвергаются только переменные из списка ввода/вывода программы, т. е. те переменные, которые содержатся в

Библиотека БГУИР

списке параметров операторов ввода/вывода программы. Назовем данный вариант метрикой Чепина ввода/вывода.

Метрики сложности потока данных для программы на языке Delphi, вычисляющей значение функции ***Y* = sin *X***, содержат табл. 2.10, 2.11. Исходный текст программы приведен в примере 1.

В тексте программы (см. пример 1) идентификаторы впервые встречаются в разделе объявлений. Поэтому значение спена ***i***-го идентификатора равно количеству его появлений в разделе операторов, т. е. значению f_{2i} в метриках Холстеда.

В список переменных ввода/вывода данной программы входят переменные ***x***, ***y*** и константа ***eps***, являющиеся параметрами операторов ввода и вывода *Readln* и *Writeln*. Остальные переменные (***n***, ***vs***) в расчете метрики Чепина ввода/вывода не участвуют.

Таблица 2.10

Спен программы

Идентификатор	***x***	***n***	***vs***	***y***	***eps***	Суммарный спен программы
Спен	6	5	5	4	2	22

Таблица 2.11

Метрики Чепина программы

Переменные	Полная метрика Чепина				Метрика Чепина ввода/вывода			
Группа переменных	***P***	***M***	***C***	***T***	***P***	***M***	***C***	***T***
Переменные, относящиеся к группе	***x***	***y***, ***n***	***vs***, ***eps***	–	***x***	***y***	***eps***	–
Количество переменных в группе	***p*** = 1	***m*** = 2	***c*** = 2	***t*** = 0	***p*** = 1	***m*** = 1	***c*** = 1	***t*** = 0
Метрика Чепина	***Q*** = 1*1 + 2*2 + 3*2 + + 0,5*0 = 11				***Q*** = 1*1 + 2*1 + 3*1 + + 0,5*0 = 6			

2.4. Вопросы и задания для самоконтроля

1. На какие группы принято подразделять метрики сложности программ?
2. Что является основой для подсчета метрик размера программ?
3. Перечислите метрики, относящиеся к метрикам размера программ.
4. Назовите базовые метрики программы, являющиеся основой метрик Холстеда.
5. Что подразумевается под операторами в метриках Холстеда?
6. Что такое словарь операторов, словарь операндов, словарь программы в метриках Холстеда?
7. Что такое длина и объем программы в метриках Холстеда?
8. Перечислите метрики, относящиеся к метрикам сложности потока управления программ.

Библиотека БГУИР

9. Как определяется значение метрики Маккейба?
10. Что показывает значение метрики Маккейба?
11. Как определяется значение метрики Джилба?
12. Какие существуют виды и расширения метрики Джилба?
13. Как определяется значение метрики граничных значений?
14. Что такое отрицательная и положительная степени вершин графа в метрике граничных значений?
15. Дайте определение принимающих вершин и вершин выбора графа в метрике граничных значений.
16. Перечислите метрики, относящиеся к метрикам сложности потока данных программ.
17. Что такое спен идентификатора? Как он рассчитывается?
18. На какой информации базируется метрика Чепина?
19. Как классифицируются переменные в метрике Чепина?
20. Назовите варианты метрики Чепина.
21. Как рассчитывается значение метрики Чепина?

3. ЛАБОРАТОРНАЯ РАБОТА №1

3.1. Тема задания

Метрики сложности потока управления программ

Для заданного преподавателем варианта индивидуального задания разработать детализированную схему алгоритма, представленную в соответствии с положениями *ГОСТ 19.701–90*. По полученному алгоритму рассчитать метрики сложности потока управления программ (метрики Маккейба, Джилба, максимальный уровень вложенности условного оператора и оператора цикла, метрику граничных значений). В алгоритме предусмотреть вывод на экран всех входных и выходных данных.

3.2. Методические указания к выполнению лабораторной работы

Для полученного варианта индивидуального задания следует разработать схему алгоритма с максимальным уровнем детализации. Это означает, что каждому блоку схемы алгоритма должен соответствовать один оператор языка программирования (применительно к заданному преподавателем для второй лабораторной работы языку). Например, для языков Delphi и *C* ввод или вывод двухмерного массива в алгоритме должен быть представлен не одним символом ввода (параллелограмм), а сложным циклом (во внешнем цикле изменяется номер строки, во внутреннем цикле – номер столбца массива).

Схема алгоритма должна быть описана. В описании должно быть приведено назначение входных, выходных и внутренних переменных, назначение блоков алгоритма.

На основании разработанного алгоритма рассчитываются значения метрик сложности потока управления будущей программы (по аналогии с примерами, приведенными в подразд. 2.2):

Библиотека БГУИР

– рассчитывается метрика Маккейба и определяются базовые независимые пути в алгоритме (по аналогии с примером по метрике Маккейба, приведенным в подразд. 2.2);

– рассчитываются абсолютная ***CL*** и относительная ***cl*** сложности программы, а также максимальный уровень вложенности условного и циклического операторов ***CLI***, используя метрику Джилба;

– рассчитываются абсолютная S_a и относительная S_o граничные сложности программы по метрике граничных значений. Результаты расчетов метрики граничных значений должны быть представлены в виде таблиц, аналогичных табл. 2.5 и 2.6;

– значения всех рассчитанных метрик сложности потока управления для разработанного алгоритма должны быть сведены в итоговую таблицу (аналогичную табл. 2.9, но включающую не три столбца для трех схем алгоритмов, а один столбец, соответствующий разработанному алгоритму).

3.3. Содержание отчета по лабораторной работе

Лабораторная работа №1 должна содержать:

– титульный лист;

– номер и условие индивидуального задания;

– детализированную схему алгоритма по ГОСТ 19.701–90;

– описание схемы алгоритма;

– расчет метрики Маккейба для разработанного алгоритма и определение базовых независимых путей;

– расчет метрики Джилба для разработанного алгоритма;

– расчет метрики граничных значений для разработанного алгоритма с результатами, представленными в виде таблиц;

– результаты расчетов метрик в виде итоговой таблицы.

4. ЛАБОРАТОРНАЯ РАБОТА №2

4.1. Тема задания

Метрики Холстеда. Метрики сложности потока данных

Для разработанной в лабораторной работе №1 схемы алгоритма написать текст программы на языке программирования, заданном преподавателем. По тексту программы рассчитать метрики Холстеда и метрики сложности потока данных (спен и метрику Чепина).

4.2. Методические указания к выполнению лабораторной работы

Для разработанной (в соответствии с индивидуальным заданием) в лабораторной работе №1 схемы алгоритма следует написать исходный текст программы на заданном языке программирования. В программе предусмотреть вывод на экран всех входных и выходных данных. Программа должна быть хорошо прокомментирована.

Библиотека БГУИР

Программа должна быть описана. В описании должно быть приведено назначение входных, выходных и внутренних переменных, назначение основных блоков программы.

На основании разработанного исходного текста программы рассчитываются значения метрик Холстеда (см. подразд. 2.1):

– шесть базовых метрик Холстеда (результаты должны быть сведены в таблицу, аналогичную табл. 2.3, 2.4);

– словарь программы;

– длина программы;

– объем программы.

При анализе исходного текста программы следует руководствоваться табл. 2.1, 2.2, представленными в теоретических сведениях к лабораторным работам.

На основании разработанного исходного текста программы рассчитываются значения метрик сложности потока данных (см. подразд. 2.3):

– спены идентификаторов и суммарный спен программы (результаты должны быть сведены в таблицу, аналогичную табл. 2.10);

– полная метрика Чепина и метрика Чепина ввода/вывода (результаты должны быть сведены в таблицу, аналогичную табл. 2.11, с соответствующими пояснениями по распределению переменных по группам).

4.3. Содержание отчета по лабораторной работе

Лабораторная работа №2 должна содержать:

– титульный лист;

– номер и условие индивидуального задания;

– исходный текст программы на заданном преподавателем языке программирования с соответствующими комментариями;

– копию экрана с результатами работы программы;

– описание программы;

– расчет метрик Холстеда для разработанной программы с результатами, представленными в виде таблицы;

– расчет спена разработанной программы с результатами, представленными в виде таблицы;

– расчет полной метрики Чепина и метрики Чепина ввода/вывода с результатами, представленными в виде таблицы.

5. ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ №1 И №2

1. Из последовательности вводимых чисел выбрать отрицательные четные числа. Их значения поместить в массив ***B*(30)**. Остаток массива ***B*** заполнить нулями. Вывести исходные числа и массив ***B*.**

2. Ввести массив ***A*(10, 10)**. Вычислить элементы массива ***Y*** по формуле

Библиотека БГУИР

$$Y_{ik} = \begin{cases} 2 * A_{ik}, \text{ если } A_{ik} < 0; \\ 0, \text{ если } A_{ik} = 0; \\ -A_{ik}, \text{ если } A_{ik} > 0. \end{cases}$$

Вывести исходный и результирующий массивы и количество элементов массива ***Y***, получивших значение 0.

3. Ввести массив ***A*(5, 7)**. Найти сумму элементов каждой строки, максимальную и минимальную из этих сумм. Вывести массив, полученные суммы, номера строк, где находятся максимальная и минимальная суммы.

4. Ввести массив ***A*(6, 7)**. Вычислить элементы массива ***X*** по формуле

$$X_{ij} = \begin{cases} A_{ij} * \cos X, \text{ если } A_{ij} > 0; \\ 0, \text{ если } A_{ij} = 0; \\ -1, \text{ если } A_{ij} < 0. \end{cases}$$

Вывести исходный и результирующий массивы и количество элементов массива ***Y***, получивших значение –1.

5. Ввести массивы ***A*(20)** и ***B*(20)**. Образовать массив ***C*(20)** из элементов, встречающихся в обоих массивах. Остаток массива ***C*** заполнить нулями. Вывести исходные и результирующий массивы.

6. Ввести массивы ***A*(8)** и ***B*(8)**. Получить массив ***C*(8)**, элементы которого получают значения по правилу

$$C_i = \max (A_i, B_i),$$

и подсчитать, сколько элементов C_i получило значение A_i. Вывести исходные массивы и результаты вычислений.

7. Ввести массив ***A*(10)**, первые девять элементов которого упорядочены по возрастанию. Поместить последний элемент массива в соответствующее место массива, чтобы не нарушить его упорядоченность. Вывести исходный и результирующий массивы и номер помещенного элемента.

8. Для значений ***X***, изменяющихся от –1 до 2 с шагом 0,1, вычислить значения функции ***Y***:

$$Y = \begin{cases} \sin X * \cos X, \text{ если } 1 > X >= -1; \\ X^2 * \sin X, \text{ если } 1{,}5 > X >= 1; \\ 0, \text{ если } 2 >= X >= 1{,}5. \end{cases}$$

Полученные значения ***Y*** занести в массив. Вывести значения ***X*** и соответствующие им значения элементов результирующего массива.

9. Ввести массив из 40 элементов. Посчитать количество положительных элементов в массиве. Сформировать массив, у которого первым элементом будет последний (40) элемент исходного массива. Вторым – предпоследний (39) элемент и т. д. (т. е. расположить элементы в обратном порядке). Новый массив сформиро-

Библиотека БГУИР

вать на месте исходного. Вспомогательный массив не использовать. Вывести исходный и результирующий массивы и количество положительных элементов.

10. Ввести массив ***A*(10, 10)**. Найти максимальный элемент в главной диагонали и минимальный элемент в побочной диагонали массива ***A*** и поменять данные элементы местами. Вывести найденные максимальный и минимальный элементы, исходные номера строк и столбцов, в которых они находились, исходный и результирующий массивы.

11. Для значений ***X***, изменяющихся от –2 до 2 с шагом 0,2, вычислить значения функции ***Y***:

$$Y = \begin{cases} e^X + 1, \text{ если } 2 >= X > 1; \\ \ln X, \text{ если } 1 >= X > 0; \\ \sin X + X * \cos X, \text{ если } 0 >= X >= -2. \end{cases}$$

Полученные значения ***Y*** занести в массив. Вывести значения ***X*** и соответствующие им значения элементов результирующего массива.

12. Ввести массив ***A*(100)**. Найти в нем первые десять элементов, которые больше 15. Вывести их значения и их номера в исходном массиве. Если их меньше десяти, вывести текст «Таких значений больше нет». Если их нет вообще, вывести текст «Таких значений нет».

13. Ввести массив ***A*(8, 8)**. Если минимальный элемент данного массива отрицателен, поменять местами главную и побочную диагонали массива. Вывести минимальный элемент, номера строки и столбца, в которых он находится, исходный и результирующий массивы.

14. Для значений ***X***, изменяющихся от –1 до 3 с шагом 0,25, вычислить значения функции ***Y***:

$$Y = \begin{cases} e^X - 1, \text{ если } 3 >= X > 1; \\ \sin^2 X, \text{ если } 1 >= X > 0{,}5; \\ 1, \text{ если } 0{,}5 >= X >= -1. \end{cases}$$

Полученные значения ***Y*** занести в массив. Вывести значения ***X*** и соответствующие им значения элементов результирующего массива.

15. Ввести матрицу ***A*(10, 10)**. Найти максимальные элементы в строках и максимальный элемент матрицы. Вывести исходную матрицу, найденные значения элементов и номера строк и столбцов, где они находятся.

16. Ввести массив ***A*(8, 8)**. Найти минимальный элемент в побочной диагонали массива ***A***. Поменять местами столбец, в котором находится данный элемент, с первым столбцом массива ***A***. Вывести минимальный элемент, номера строки и столбца, в которых он находится, исходный и результирующий массивы.

17. Для значений ***X***, изменяющихся от –1,5 до 2 с шагом 0,01, вычислить значения функции ***Y***:

Библиотека БГУИР

$$Y = \begin{cases} \sin X + \cos X + X, \text{ если } 1 > X >= -1{,}5; \\ X^2 * \cos X, \text{ если } 1{,}5 > X >= 1; \\ \sin X + X^2, \text{ если } 2 >= X >= 1{,}5. \end{cases}$$

Полученные значения ***Y*** занести в массив. Вывести значения ***X*** и соответствующие им значения элементов результирующего массива.

18. Ввести массив ***A*(10, 10)**. Заменить элементы массива, находящиеся на пересечении строк и столбцов с четными номерами, максимальным элементом данного массива. Вывести максимальный элемент, номера строки и столбца, в которых он находится в исходном массиве, исходный и результирующий массивы.

19. Вычислить элементы массива ***S*(10)** по формуле:

$$S_n = \max (A_n, B_n),$$

если

$$A_n = 3n^2 - 10n + 6;$$
$$B_n = 2n + 1;$$
$$n = 1, 2, \ldots, 10.$$

Вывести значения n, A_n, B_n, S_n.

20. Ввести массив ***A*(7, 5)**. Сформировать одномерный массив ***B*(35)** из четных положительных элементов массива ***A***. Остаток массива ***B*** заполнить нулями. Вывести массивы ***A*** и ***B***.

21. Ввести массивы ***A*(8)** и ***B*(8)**. Получить массив ***C*(8)**, элементы которого формируются по правилу

$$C_i = \min (4*A_i, B_i^2),$$

и подсчитать, сколько элементов C_i получило значение B_i^2. Вывести значения массивов ***A***, ***B***, ***C*** и полученное количество элементов.

22. Ввести массивы ***A*(4, 5)** и ***B*(5, 7)**. Поменять местами строку массива ***A***, содержащую максимальный элемент данного массива, и столбец массива ***B***, содержащий минимальный элемент массива ***B***. Вывести максимальный и минимальный элементы, номера строк и столбцов, в которых они находятся, исходные и результирующие массивы.

23. Ввести массивы ***A*(8)** и ***B*(8)**. Вычислить

$$C = \sum_{j=1}^{8} (A_j / B_j)^2$$

для пар A_j и B_j, удовлетворяющих условию $A_j > B_j$. Вывести ***A***, ***B***, ***C*** и номера элементов массивов, участвующих в вычислениях ***C***.

24. Ввести массив ***A*(7, 7)**. Найти максимальный и минимальный элементы

Библиотека БГУИР

в побочной диагонали и поменять местами столбцы массива, в которых они находятся. Вывести максимальный и минимальный элементы, номера столбцов, в которых они находятся в исходном массиве, исходный и результирующий массивы.

25. Ввести массивы ***X*(6)** и ***Y*(6)**. В массиве ***X*** заменить значения тех элементов X_i, для которых выполняется условие

$$|X_i - Y_i| <= 10,$$

значениями элементов Y_i. Вывести исходные и результирующий массивы.

26. Ввести массивы ***A*(5, 7)** и ***B*(3, 6)**. Если максимальный элемент массива ***A*** больше минимального элемента массива ***B***, поменять данные элементы местами. Вывести максимальный и минимальный элементы, номера строк и столбцов, в которых они находятся в исходном массиве, исходные и результирующие массивы.

27. Ввести массив ***A*(7, 8)**. Найти минимальные элементы в столбцах и минимальный элемент массива. Вывести исходный массив, найденные значения минимальных элементов и номера строк и столбцов, где они находятся.

28. Ввести массивы ***A*(10)** и ***B*(10)**. Получить массив ***C*(10)**, элементы которого получают значения по правилу

$$C_j = \begin{cases} B_i / A_i, \text{ если } A_i < B_i; \\ A_i / B_i, \text{ если } A_i > B_i; \\ 0, \text{ если } A_i = B_i. \end{cases}$$

Подсчитать, сколько элементов массива ***C*** получило значение 0. Вывести исходные массивы и результаты вычислений.

29. Ввести массив ***A*(6, 6)**. Найти максимальный и минимальный элементы в главной диагонали и поменять местами строки массива, в которых они находятся. Вывести максимальный и минимальный элементы, номера строк и столбцов, в которых они находятся, исходный и результирующий массивы.

30. Ввести массив ***A*(7, 8)**. Найти сумму элементов каждого столбца, максимальную и минимальную из этих сумм. Вывести массив, полученные суммы, номера столбцов, где находятся максимальная и минимальная суммы.

Библиотека БГУИР

ЛИТЕРАТУРА

1. ГОСТ 19.701–90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – Введ. 1992-01-01. – М. : Изд-во стандартов, 1991.

2. ГОСТ 28195–99. Оценка качества программных средств. Общие положения. – Введ. 2000-03-01. – Минск : Межгосударственный совет по стандартизации, метрологии и сертификации, 2001.

3. ГОСТ 28806–90. Качество программных средств. Термины и определения. – Введ. 1992-01-01. – М. : Изд-во стандартов, 1991.

4. СТБ ИСО/МЭК 9126-2003. Информационные технологии. Оценка программной продукции. Характеристики качества и руководства по их применению. – Введ. 2003-11-01. – Минск : Госстандарт Респ. Беларусь, 2003.

5. СТБ ISO/IEC 25000-2009. Разработка программного обеспечения. Требования к качеству и оценка программного продукта (SQuaRE). Руководство по SQuaRE. – Введ. 2010-01-01. – Минск : Госстандарт Респ. Беларусь, 2009.

6. СТБ ISO/IEC 25001-2009. Разработка программного обеспечения. Требования к качеству и оценка программного продукта (SQuaRE). Планирование и управление. – Введ. 2010-01-01. – Минск : Госстандарт Респ. Беларусь, 2009.

7. ISO/IEC 14598–1:1999. Информационная технология. Оценка программного продукта. – Ч. 1: Общий обзор. – Введ. 1999-04-15. – Женева : ISO/IEC, 2001.

8. ISO/IEC 14598–2:2000. Программная инженерия. Оценка продукта. – Ч. 2: Планирование и управление. – Введ. 2000-02-01. – Женева : ISO/IEC, 2000.

9. ISO/IEC 14598–3:2000. Программная инженерия. Оценка продукта. – Ч. 3: Процесс для разработчиков. – Введ. 2000-02-01. – Женева : ISO/IEC, 2000.

10. ISO/IEC 14598–4:1999. Программная инженерия. Оценка продукта. – Ч. 4: Процесс для заказчиков. – Введ. 1999-10-01. – Женева : ISO/IEC, 1999.

11. ISO/IEC 14598–5:1998. Информационная технология. Оценка программного продукта. – Ч. 5: Процесс для оценщиков. – Введ. 1998-07-01. – Женева : ISO/IEC, 1998.

12. ISO/IEC 14598–6:2001. Программная инженерия. Оценка продукта. – Ч. 6: Документация модулей оценки. – Введ. 2001-06-01. – Женева : ISO/IEC, 2001.

13. ISO/IEC 15939:2007. Программная инженерия. Процесс измерения программных средств. – Введ. 2008-10-01. – Женева : ISO/IEC, 2007.

14. ISO/IEC 25000:2005. Программная инженерия. Требования к качеству и оценка программного продукта (SQuaRE). Руководство по SQuaRE. – Введ. 2005-08-01. – Женева : ISO/IEC, 2005.

15. ISO/IEC 25001:2007. Программная инженерия. Требования к качеству и оценка программного продукта (SQuaRE). Планирование и управление. – Введ. 2007-02-01. – Женева : ISO/IEC, 2007.

16. ISO/IEC 25010:2011. Системная и программная инженерия. Требования к качеству и оценка программного продукта (SQuaRE). Модели качества

Библиотека БГУИР

систем и программных средств. – Введ. 2011-03-01. – Женева : ISO/IEC, 2011.

17. ISO/IEC 25012:2008. Программная инженерия. Требования к качеству и оценка программного продукта (SQuaRE). Модель качества данных. – Введ. 2008-12-15. – Женева : ISO/IEC, 2008.

18. ISO/IEC 25020:2007. Программная инженерия. Требования к качеству и оценка программного продукта (SQuaRE). Эталонная модель измерений. – Введ. 2007-05-15. – Женева : ISO/IEC, 2007.

19. ISO/IEC 25021:2012. Системная и программная инженерия. Требования к качеству и оценка программного продукта (SQuaRE). Элементы мер качества. – Введ. 2012-11-01. – Женева : ISO/IEC, 2012.

20. ISO/IEC 9126:1991. Информационная технология. Оценка программного продукта. Характеристики качества и руководства по их применению.

21. ISO/IEC 9126–1:2001. Программная инженерия. Качество продукта. – Ч. 1: Модель качества. – Введ. 2001-06-15. – Женева : ISO/IEC, 2001.

22. ISO/IEC TR 9126–2:2003. Программная инженерия. Качество продукта. – Ч. 2: Внешние метрики. – Введ. 2003-07-01. – Женева : ISO/IEC, 2003.

23. ISO/IEC TR 9126–3:2003. Программная инженерия. Качество продукта. – Ч. 3: Внутренние метрики. – Введ. 2003-07-01. – Женева : ISO/IEC, 2003.

24. ISO/IEC TR 9126–4:2004. Программная инженерия. Качество продукта. – Ч. 4: Метрики качества в использовании. – Введ. 2004-04-01. – Женева : ISO/IEC, 2004.

25. Бахтизин, В. В. Стандартизация и сертификация программного обеспечения: учеб. пособие / В. В. Бахтизин, Л. А. Глухова. – Минск : БГУИР, 2006.

26. Благодатских, В. А. Стандартизация разработки программных средств: учеб. пособие / В. А. Благодатских, В. А. Волнин, К. С. Поскакалов. – М. : Финансы и статистика, 2003.

27. Изосимов, А. В. Метрическая оценка качества программ / А. В. Изосимов, А. Л. Рыжко. – М. : Изд-во МАИ, 1989.

28. Холстед, М. Х. Начала науки о программах / М. Х. Холстед. – М. : Финансы и статистика, 1981.

29. Черников, Б. В. Управление качеством программного обеспечения: учебник / Б. В. Черников. – М. : ИД «Форум»: ИНФРА-М, 2012.

Библиотека БГУИР

Св. план 2013, поз. 32

Учебное издание

Бахтизин Вячеслав Вениаминович
Глухова Лилия Александровна
Неборский Сергей Николаевич

МЕТРОЛОГИЯ, СТАНДАРТИЗАЦИЯ И СЕРТИФИКАЦИЯ В ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЯХ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *И. В. Ничипор*
Корректор *Е. И. Герман*
Компьютерная правка, оригинал-макет *Е. Г. Бабичева*

Подписано в печать 31.10.2013. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 3,72. Уч.-изд. л. 3,3. Тираж 100 экз. Заказ 142.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6

Библиотека БГУИР