ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г. DOI 10.17587/issn.2220-3397 ISSN 2220-3397

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индекс по Объединенному каталогу "Пресса России" — **22765**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
Журнал включен в систему Российского индекса научного цитирования и базу данных RSCI на платформе Web of Science.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 10
N 3
2019

Published since September 2010 **DOI 10.17587/issn.2220-3397** **ISSN 2220-3397**

CONTENTS

Information about the jornal is available online at:
http://novtex.ru/prin/eng e-mail:prin@novtex.ru

УДК 004.451.9; 004.056.53 DOI: 10.17587/prin.10.99-104

А. И. Грюнталь, зав. отделом, grntl@niisi.ras.ru, **С. Г. Дышленко,** зав. сектором, dishlenko@niisi.ras.ru, Федеральное государственное учреждение "Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук" (ФГУ ФНЦ НИИСИ РАН), г. Москва

Разграничение доступа в автоматизированных системах, функционирующих под управлением операционных систем семейства "Багет"

Описан метод создания автоматизированных систем, функционирующих в среде операционных систем семейства "Багет" и обеспечивающих разграничение доступа и защиту ресурсов этих систем от несанкционированного доступа.

Ключевые слова: *автоматизированная система, несанкционированный доступ, принцип предопределенности доступа, вычислительный домен, интерактивный домен, операционная система реального времени*

Введение

В ФГУ ФНЦ НИИСИ РАН разработано семейство операционных систем реального времени "Багет", предназначенных для разработки и эксплуатации вычислительных систем, функционирующих в реальном масштабе времени [1, 2]. Операционные системы реального времени семейства "Багет" (ОС РВ) не поддерживают понятия пользователя (субъекта доступа). Таким образом, в ОС РВ не реализованы правила разграничения доступа. Однако в автоматизированных системах (АС) на базе ОС РВ может возникнуть необходимость в обеспечении многопользовательского режима, при котором с одной АС работают несколько операторов. При этом требуется обеспечить разделение ресурсов между этими операторами. Соответственно возникает задача защиты от несанкционированного доступа (НСД) [3], состоящая в том, чтобы доступ каждого оператора (субъекта доступа) был бы возможен к тем и только к тем ресурсам, которые определены для данного оператора правилами разграничения доступа (ПРД).

Для таких АС понятие субъекта доступа должно быть введено и поддержано средствами специального программного обеспечения (СПО), реализующими прикладные функции АС. Требование поддержки средств разграничения доступа в этом случае также рассматривается как прикладной аспект функционирования автоматизированной системы.

В настоящей работе представлены концептуальные положения, принятые при создании автоматизированных систем под управлением ОС семейства "Багет", обеспечивающих разграничение доступа к ресурсам таких систем и их защиту от НСД. В основе механизма защиты от НСД лежит принцип предопределенности доступа, определяющий заданный при проектировании АС порядок интерактивного доступа субъектов к ресурсам СПО и операционной системы.

Предопределенность доступа субъекта к ресурсам обеспечивается конфигурацией ОС РВ, исключающей инициализацию встроенных в ОС средств интерактивного взаимодействия, структурой СПО, а также аппаратной конфигурацией АС, которая учитывается при проектировании СПО.

Кросс-технология разработки СПО

Разработка СПО для автоматизированных систем, представляющих собой ЭВМ (или несколько ЭВМ, объединенных сетью), функционирующую под управлением операционной системы семейства "Багет", осуществляется с помощью технологии, которая именуется кросс-технологией.

Суть кросс-технологии заключается в том, что разработка и отладка СПО осуществляется на инструментальной ЭВМ, а исполнение СПО совместно с ОС РВ — на другой ЭВМ (целевой), которая функционирует в составе объекта эксплуатации. При этом инструментальные средства устанавливаются на инструментальную ЭВМ. На такую ЭВМ устанавливаются (записываются в требуемые каталоги) также необходимые для исполнения СПО модули ОС РВ. Целевая ЭВМ при этом не содержит инструментальных средств разработки СПО.

При разработке СПО целевая ЭВМ находится в технологическом режиме эксплуатации, обеспечивающем сетевое взаимодействие с инструментальной ЭВМ. В технологическом режиме готовые к исполнению модули СПО вместе с модулями ОС РВ,

также размещенными на инструментальной ЭВМ, собираются в единую программу, которая именуется "исполняемый образ операционной системы". Исполняемый образ представляет собой предназначенную для функционирования на целевой ЭВМ программу в загрузочном виде.

В технологическом режиме исполняемый образ переписывается через сеть в оперативную память целевой ЭВМ и затем там исполняется. За исполнением можно наблюдать с инструментальной ЭВМ с использованием отладчика или других инструментальных средств.

Для штатной эксплуатации готовый исполняемый образ записывается в энергонезависимую память целевой ЭВМ, и затем целевая ЭВМ аппаратными средствами переводится в штатный режим эксплуатации. При включении питания целевой ЭВМ в штатном режиме исполняемый образ переписывается из энергонезависимой памяти в оперативную память и затем получает управление. Запись исполняемого образа в оперативную память целевой ЭВМ и передача управления исполняемому образу осуществляются программой, резидентно (на постоянной основе) записанной в целевую ЭВМ. Эта программа, называемая программой ПЗУ, представляет собой аналог программ BIOS и UEFFI [4].

В штатном режиме после передачи управления исполняемому образу целевая ЭВМ начинает функционировать в соответствии с алгоритмом, заложенным в специальном программном обеспечении. Инструментальная ЭВМ при штатной работе целевой ЭВМ не используется. Таким образом, кросс-технология разработки СПО гарантирует отсутствие инструментальных средств на целевой ЭВМ.

Принцип предопределенности доступа

Взаимодействие с оператором вычислительной системы, состоящей из целевой ЭВМ, данных и исполняемого образа, возможно с использованием средств операционной системы или заложенных в СПО средств.

Для взаимодействия с оператором в состав ОС РВ входит программа, которая называется "командный интерпретатор". Командный интерпретатор обеспечивает ввод данных с консоли или вывод данных на консоль. Других средств ввода-вывода данных для взаимодействия с оператором ОС РВ не предоставляет.

Взаимодействие с оператором может быть локальным, через локальный терминал, или удаленным, через подключаемый по сети удаленный терминал. Командный интерпретатор предназначен для отладки СПО и должен отключаться от ОС РВ при работе в штатном режиме. Подключение и отключение командного интерпретатора обеспечивается средствами конфигурирования ОС РВ. Конфигурирование — это однократно выполняемая технологическая процедура, исполняемая перед сборкой исполняемого образа. Она выполняется разработчиком СПО и определяет, какие ресурсы СПО включаются в исполняемый образ. Процедура конфигурирования проводится на целевой ЭВМ при завершении разработки и отладки СПО.

При отключенном командном интерпретаторе взаимодействие оператора и вычислительной системы, состоящей из целевой ЭВМ и исполняемого образа, возможно только в том объеме и по тем правилам, которые определяются алгоритмом работы СПО.

Отметим, что в штатном режиме эксплуатации программное обеспечение целевой ЭВМ включает исполняемый образ и программу ПЗУ, назначение которой — загрузка исполняемого образа в оперативную память целевой ЭВМ и передача управления исполняемому образу.

Программное обеспечение целевой ЭВМ, находящейся в штатном режиме эксплуатации, инструментальных средств не содержит. Поэтому это программное обеспечение не имеет средств, использование которых может привести к изменению состава программного обеспечения. Таким образом, любое интерактивное взаимодействие оператора с операционной системой и с ресурсами, доступ к которым обеспечивает операционная система, потенциально возможно только через вызовы функций СПО. Такой подход означает, что в штатном режиме эксплуатации СПО представляет собой экранирующую прослойку, посредством которой осуществляется взаимодействие между оператором и ОС РВ и вычислительной системой в целом. Взаимодействие оператора с ресурсами автоматизированной системы изображено на рис. 1.

Функционирование исполняемого образа происходит по алгоритму, заложенному в СПО. Операционной системе отводится пассивная роль пула вычислительных ресурсов, которые выделяются в том порядке и в том объеме, который заложен в СПО. Это означает, что находящаяся в памяти ЭВМ программа может быть запущена только в том случае, когда эта программа входит в состав исполняемого образа, а алгоритм работы СПО предусматривает ее запуск.

Описанное выше свойство СПО, состоящее в том, что, во-первых, доступ к ресурсам вычислительной системы экранируется и контролируется СПО, а во-вторых, гарантировано отсутствует возможность запуска программ, не входящих в состав СПО, называется принципом предопределенного доступа.

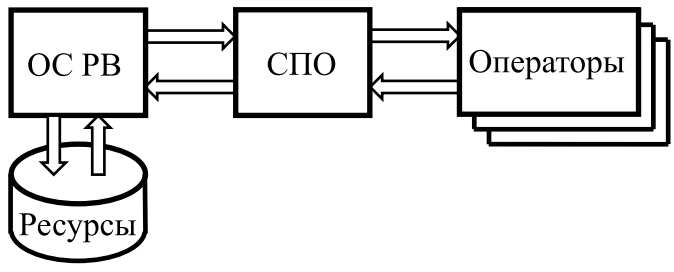

Рис. 1. Взаимодействие оператора и вычислительной системы

Локально ограниченные автоматизированные системы

Операционные системы реального времени семейства "Багет" предназначены в основном для автоматических систем, в которых участие оператора не предусмотрено. В связи с отсутствием субъекта доступа задача по разграничению доступа и защите от НСД для автоматических систем не ставится. Однако операционные системы семейства "Багет" могут также применяться в приложениях, требующих взаимодействия с оператором.

Опишем методы использования ОС РВ в автоматизированных системах, обеспечивающие разграничение доступа и защиту от НСД. Для этого введем классификацию автоматизированных систем. Классифицирующими признаками являются: число операторов, работающих с вычислительной системой, число процессоров, входящих в состав АС, и топология сети, обеспечивающей межпроцессорное взаимодействие.

Определим понятие вычислительного домена. Вычислительный домен представляет собой однопроцессорную ЭВМ с локальной, доступной только для этого процессора оперативной памятью. Домен должен функционировать под управлением ОС РВ. При этом должен быть установлен штатный режим его функционирования. Несколько вычислительных доменов могут быть объединены в вычислительную сеть. Автоматизированную систему, содержащую несколько доменов, будем называть многодоменной. Автоматизированную систему будем называть локально ограниченной, если образующие АС домены не имеют сетевых соединений с другими ЭВМ, внешними по отношению к автоматизированной системе.

Домены, образующие многодоменную АС, физически могут находиться в одном корпусе ЭВМ или в различных корпусах, оставаясь при этом соединенными в одну сеть. С точки зрения топологии сетевых соединений многопроцессорная ЭВМ, включающая в свой состав несколько доменов, не отличается от АС, представляющей собой несколько соединенных сетью ЭВМ. Поэтому не будем различать между собой локально ограниченные многодоменные АС, физически содержащие одну или несколько ЭВМ.

Старт многодоменной системы выполняется в два этапа. На первом этапе происходит старт каждого из доменов автоматизированной системы. Начинается сеанс установки сетевого взаимодействия между доменами. После того как сетевое взаимодействие между доменами установлено, начинается совместное функционирование доменов. После включения питания многодоменная АС начинает функционировать как несколько однопроцессорных ЭВМ, осуществляющих обмен данными по сети.

Вычислительный домен может содержать средства интерактивного графического взаимодействия. Такой домен будем называть интерактивным. Многодоменные АС могут содержать несколько интерактивных доменов.

Задача разграничения доступа и защиты от НСД применима и к АС (однодоменным или многодоменным), содержащим только один интерактивный домен. В этом случае разграничение доступа должно обеспечиваться для операторов, последовательно работающих за одним рабочим местом (на общем интерактивном домене).

Также будем считать, что локально ограниченные АС обладают постоянством аппаратного состава. Аппаратный состав таких АС задается на этапе проектирования, остается постоянным на все время эксплуатации. Постоянство состава аппаратуры АС должно гарантироваться организационно-техническими мерами.

Далее будем рассматривать локально ограниченные автоматизированные системы с постоянным аппаратным составом. Для таких систем приведем методы использования ОС РВ, обеспечивающие разграничение доступа и защиту от НСД.

Разграничение ресурсов в локально ограниченных АС

Задача разграничения ресурсов возникает в случае многопользовательского доступа, когда работу с АС осуществляют несколько операторов (далее — субъектов). Как отмечалось выше, доступ может быть многопользовательским, и в случае наличия в составе АС одного интерактивного домена различные субъекты могут работать последовательно, используя один общий интерактивный домен.

Для организации многопользовательского режима доступа и защиты от НСД в АС должны быть реализованы механизмы, обеспечивающие:

а) ведение базы данных ресурсов (объектов), доступ к которым ограничен;

б) ведение базы данных субъектов, осуществляющих доступ к объектам;

в) ведение базы данных, содержащей правила разграничения доступа (ПРД);

г) реализацию ПРД;

д) целостность указанных баз данных и программ, реализующих ПРД.

Кроме того, механизмы организации многопользовательского режима доступа должны обеспечивать отсутствие возможности (далее для краткости — невозможность) обхода правил разграничения доступа.

Многопользовательский доступ в однодоменных АС

В качестве модельной задачи опишем архитектуру СПО, обеспечивающую разграничение доступа для однодоменных АС, в которых этот один домен интерактивный. Указанная архитектура будет основой для многодоменных АС с разграничением доступа с несколькими интерактивными доменами. Программное обеспечение однодоменной АС состоит из операционной системы и СПО, которое содержит программы, реализующие прикладные функции.

Как было отмечено выше, в штатном режиме эксплуатации для однодоменных автоматизированных систем выполняется свойство предопределенности доступа. При наличии интерактивного домена предопределенность доступа означает, что любой запрос оператора к ресурсам АС выполняется путем вызова соответствующих функций СПО. Эти функции обеспечивают организацию интерактивного диалога с оператором, вызов и исполнение прикладных программ, входящих в состав СПО, и при необходимости обращение к операционной системе. В силу предопределенности доступа доступ к ресурсам, в частности, к программам осуществляется по правилам, заложенным в СПО на этапе проектирования и реализованным в виде программного кода на этапе разработки. Доступ к вычислительным ресурсам помимо СПО, например, путем непосредственного обращения к операционной системе, невозможен. Это обстоятельство позволяет встроить в СПО механизмы организации многопользовательского режима доступа, обеспечивающие правила разграничения доступа и защиту от НСД.

Для решения отмеченных выше задач СПО должно содержать базы данных ресурсов, субъектов, правил разграничения доступа. Такое СПО должно обеспечивать целостность этих объектов, целостность программ, реализующих ведение баз данных, доступ к объектам согласно ПРД и невозможность обхода ПРД. Таким образом, в СПО должен входить программный модуль, обеспечивающий реализацию указанных выше функций. Назовем его модулем обеспечения ПРД. Однако отметим, что этот модуль отвечает не только за реализацию, но и за весь блок задач, связанных с реализацией многопользовательского доступа и защитой от НСД.

Архитектурно возможны различные реализации модуля обеспечения ПРД. Рассмотрим простейшую модельную схему — схему с разделением пользовательских сценариев.

Пользовательский сценарий представляет собой часть кода СПО, который обеспечивает доступ к предопределенному на этапе проектирования набору ресурсов АС, включая данные, программы, реализующие заданные функции, и ресурсы операционной системы.

В начале сессии выполняется процедура идентификации/аутентификации. Выполнение этой процедуры обеспечивается средствами СПО путем обращения к базе данных субъектов, в которой наряду с идентификационными признаками субъектов хранятся метки пользовательских сценариев. В результате выполнения процедуры идентификации/аутентификации определяется метка сценария. По метке определяется сценарий, а затем этот сценарий начинает выполняться. Каждый сценарий выполнения СПО определяет однозначный (для данного сценария) доступ к файлам и программам.

Ведение баз данных, регламентирующих порядок выбора сценариев, осуществляется субъектом с правами администратора АС. Администрирование также выполняется средствами модуля реализации ПРД.

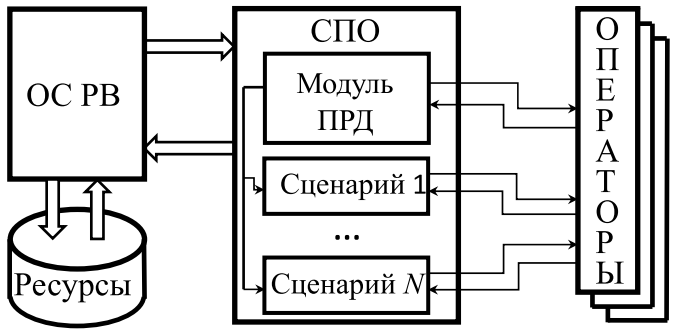

Рис. 2. Схема модуля реализации ПРД с разделением пользовательских сценариев

В связи с тем, что СПО представляет собой экранирующую прослойку между субъектом и операционной системой, целостность данных, обеспечивающих многопользовательский доступ и защиту от НСД, реализуется средствами входящего в состав СПО модуля ПРД.

Схема реализации ПРД с разделением пользовательских сценариев приведена на рис. 2.

Результатом логической прозрачности средств разграничения доступа и защиты от НСД является простота анализа исходного кода в той части, которая относится к защите информации, что повышает надежность средств защиты от НСД.

Многопользовательский доступ в многодоменных АС

Вначале приведем схему организации многопользовательского доступа в локально ограниченной многодоменной АС, содержащей один интерактивный домен. В многодоменной АС различные домены соединены сетью, и субъект имеет сетевой доступ к различным доменам, образующим АС. В связи с тем, что АС является локально ограниченной, она изолирована в сетевом смысле от компьютеров, не входящих в АС, и доступ к ресурсам доменов возможен только с интерактивного домена.

Выполняемое в штатном режиме СПО имеет свойство предопределенности доступа. Поэтому доступ к ресурсам домена возможен только путем обращения субъекта к СПО, выполняющего роль экранирующей прослойки. Правила и программная реализация доступа к ресурсам регламентируются входящим в СПО и функционирующим на интерактивном домене модулем обеспечения ПРД.

К этим ресурсам относятся и запросы на сетевой доступ к другим, отличным от интерактивного доменам АС. В силу постоянства аппаратного состава АС правила доступа к доменам АС могут быть заданы на этапе проектирования и затем реализованы в виде программного кода на этапе генерации кода.

Таким образом, модуль обеспечения ПРД, находящийся на интерактивном домене, должен быть спроектирован так, чтобы регламентировать доступ ко всем доменам локально ограниченной АС в целом.

Правила доступа как к локальным, так и к удаленным ресурсам, находящимся на доменах, отличных от интерактивного, однозначно определяются модулем обеспечения ПРД, входящим в состав СПО интерактивного домена. Правила, локализованные в модуле обеспечения ПРД, должны носить "глобальный характер", т. е. относиться ко всем доменам АС в целом, а не только к интерактивному домену.

Как и в случае однодоменной АС, на интерактивном домене средствами СПО выполняется процедура идентификации/аутентификации. В результате выполнения этой процедуры определяется метка пользовательского сценария. Выбранный сценарий запускается. Запущенный сценарий обеспечивает доступ к предопределенным (для этого сценария) ресурсам АС, как локальным, расположенным на интерактивном домене, так и удаленным, расположенным на других доменах. Невозможность обхода ПРД обеспечивается предопределенностью доступа к данным. Схема реализации ПРД для многодоменных АС приведена на рис. 3.

Опишем схему организации многопользовательского доступа и разграничения доступа для многодоменных АС, содержащих несколько интерактивных доменов. В целом эта схема аналогична схеме случая, когда в АС присутствует только один интерактивный домен. В этом случае каждый интерактивный домен должен содержать программный модуль обеспечения ПРД.

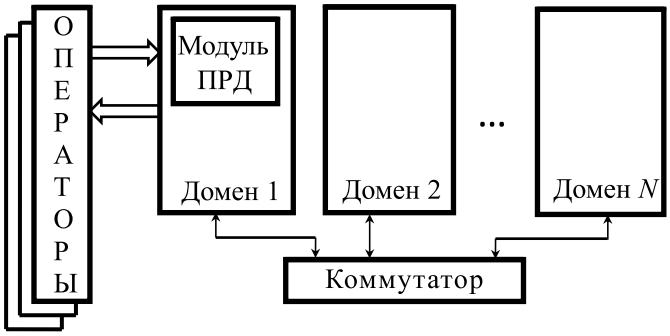

Рис. 3. Схема реализации ПРД для многодоменных АС с одним интерактивным доменом

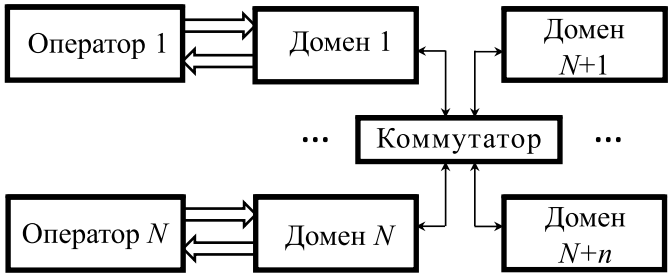

Рис. 4. Схема реализации ПРД для многодоменных АС, содержащих несколько интерактивных доменов

Каждый такой модуль обеспечения ПРД должен включать глобальную модель ресурсов АС в целом и правила разграничения доступа ко всем ресурсам АС. В силу предопределенности доступа к ресурсам домена указанная схема взаимодействия доменов обеспечивает целостность данных и программ, реализующих ПРД.

Схема реализации ПРД для многодоменных АС, содержащих несколько интерактивных доменов, приведена на рис. 4.

Заключение

Представлен метод (метод локализации доступа) создания автоматизированных систем под управлением операционных систем реального времени семейства "Багет". Метод локализации доступа отличается простотой реализации, что повышает надежность механизмов защиты от НСД. В силу логической прозрачности предложенный авторами метод допускает исчерпывающую проверку каждой реализации, что повышает надежность данного метода.

Предложенный метод допускает вариативный подход при реализации. На основе этого метода, например, могут создаваться сети ЭВМ с защитой от НСД, функционирующие в сетях, содержащих ЭВМ, средства защиты которых отсутствуют или не соответствуют локализации доступа. Для защиты от НСД в таких АС необходимо наличие доменов, выполняющих роль межсетевых экранов и отделяющих локально ограниченные сети от потенциально небезопасных фрагментов сети. Рассмотрение соответствующих методов авторы планируют продолжить в дальнейшем.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований по теме (проекту) "39. Архитектура, системные решения, программное обеспечение, стандартизация и информационная безопасность информационно-вычислительных комплексов и сетей новых поколений. Разработка методов и средств контролируемого выполнения приложений, функционирующих в реальном масштабе времени (№ 0065—2018—0011), АААА—А18—118041190171—0".

Список литературы

1. **Годунов А. Н.** Операционная система реального времени Багет 3.0 // Программные продукты и системы. 2010. № 4. С. 15—19.

2. **Годунов А. Н., Солдатов В. А.** Операционные системы семейства Багет (сходства, отличия и перспективы) // Программирование. 2014. № 5. С. 68—76.

3. **ГОСТ** Р 53114—2008. Защита информации. Обеспечение информационной безопасности в организации. М.: Стандартинформ, 2009.

4. **Cooper D., Polk W., Regenscheid A., Souppaya M.** BIOS Protection Guidelines. Recommendations of the National Institute of Standards and Technology, Special Publication 800-147, April 2011.

Access Control in Automated Systems Based on "Baguette" Real-Time Operating Systems

A. I. Gryuntal, grntl@niisi.ras.ru, **S. G. Dyshlenko,** dishlenko@niisi.ras.ru, Federal state institution "Federal scientific center research Institute of system studies of the Russian Academy of Sciences", Moscow, 117218, Russian Federation,

Corresponding author:

Gryuntal Andrej I., Head of Department, Federal state institution "Federal scientific center research Institute of system studies of the Russian Academy of Sciences", Moscow, 117218, Russian Federation,
E-mail: grntl@niisi.ras.ru

Received on November 16, 2018
Accepted on December 11, 2018

The article provides a conceptual basis for automated systems development which exploits the "Baguette" real-time operating systems and provides unauthorized access protection. The "Baguette" operating systems do not include access control means. That is why access control and unauthorized access protection mechanisms are to be implemented using application software.

By means of the "Baguette" operating systems configuration facilities interactive programs contained in the OS could be excluded. The "Baguette" operating systems utilize cross-technology application development scheme. This implies that no unauthorized code could be injected or executed in application hardware environment. It follows that the only way to get access to some application resource is to make use of algorithms included in the application algorithms which determine interactive communication with the operator.

This means that predetermined access principle is fulfilled — all access attempts are under application software control and any access act could not be carried out in case it is not authorized by the application. Hence access control means could be localized in the application software.

This scheme could be used for a system consisting of a single computer with "Baguette" operating system installed, or in case if automated system includes a few computers, which are connected by LAN isolated from other computers. The hardware composition of these automated systems is to be permanent.

Keywords: *automated system, unauthorized access, principle of certainty of access, computing domain, interactive domain, real-time operating system*

For citation:

Gryuntal A. I., Dyshlenko S. G. Access Control in Automated Systems Based on "Baguette" Real-Time Operating Systems, *Programmnaya Ingeneria,* 2019, vol. 10, no. 3, pp. 99—104.

DOI: 10.17587/prin.10.99-104

References

1. **Godunov A. N.** Operacionnaja sistema real'nogo vremeni Baget 3.0 (Real-time operating system Baguette 3.0), *Programmnye produkty i sistemy*, 2010, no. 4, pp. 15—19 (in Russian).

2. **Godunov A. N., Soldatov V. A.** Operacionnye sistemy semejstva Baget (shodstva, otlichija i perspektivy) (Operating systems of the Baguette family (similarities, differences and prospects)), *Programmirovanie,* 2014, no. 5, pp. 68—76 (in Russian).

3. **GOST** R 53114—2008 Obespechenie informacionnoj bezopasnosti v organizacii, Moscow, Standartinform, 2009. (in Russian).

4. **Cooper D., Polk W., Regenscheid A., Souppaya M.** BIOS Protection Guidelines. Recommendations of the National Institute of Standards and Technology, Special Publication 800-147, April 2011.

УДК 004.4'2 DOI: 10.17587/prin.10.105-113

П. А. Ченцов, канд. физ.-мат. наук, науч. сотр., e-mail: chentsov.p@uran.ru,
Институт математики и механики им. Н. Н. Красовского УрО РАН, г. Екатеринбург

Об одном подходе к разработке кроссплатформенных приложений на языке C++

Рассмотрена технология разработки кроссплатформенного программного обеспечения на языке C++. Построение программ осуществляется с использованием специальной библиотеки классов, в которой локализованы основные конструкции, обеспечивающие переносимость кода. Данная библиотека легко адаптируется к разным операционным системам, компиляторам и базовым библиотекам C++. При этом зависимость пользовательского интерфейса программ от системы разработки сохраняется, но предлагается набор приемов, призванных ее минимизировать.

Ключевые слова: *C++, кроссплатформенное программирование*

Введение

Язык C++ востребован для решения целого ряда сложных задач. В силу своей архитектуры C++ позволяет создавать программы, выполняемые значительно быстрее аналогичных, созданных с использованием иных языков программирования (C#, Python, Java и др.). Компиляторы C++ прошли значительную эволюцию в течение десятилетий и позволяют формировать качественный и эффективный код с высокой степенью оптимизации. Перечисленные особенности выходят на передний план, когда требуется разработать вычислительные программы, редакторы видео, трехмерные игры и другие ресурсоемкие приложения.

Существует несколько десятков компиляторов C++ и библиотек кода для различных операционных систем. Данный набор средств (операционная система, компилятор и библиотеки) далее будем называть системой разработки. Учитывая это, исключительно полезной представляется возможность переноса программ между разными системами разработки.

Базовые функциональные возможности языка C++ недостаточны для создания сложных приложений. Поэтому для разработки программ используют различные библиотеки (построение интерфейсов, контейнеры, работа со строками, потоки и т. д.). Наиболее известной является стандартная библиотека C++ [1]. При разработке программы только с использованием стандартной библиотеки появляется возможность компиляции ее в любой системе разработки (с версией стандарта C++ и стандартной библиотекой, не ниже использованных в программе). Однако сложность заключается в том, что стандартная библиотека имеет ограниченные функциональные возможности. Например, в ней отсутствует код для разработки пользовательских интерфейсов и графики. В некотором смысле стандартная библиотека является достаточно громоздкой и сложной для освоения. Данное обстоятельство порой отталкивает некоторых программистов. Таким образом, полностью задачу переносимости стандартная библиотека не решает.

В работе [2] рассмотрены общие приемы, обеспечивающие переносимость программ. Речь идет о приемах, которые позволяют сделать относительно простым перенос программ между различными системами разработки.

Существуют также кроссплатформенные библиотеки, позволяющие разрабатывать программы с графическим интерфейсом и содержащие богатые возможности во многих других аспектах. Такие библиотеки разрабатывают под выбранные операционные системы для выбранных компиляторов. Разработанную с использованием этих библиотек программу можно компилировать на любом из поддерживаемых компиляторов в любой из поддерживаемых операционных систем, причем без изменений в коде. Примерами таких библиотек являются Qt [3], GTK+ [4], Juce [5], wxWidgets [6]. Однако если в основу своей программы положить одну из таких библиотек, то переход на неподдерживаемые разработчиками этой библиотеки компиляторы и операционные системы будет крайне затруднителен. Затруднения будет вызывать и переход на другую подобную библиотеку, так как в основу программы будут положены библиотечные классы. Например, классы работы с контейнерами, строками, временем, потоками, сетью, базами данных и т. д. Каждый класс содержит свой набор методов, свойств и полей. Этот набор отличается от аналогичного по назначению в классе другой библиотеки. Иногда различается и сама логика работы аналогичных классов в разных библиотеках. Таким образом, просто переименовать классы будет недо-

статочно. С учетом отмеченных выше особенностей отрицательной стороной использования упомянутых разработок является полная зависимость программы от выбранной библиотеки. Однозначно сказать, что это плохо, нельзя. Однако каждый разработчик должен понимать, что он вкладывает в программы значительные интеллектуальные и временные ресурсы. Как следствие, если по какой-либо причине ему потребуется сменить систему разработки, то фактически он лишается возможности использовать свои наработки. Кроме того, некоторые полезные программы могли бы найти более широкое применение в случае распространения их на разные платформы.

Резюмируя изложенное выше, можно констатировать, что вопрос переносимости кода C++ к настоящему времени полностью не решен. В статье предлагается нестандартный взгляд на частичное разрешение этого вопроса. В контексте настоящей работы целью является не обеспечение полной переносимости кода, а достижение возможности перехода к другим системам разработки с малыми изменениями в коде. При этом необходимо выполнение условия, что подобные изменения должны быть в максимально возможной степени сосредоточены во внешней, общей для всех программ библиотеке. Такая библиотека должна выступать в качестве своего рода буфера между самой программой и системой разработки.

Постановка задачи

Для обеспечения возможности быстрого переноса программного кода между системами разработки требуется выработать подход, при котором объем системно-зависимого кода будет минимизирован, а сам код локализован. Общий для всех приложений системно-зависимый код, а также код, обеспечивающий реализацию приемов переносимости (например, порядок следования байт в переменных различных типов при сохранении и чтении из потоков), следует вынести в отдельную общую библиотеку. Строгое следование использованию конструкций такой библиотеки автоматически сведет к минимуму усилия программистов по обеспечению возможностей переносимости кода их приложений. Это особенно важно, когда таковых приложений значительное число.

Наиболее распространенные и востребованные программные конструкции следует реализовать в виде отдельных классов. Сюда следует отнести классы контейнеров, классы работы со строками, временем, случайными величинами, потоками ввода-вывода, синхронизацией потоков выполнения и т. д. При этом весь системно-зависимый код должен быть помещен в функции и классы, располагаемые в отдельном изолированном файле (прием изоляции системно-зависимого кода). Причем в отличие от работы [2], где изоляция кода рассматривается как прием разработки конкретных программ, в настоящей работе предлагается выполнить изоляцию системно-зависимого кода в базовой библиотеке. Для функций работы с графикой и пользовательским интерфейсом должны быть созданы аналогичные файлы с системно-зависимым кодом.

В плане пользовательского интерфейса в рамках предлагаемого подхода требуется выделить часто используемые и наиболее стандартные конструкции и, по сути, реализовать для них универсальный программный интерфейс. То же самое касается работы с графикой. Применение таких интерфейсов избавляет разработчика от необходимости реализовывать указанные функции в своих программах. В этом случае нет необходимости использовать какие-либо приемы программирования для адаптации кода к выбранной системе разработки.

В настоящее время все больше компьютеров оснащаются 64-битными операционными системами. Тем не менее 32-битные системы, скорее всего, будут использоваться еще достаточно долго. Таким образом, библиотека должна обеспечивать как 64-битную, так и 32-битную компиляцию без изменений программного кода. Появляется необходимость учесть все проблемные ситуации, которые возникают при переходе на 64-битный режим. Всюду, где осуществляется работа с указателями и адресацией, следует использовать типы данных ptrdiff_t и size_t из стандартной библиотеки.

Для обеспечения независимости целочисленных типов они должны быть переопределены так, чтобы размер и использование знака всюду были едиными. Это не относится к типам данных ptrdiff_t и size_t. При этом следует особое внимание уделить механизмам сохранения и чтения переменных этих типов (ptrdiff_t и size_t) из потоков как на 32-битных, так и 64-битных системах.

Следует также отметить, что совместное использование данной переносимой библиотеки и стандартной библиотеки C++ является вполне допустимым.

Общее описание технологии

Обычно проект C++ включает в себя два основных компонента: саму программу и библиотеки, реализующие те или иные функциональные механизмы, необходимые для работы программы. В свете этого обстоятельства основные проблемные вопросы, связанные с переносимостью, условно можно разделить на три типа.

1. Вопросы, связанные с переходом на младшую версию стандарта C++ или использованных библиотек.

2. Особенности программирования: вопросы, связанные с порядком следования байт при сериализации, различиями в размерах типов данных, использования знака в типе char, интернационализацией и т. д. [2].

3. Вопросы, связанные с использованием библиотек, которые реализованы в конкретных системах разработки (не считая стандартной библиотеки C++).

Бороться с трудностями переносимости можно в каждом отдельно взятом приложении. Однако в та-

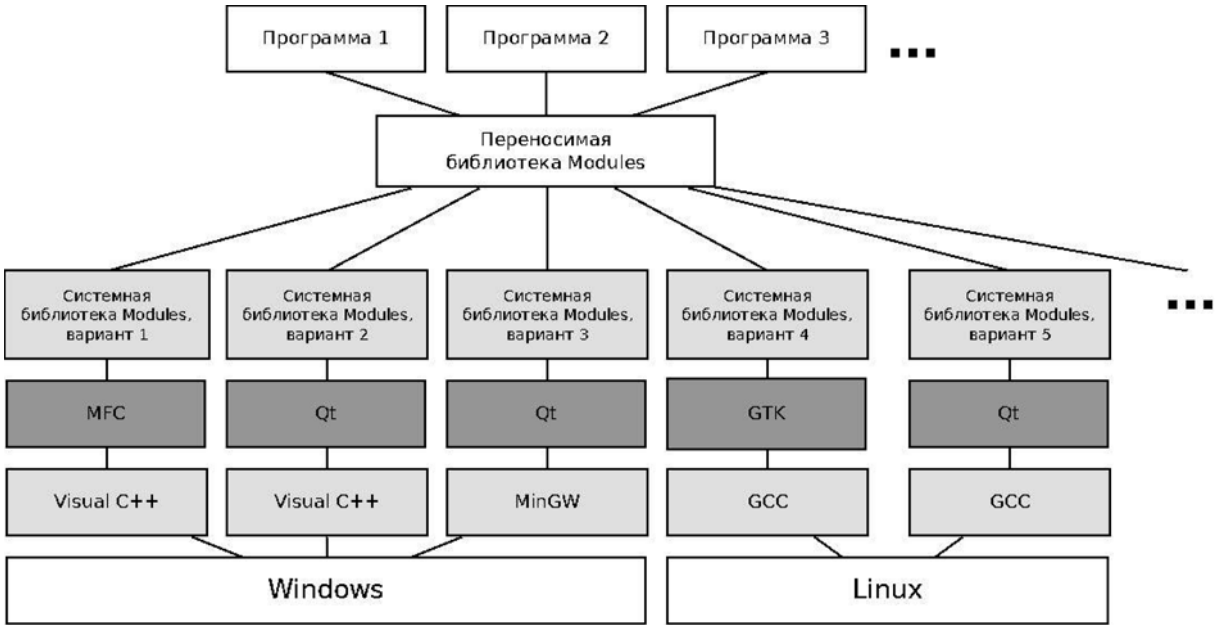

Рис. 1. Схема реализации технологии Modules

ком случае возникнет многократно повторяемый код, и при переходе к новой системе разработки появляется потребность в значительной доработке каждой программы. Таким образом, вполне разумным представляется создание базовой библиотеки, решающей основные вопросы переносимости. Такая библиотека была разработана автором статьи и получила название Modules.

Для устранения вопросов первого типа в рамках рассматриваемого подхода возникает потребность избегать самых новых стандартов C++. Эта рекомендация не означает, что программа будет медленно или некорректно работать. Новые стандарты позволяют использовать новые ключевые слова и конструкции, в некотором смысле облегчающие процесс разработки.

Для решения вопросов второго типа в рамках библиотеки реализованы основные приемы, обеспечивающие их решение и связанные с особенностями программирования. Например, строго определены все основные типы данных. Для каждого типа фиксирован признак знаковости и размер в байтах. В случае работы с однобайтными числовыми переменными вместо типа char предписывается использовать знаковый или беззнаковый однобайтные типы, определенные в библиотеке. Вопросы, связанные с порядком следования байт, решаются в классах потоков ввода-вывода. Для обеспечения интернационализации используется полномасштабная поддержка юникода. Для поддержки работы в 32- и 64-битных системах повсеместно используются типы данных ptrdiff_t и size_t с поддержкой единообразной сериализации значений (архитектуры).

Для решения вопросов третьего типа предлагается использовать прием, названный в работе [2] изоляцией системно-зависимого кода. Создается несколько отдельных программных модулей, в которых реализуются функции и классы, содержащие внутри себя вызовы системных функций. Данную часть библиотеки далее будем называть системной библиотекой Modules. Вторую часть библиотеки, не использующую напрямую системные функции и классы, будем называть переносимой библиотекой Modules.

Переносимая часть библиотеки использует функции и классы из системной. Таким образом, переносимая библиотека сама по себе является полностью переносимой, также как и разработанные на ее основе программы, в которых вся работа происходит именно с ней. Значит, для переноса программы в другую систему разработки достаточно реализовать все функции системной части библиотеки в этой новой системе разработки. Причем сделать это нужно один раз. Далее все программы, разработанные в данной технологии, используют специфичный для новой системы разработки код, и будут компилироваться без ошибок. Общая структура технологии показана на рис. 1.

В рамках данной схемы также может быть реализована, например, работа с графикой и некоторыми конструкциями пользовательского интерфейса, что важно для графических приложений.

Пользовательский интерфейс

Несмотря на широкие возможности, которые открывает предлагаемый подход, построить универсальный пользовательский интерфейс таким способом вряд ли получится. Однако автор и не пытается построить полностью переносимый код, что отме-

чалось ранее. Основная цель в контексте представленного подхода — минимизировать и локализовать системно-зависимый код.

Существуют несколько подходов к организации архитектуры программ с пользовательским интерфейсом. Один из таковых — Document-View [6]. Суть его состоит в том, чтобы разделить данные, механизмы их обработки и отображения. Вне зависимости от используемой системы построения интерфейса, способ взаимодействия с кодом бизнес-логики приложения, как правило, сводится к следующему. Разрабатываются классы, унаследованные от классов графической библиотеки и, соответственно, являющиеся системно-зависимыми. В методах-обработчиках этих новых классов для различных действий (нажатие кнопки мыши, выбор пункта меню и т. д.) выполняется реализация программного кода, связанного с решаемой задачей. Кроме того, в полях этих классов хранятся данные программы. Кроме методов-обработчиков создаются также различные методы, связанные с бизнес-логикой программы. Следует особо отметить, что многие методы с кодом приложения часто переплетены с такими конструкциями интерфейса, как окна сообщений, диалоги получения имени файла или каталога, диалоги работы с параметрами и т. д. Все эти обстоятельства делают программу системно-зависимой.

Для частичного разрешения обозначенных вопросов предлагается непосредственно код приложения разрабатывать в отдельном классе в рамках библиотеки Modules. Основная часть взаимодействия классов интерфейсной библиотеки и классов бизнес-логики должна осуществляться посредством вызовов методов объекта данного класса приложения внутри методов интерфейсных классов. Иными словами, на каждый обработчик события в классе интерфейса будет свой метод класса приложения. Такой подход позволяет всю интерфейсную систему представить как контейнер, а класс приложения — как независимое содержимое.

Необходимо отметить, что программы бывают различными — простыми и сложными с функциональных позиций, с минимальным интерфейсом или интерфейсом с множеством меню, диалогов, панелей. Однако в любом случае переписать сам контейнер проще, чем все приложение. Можно создать меню, в каждом обработчике вызвать соответствующий метод класса приложения. Получается несколько строк на каждый пункт меню. То же самое относится и к реакциям на события от мыши и клавиатуры. Иными словами, во многих случаях этот контейнер может оказаться очень простым и компактным, а его реализация может быть проведена в кратчайшие сроки.

Но реализации предлагаемого подхода препятствуют различные интерфейсные конструкции, которые не могут быть отделены от кода класса приложения. К их числу, например, можно отнести диалоги выбора файла или каталога и окна сообщений. Код метода открытия файла может содержать проверку режима работы, создание и активацию диалога выбора имени файла, проверку формата, реакцию на неверный формат или чтение данных из файла. При классическом подходе весь этот код должен располагаться в обработчике интерфейсного класса, так как в нем используется диалог открытия файла из интерфейсной библиотеки. Для диалога сообщений ситуация может быть даже более критичной, так как различных сообщений пользователю в одной программе могут быть сотни. Причем формируются они не только в самих методах класса приложения, но и в объектах, с которыми работают эти методы. Таким образом, все это противоречит выбранному подходу. Чтобы исправить ситуацию, предлагается предпринять описанные далее действия.

Отмеченные выше интерфейсные диалоги, с одной стороны, имеют однотипный вид, а с другой стороны, часто используются в программах. Следовательно, в системной части библиотеки можно реализовать функции, активирующие диалоги открытия и сохранения файла, а также возвращающие имя файла. Реализация таких функций будет использовать конструкции интерфейсной библиотеки. При этом весь код программы открытия файла, использующий такие функции, можно будет упаковать в метод класса приложения. Программисту, использующему предлагаемый подход в отношении переносимости кода, нет необходимости знать о том, что именно происходит внутри подобной функции (выбор имени файла). Важно, что она возвращает результат выбора файла (подтверждение или отказ выбора файла и имя выбранного файла). То же самое можно реализовать для выбора каталога и для диалога сообщений. Здесь важно отметить, что использование подобных функций также не делает программу (класс приложения) системно-зависимой. Сами реализации функций выбора файла или каталога и вывода сообщения реализуются при этом в системной части библиотеки, причем для каждой системы разработки по-своему.

Подобный подход можно распространить и на другие часто используемые конструкции. К таким конструкциям можно отнести, например, диалоги параметров. Во многих программах появляется необходимость выполнять то или иное конфигурирование или настройку параметров алгоритмов. Обычно в такой ситуации для каждого конкретного случая создается свое диалоговое окно, в котором располагаются элементы ввода — поля ввода, флажки, переключатели и т. д. Перед использованием такого диалога сначала необходимо поместить в его поля (соответствующие элементам ввода) текущие значения параметров, после чего активировать диалог в модальном режиме. После завершения работы диалога в случае, если пользователь подтвердил сохранение значений (обычно кнопкой "OK" или "Сохранить"), их следует переместить в обратную сторону. А именно — из полей диалога в поля параметров класса. Если таких диалогов много и они работают с большим числом параметров, то объем интерфейсного кода, связанного с функцией конфигурирова-

ния, может оказаться значительным, учитывая непосредственно разработку диалогов.

Представляется возможным автоматизировать данный процесс путем сведения к минимуму объема программного кода. Для этого разработан специальный класс CParameters. Его назначение — регистрировать параметры классов и передавать информацию о них в универсальный диалог параметров, который формируется единожды для каждой новой системы разработки в системной части библиотеки. Далее будем именовать указанный диалог CParametersDialog. У пользователя библиотеки Modules не возникает необходимости работать непосредственно с CParametersDialog. Получение этим диалогом значений параметров, как и информации о них самих, а также возврат значений параметров в поля класса осуществляются автоматически. Достаточно вызывать функцию EF_ParametersDialog, в которую передается имя заголовка окна и объект класса CParameters. Внутри функции EF_ParametersDialog осуществляются создание объекта класса CParametersDialog, установка его заголовка и передача ему объекта класса CParameters. Ниже представлен пример использования класса CParameters в виде исходного кода.

```
class CNewClass { public:
t_int32 m_Value1;
t_double m_Value2;
CParameters m_Parameters;
CNewClass() {
m_Value1 = 1;
m_Parameters.LinkTInt32Parameter("Первый
параметр",
"Подробное описание первого параметра",
true, &m_Value1, 0, 100);
m_Value2 = 10.0;
m_Parameters.LinkTDoubleParameter("Второй
параметр",
"Подробное описание второго параметра",
true, &m_Value2, 0.0, 100.0);
}
void EditParameters() {
if (EF_ParametersDialog("Параметры", &m_
Parameters))
{...Действия, выполняемые при изменении
параметров... }
}
};
```

Внутри конструктора класса CNewClass происходит регистрация в объекте параметров (m_Parameters), доступных для редактирования полей класса. В данном случае это делается с использованием двух методов класса CParameters: LinkTInt32Parameter() и LinkTDoubleParameter(). Каждому определенному в рамках библиотеки типу данных ставится в соответствие свой подобный метод регистрации. В такие методы передаются имя и подробное текстовое описание параметра, признак того, будет ли он отображаться в списке (например, если поле используется только в классах-потомках), указатель на поле класса и пределы для значения. В методе EditParameters() указатель на объект параметров вместе с текстом заголовка окна диалога передаются функции EF_ParametersDialog(). Такая функция отмечалась ранее, она непосредственно обеспечивает диалог редактирования параметров на основе информации из объекта m_Parameters.

Внутри функции EF_ParametersDialog() конструируется стандартный диалог (объект класса CParametersDialog). В качестве его параметров конструктору передаются все тот же текст заголовка и указатель на объект параметров. Внутри конструктора диалога происходит построение таблицы параметров на основе информации из указателя на объект параметров. При этом извлекаются все зарегистрированные данные: комментарий; подробное описание; пределы и указатель на поле объекта. Через данный указатель автоматически извлекается значение параметра, помещаемое во вторую колонку таблицы. В случае подтверждения ввода через кнопку "OK" новые значения из таблицы автоматически перемещаются в поля класса с использованием указателей. Нажатие на кнопку "Отмена" приведет к закрытию диалога без изменения значений его параметров. На рис. 2 показана схема организации диалога параметров.

При нажатии левой кнопкой мыши на значение, можно выполнить ввод прямо в таблице. При нажатии на имя параметра открывается диалог ввода параметра, содержащий развернутое описание. При подтверждении ввода в обоих случаях проводится проверка пределов, значения которых также извлекаются из объекта параметров.

Для логических полей в качестве элемента ввода используется флажок, для предустановленных значений — выпадающий список. В случае необходимости число различных элементов ввода можно расширить, например, добавив возможность ввода информации о цвете. Следует также заметить, что объекты класса CParameters могут использоваться и для работы с локальными переменными функций и методов, а также глобальными переменными.

Другая полезная особенность предлагаемого подхода состоит в том, что классы-потомки в своих конструкторах могут добавлять в m_Parameters свои, специфичные для них поля. При этом поля базовых классов уже будут зарегистрированы в m_Parameters (в конструкторах базовых классов). При необходимости любые параметры можно исключать из работы или, наоборот, возвращать. Все это особенно удобно в случае длинных цепочек наследования, так как избавляет от необходимости всякий раз конструировать отдельный диалог работы с параметрами, а также передавать в него текущие значения полей и принимать обратно новые.

Итак, роль графической конструкции, обеспечивающей доступ к данным, играет класс CParametersDialog, реализуемый для каждой системы разработки в системной части библиотеки (причем

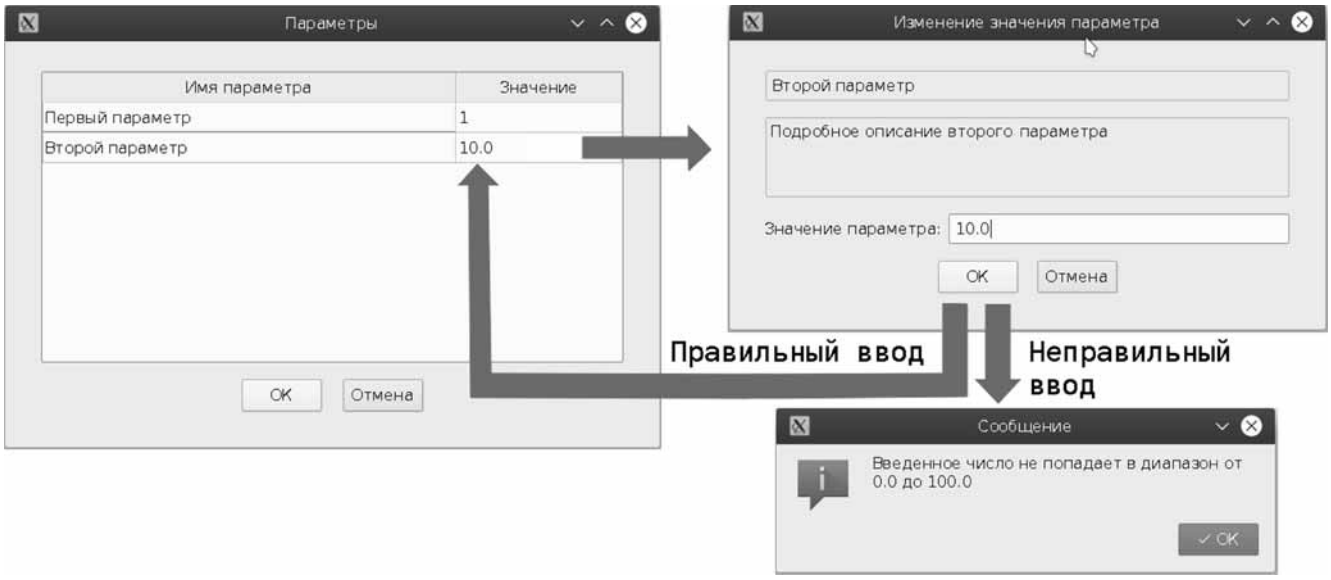

Рис. 2. Работа с параметрами

единожды для всех приложений, что важно). После этого все приложения, разработанные в рамках рассматриваемой библиотеки, получат способ формировать окна ввода и редактирования своих параметров. Таким образом, в программе приложения (в классе приложения или в других классах, с которыми оно взаимодействует) будут встречаться только лишь вызовы функции EF_ParametersDialog, что никаким образом не влияет на переносимость.

Следует отметить, что помимо обеспечения переносимости, функция EF_ParametersDialog и используемый в ней класс CParametersDialog ввода значений параметров также позволяют снизить степень рутинности разработки. Такой ввод позволит, с одной стороны, уменьшить число случайных ошибок и сократить объем исходного кода, а с другой стороны, упростить развитие проекта. Для новых полей классов больше не возникает необходимости в доработке соответствующих диалогов, а именно, в создании новых элементов ввода. Кроме того, больше не требуется вручную помещать значения из полей параметров в поля диалога и обратно.

Графика

В рамках рассматриваемого в настоящей статье подхода используется работа с 2D-графикой. При этом автор не пытается охватить все возможности, которые предоставляют для рисования те или иные библиотеки. Предлагается реализовать наиболее востребованные и общие для всех систем конструкции.

В системной части библиотеки для работы с перьями и кистями созданы два класса характеристик. Перья — цвет, толщина и стиль. Кисть характеризуется цветом и стилем. Данные параметры характерны для всех библиотек работы с графикой. При этом из стилей выбраны наиболее общие (встречающиеся во всех библиотеках работы с графикой). Таким образом, реализация функциональных возможностей данных классов в системной части Modules с использованием разных библиотек не представляет больших сложностей.

Для рисования реализованы следующие функции: рисование точки, линии, дуги, эллипса, прямоугольника, полигона. Такие функции присутствуют в любой библиотеке работы с графикой. На их основе разработаны функции рисования кривых, образованных отрезками и дугами, сплайнов Безье, а также сплошных фигур. Создан класс, реализующий вывод на экран шрифтов (на основе фигур из сплайнов). В данной редакции библиотеки используется собственный шрифт. Описание начертаний символов расположено прямо в коде библиотеки и не зависит от системы разработки. В дальнейшем предполагается реализовать загрузку из стандартных файлов шрифтов. Потребность в классе, реализующем работу со шрифтами на низком уровне (включая прорисовку символов), обусловлена перечисленными далее обстоятельствами.

- Не всегда одни и те же шрифты присутствуют в разных операционных системах.
- Работа со шрифтами привязывает приложение к конкретной системе разработки сильнее, чем работа с иными графическими конструкциями.
- В разных системах могут быть некоторые разночтения, что приведет к искажению формы и расположения шрифтов.
- Обычно высота шрифта измеряется в пикселях — целое число. Длина текста пропорциональна его высоте. Увеличение высоты шрифта на один пиксель может дать увеличение ширины текст на десятки или даже сотни пикселей без промежуточных значений. Свой шрифт дает возможность изменять произвольно и высоту, и ширину текста в любых

пределах с точностью до пикселя и вписывать текст в любые прямоугольники.

- Собственный шрифт изначально подготовлен для произвольного масштабирования в рамках используемых систем координат.

Обычно рисование проводится в координатной системе монитора, где левый верхний угол соответствует координате (0,0), а правый нижний, например, (1920, 1080). Если требуется преобразовать масштаб, то, как правило, данная задача возлагается на разработчика. В рамках библиотеки Modules изначально предлагается использовать иерархические координатные системы. Например, если происходит вывод на экран чертежа, то рисование проводится в координатах чертежа. Кроме того, на чертеже может быть текст со спецификацией, располагаемый в выделенной прямоугольной области. Этот текст, очевидно, удобнее выводить в системе координат указанного прямоугольника. В Modules для решения данной задачи используется специальный класс, используемый для стыковки координатных систем. Такие координатные системы при этом могут вкладываться одна в другую, и рисование может проводиться одновременно во всех системах, в зависимости от потребности. Для попиксельной прорисовки существует возможность получать точные координаты пикселя и его размер в координатах любой координатной системы.

Ранее, когда рассматривался вопрос о внутреннем шрифте библиотеки, одним из положительных моментов была отмечена масштабируемость. А именно, возможность вписывать текст в заданный прямоугольник (в локальной системе координат), а также правильно соотносить масштаб с остальными графическими конструкциями. Особую значимость такой механизм принимает в случае прорисовки текста мелким шрифтом с расположенными рядом графическими конструкциями. Речь идет о едином способе округления значения координат при переходе от локальной системы координат к оконной (в пределах размеров пикселя). Данный механизм позволяет избежать наползаний текста на линии либо, наоборот, устранить большие пробелы, не соответствующие реальному масштабу.

Типы данных

При разработке кроссплатформенных приложений очень важно обеспечить единообразное понимание используемых типов в разных системах разработки. В принципе, можно было бы использовать типы данных стандартной библиотеки в режиме "как есть". Однако их переопределение, с одной стороны, никак не скажется на функционировании программ, а с другой стороны, позволит обеспечить дополнительную степень свободы на случай необходимости.

В системе определен тип t_char, который определяется как char. Кроме того, определены шесть целочисленных типов: знаковые и беззнаковые 8-, 16- и 32-битные (t_int8, t_uint8, t_int16, t_uint16, t_int32, t_uint32). Один из известных проблемных вопросов переносимости состоит в том, что тип char в одних компиляторах является знаковым типом, а в других — беззнаковым. Для решения этого вопроса при использовании арифметических действий с однобайтными переменными предписывается использовать типы t_int8 и t_uint8.

Для совместимости с 64-битными системами используются типы данных ptrdiff_t и size_t из стандартной библиотеки. В 32-битных системах их размер 32 бита, в 64-битных — 64 бита, соответственно. Особая значимость данных типов состоит в том, что все операции по работе с памятью должны основываться именно на них. Важная особенность поддержки данных типов состоит в том, что при чтении и сохранении с использованием классов потоков ввода-вывода библиотеки Modules всегда записываются 64 бита, даже в 32-битных системах (старшие четыре байта для 32-битных систем заполняются нулями). Для работы с числами с плавающей запятой используются типы float (32 бита) и double (64 бита), переопределенные как t_float и t_double.

Еще один известный вопрос совместимости связан с порядком следования байт. Решается он на уровне потоков ввода-вывода библиотеки Modules. В зависимости от архитектуры системы, чтение и запись данных проводится таким образом, чтобы всегда обеспечивать единообразное расположение байтов в потоке.

Хорошим подходом в плане обеспечения системной независимости форматов данных считается использование известных текстовых форматов. В рамках библиотеки Modules предлагается использовать формат JSON. Реализован специализированный класс для работы с этим форматом.

Краткий обзор возможностей библиотеки Modules

Если библиотека не будет содержать широкий спектр классов различной направленности, то разработчику придется прибегать к использованию сторонних системно-зависимых библиотек, что будет нарушать общие положения предлагаемого подхода (за исключением стандартной библиотеки). Вместе с тем можно отметить и другой подход, который предполагает использование конструкций из стандартной библиотеки, что не сделает программу системно-зависимой. Итак, в библиотеку Modules входят перечисленные далее компоненты.

- Шаблоны контейнеров, позволяющие строить различные массивы и списки, облегчающие процесс разработки. Важная особенность контейнеров-массивов Modules состоит в том, что используется прямая адресация к массиву (поле шаблона класса контейнера — указатель на массив, размерностью которого заведует контейнер). Прямая адресация всегда работает быстрее вызова метода. При многократном обращении к ячейкам массива выигрыш во времени по сравнению с контейнерами

стандартной библиотеки может достигать десятков раз. Кроме того, есть возможность управлять шагом изменения буфера массива, что может обеспечить лучшую производительность при частом обращении на добавление и удаление элементов. Предусмотрен также контейнер-список. В случае недостатка функциональных механизмов всегда присутствует возможность использовать контейнеры стандартной библиотеки.

- Некоторые алгоритмы для контейнеров. Такие, например, как поиск в упорядоченном массиве методом половинного деления.
- Класс работы со строками. Кроме многобайтной кодировки есть полноценная поддержка юникода (который, по сути, принят за основу). Для быстрого доступа к произвольным символам в целях обеспечения высокой производительности (хотя и за счет большего использования памяти) для внутреннего представления строки используется массив 32-битных целых беззнаковых чисел.
- Класс для работы со временем.
- Класс работы с GUID и его константная версия для обеспечения удобной идентификации классов и любых уникальных наборов данных.
- Набор классов двоичных потоков ввода-вывода.
- Класс текстового потока ввода-вывода, базирующийся на двоичных потоках.
- Класс для автоматизации работы с параметрами.
- Классы для работы с данными (формат JSON и собственный двоичный формат с похожей организацией структуры данных).
- Классы для синхронизации потоков выполнения.
- Набор функций, облегчающих отладку.
- Битовые массивы.
- Класс для работы с координатными системами.
- Класс, содержащий математические функции (часть из них является упаковкой часто используемых математических функций, другая часть содержит набор функций для вычислительной геометрии, разработанных автором библиотеки).
- Набор классов для работы с графикой.
- Класс для работы с файловой системой.
- Класс для работы со случайными значениями.
- Набор классов для моделирования движущихся объектов и изменяющихся во времени координат с функцией паузы и масштабирования времени.

Следует также отметить, что библиотека активно развивается. В случае обнаружения областей, не охваченных ее возможностями, проводится разработка соответствующих классов.

На основе предлагаемой библиотеки уже построены несколько программы, функционирующих в операционных системах Windows и Linux:

- специализированный текстовый редактор для работы с системой LaTeX;
- программа оптимизации перемещений инструмента на машинах листовой резки металла с ЧПУ (с графическим отображением условий задачи и решения);
- программа оптимизации работы связной сети, образованной движущимися объектами с коммуникационными устройствами ограниченного радиуса действия (с визуальным отображением процесса и редактированием данных);
- утилита шифрования файлов.

Заключение

Предложен подход к разработке приложений C++, позволяющий обеспечить возможность относительно легкого переноса программ в различные системы разработки. Такая возможность может быть востребована многими программистами, так как открывает новые перспективы применения их программного обеспечения. Суть подхода состоит в построении промежуточной библиотеки классов, использующей основные принципы переносимости программного обеспечения. Использование такой библиотеки избавляет программистов прикладного программного обеспечения от необходимости применения данных принципов в коде каждой своей программы напрямую. Реализована библиотека классов C++, получившая название Modules. Подробное описание самой библиотеки отсутствует в статье, с одной стороны, по соображениям экономии места, а с другой стороны, в настоящей статье автор предполагал в большей степени изложить концептуально-методологические аспекты предлагаемого подхода, чем проиллюстрировать его практическую сторону. Системная часть библиотеки Modules на данный момент частично использует функциональные возможности библиотеки Qt, частично — стандартной библиотеки. По мере необходимости планируется разработка версии для иных систем разработки (например, Microsoft Visual C++, Borland C++ Builder и т. д.).

На основе библиотеки Modules было разработано несколько программ, функционирующих под управлением операционных систем Windows и Linux. В ходе разработки этих программ выполнены значительные доработки как самой библиотеки, так и в направлении совершенствования основ самой методологии.

Список литературы

1. **Джосаттис Н. М.** Стандартная библиотека C++: Справ. руководство, 2-е изд. М.: Вильямс, 2014. 1136 с.
2. **Керниган Б., Райк Р.** Практика программирования. М.: Вильямс, 2004. 288 с.
3. **Шлее М.** Qt 5.3. Профессиональное программирование на C++. СПб.: БХВ-Петербург, 2015. 928 с.
4. **Чирков М.** Перевод руководства по GTK + 2.0. URL: http://www.opennet.ru/docs/RUS/gtk_plus/
5. **Кузнецов А. Н.** Программирование на C++ с JUCE 4.2.x: Создание кроссплатформенных мультимедийных приложений с использованием библиотеки JUCE на простых примерах. Алматы: Linmedsoft, 2016. 383 с.
6. **Smart J., Hock K., Csomor S.** Cross-Platform GUI Programming with wxWidgets. Upper Saddle River: Prentice Hall PTR, 2005. 262 p.

One Approach to Developing Cross-Platform Applications in C++

P. A. Chentsov, chentsov.p@mail.ru, Institute of Mathematics and Mechanics Ural Branch of RAS, Yekaterinburg, 620219, Russian Federation

Corresponding author:

Chentsov Pavel A., Senior Researcher, Institute of Mathematics and Mechanics Ural Branch of RAS, Yekaterinburg, 620219, Russian Federation,
E-mail: chentsov.p@mail.ru

Received on October 10, 2018
Accepted on November 26, 2018

The technology of developing cross-platform software in the C++ language is considered. Programs are developed with using of a special library of classes. This library contains software constructions ensuring portability of the code, and is easily adaptable to different operating systems, compilers, and C++ base libraries. The dependence of the user interface of programs on the operating systems, compilers and base libraries persists, but some techniques of this dependence minimization are proposed.

Keywords: *C++, cross-platform programming*

For citation:

Chentsov P. A. One Approach to Developing Cross-Platform Applications in C++, *Programmnaya Ingeneria*, 2019, vol. 10, no. 3, pp. 105—113.

DOI: 10.17587/prin.10.105-113

References

1. **Josuttis N. M.** *The C++ Standard Library. A Tutorial and Reference. Second Edition*, New York, Addison-Wesley, 2012, 1131 p.
2. **Kernighan B. W., Pike R.** *The Practice of Programming*, New York, Addison-Wesley, 1999. 267 p.
3. **Shlee M.** *Qt 5.3. Professional'noe programmirovanie na C++*, Saint-Petersburg, BHV-Peterburg, 2015, 928 p. (in Russian).
4. **Gale T., Main I. & the GTK team.** *GTK + 2.0 Tutorial*, available at: https://developer.gnome.org/gtk-tutorial/stable/
5. **Kuznecov A. N.** *Programmirovanie na C++ s JUCE 4.2.x: Sozdanie krossplatformennyh mul'timedijnyh prilozhenij s ispol'zovaniem biblioteki JUCE na prostyh primerah*, Almaty, Linmedsoft, 2016, 383 p. (in Russian).
6. **Smart J., Hock K., Csomor S.** *Cross-Platform GUI Programming with wxWidgets*, Upper Saddle River: Prentice Hall PTR, 2005, 262 p.

ИНФОРМАЦИЯ

С 21 по 24 мая 2019 г. в Самаре состоится

V Международная конференция и молодежная школа

«Информационные технологии и нанотехнологии»

(ИТНТ-2019)

Тематика конференции

- ◇ Компьютерная оптика и нанофотоника
- ◇ Обработка изображений и дистанционное зондирование Земли
- ◇ Математическое моделирование физико-технических процессов и систем
- ◇ Науки о данных

Подробности на сайте конференции: http://itnt-conf.org

УДК 004.41+65.01 DOI: 10.17587/prin.10.114-124

В. К. Батоврин, канд. техн. наук, зав. кафедрой, e-mail: batovrin@mirea.ru, Российский технологический университет (МИРЭА), г. Москва, **Б. А. Позин,** д-р техн. наук, проф., техн. директор ЗАО "ЕС-лизинг", e-mail: bpozin@ec-leasing.ru, "ЕС-лизинг", Национальный исследовательский университет Высшая школа экономики, г. Москва

Инженерия требований на современном промышленном предприятии

Выделены ключевые проблемные вопросы производственной инженерии требований. В контексте разрешения этих вопросов обсуждены: схема формирования и использования требований в жизненном цикле системы; взаимосвязь между отдельными процессами инженерии требований и их результатами; классификация требований; особенности обеспечения прослеживаемости требований и их распределения между элементами системы.

Ключевые слова: *инженерия требований, определение требований, качество требований, документирование требований, прослеживаемость требований*

Введение

Хорошо налаженная инженерия требований является одним из ключевых способов достижения успеха при создании систем различной сложности и назначения. В настоящее время развитие методов и средств практического использования инженерии требований связывают, как правило, с проблематикой создания программных и программно-насыщенных систем. Здесь можно, например, отметить работы К. Вигерса [1], К. Поля [2], Э. Халл и соавторов [3], а также ряда других специалистов. При рассмотрении проблемных вопросов программной инженерии тематика инженерии требований к программному продукту всегда выделяется в качестве раздела, требующего отдельного, подробного рассмотрения (например, [4]). Основные стандарты инженерии требований разрабатываются под эгидой Первого объединенного технического комитета ИСО и МЭК (ISO/IEC Joint Technical Committee 1, JTC1), занятого стандартизацией в области информационных технологий. Таким образом, складывается положение, когда некоторые специалисты связывают инженерию требований главным образом с деятельностью по формированию и развитию ИТ-решений.

Принимая во внимание отмеченную выше связь, следует, однако, подчеркнуть, что практика использования инженерии требований на предприятиях, занятых созданием сложных технических систем, таких как энергетические машины, средства транспорта, системы специального назначения и многих других, требует учета особенностей их архитектуры и проверки соответствия требованиям, предъявляемым к этим объектам (далее для краткости изложения — требованиям). При работе с такими требованиями приходится обязательно учитывать особенности технологий производства систем, их комплексирования, управления конфигурацией, испытаний, сопровождения, а также логистические аспекты, включая налаживание эффективного взаимодействия с множеством поставщиков, как готовых, так и заказных решений. Таким образом, на современном промышленном предприятии инженерия требований не замыкается только на ИТ-решениях и не ограничивается, как это было принято "старой школой", этапами разработки концепции системы и технического задания. На таком предприятии работа с требованиями является неотъемлемой частью инженерной деятельности по поиску и реализации решений, относящихся и к созданию системы, и к ее материальному воплощению, и к ее испытаниям, и к успешному применению и, наконец, к прекращению использования. Следовательно, в центре внимания специалиста по инженерии требований должны находиться полный жизненный цикл (ЖЦ), включая разработку и изготовление системы, а также предполагаемое окружение системы и особенности среды, окружающей предприятие, ответственное за разработку. Подобный подход можно условно назвать производственной инженерией требований.

В настоящей работе предпринята попытка выделения и анализа основных парадигм и методов производственной инженерии требований в целях выявления ключевых аспектов, определяющих успешный характер этой деятельности. Кроме того, рассмотрены вопросы адаптации типовых процессов инженерии требований к условиям промышленного предприятия.

Проблемы производственной инженерии требований

Важнейшими задачами работы с требованиями в условиях промышленного предприятия являются учет сложности, гетерогенности и многовариантности создаваемых технических систем, а также сложности

логистической поддержки ЖЦ систем. Например, при создании современной авиационной техники используют сотни различных технологий. К самолетам предъявляют десятки тысяч оригинальных требований, в их числе — учет норм летной годности самолетов, которые дополнительно регламентируют тысячи обязательных требований к воздушным судам [5]. Для того чтобы охарактеризовать сложность логистической поддержки, достаточно оценить состав кооперации, реализующей эту поддержку. Так, в создании самолета B777 участвуют более 350 поставщиков различных конструкторских, технических, производственных, программных, технологических и других решений, а также девять поставщиков, обеспечивающих испытания. Для самолета B787 общее число таких поставщиков превышает 450, а для самолета A380 — 500 [6]. При этом для систем, подобных атомным электростанциям или морским буровым платформам, проблемы учета сложности набора требований еще масштабней, чем в случае воздушных судов и других транспортных систем.

Еще одна особенность производственной инженерии требований заключается в том, что на протяжении ЖЦ системы необходимо контролировать ее целостность и выполнение так называемых критических требований, предъявляемых к специально отобранным свойствам системы. В системной и программной инженерии сложились нормативные основы деятельности по контролю целостности [7, 8]. Кроме того, имеются общепризнанные рекомендации по увязке процессов обеспечения уровней гарантии качества и целостности системы с процессами инженерии требований. Пример подобной увязки в разрезе гарантий безопасности можно найти в стандарте ARP4754A [9], принятом в нашей стране в качестве обязательного нормативного руководства для разработчиков воздушных судов [10]. Сложности учета и реализации требований на этом направлении обусловлены тем обстоятельством, что применительно к созданию сложных инженерных объектов подобная отечественная практика по существу отсутствует. При этом некоторый опыт обеспечения целостности и гарантий качества, имеющийся у наших разработчиков программных средств, не может быть непосредственно и без глубокой переработки перенесен в область технических систем.

Зачастую по объективным причинам возникает необходимость в разработке нескольких вариантов изделия, которые базируются на единой платформе, а особенности таких вариантов связаны с использованием опциональных решений. Подобная ситуация характерна для автомобильной, авиационной и ряда других отраслей промышленности. В таких случаях не только появляется необходимость определить исходную совокупность требований и исходную конфигурацию изделия, но и требуется обеспечить гарантированную прослеживаемость в разрезах "требования к функциям (вплоть до объектов конфигурации) — требования к системе (вплоть до объектов конфигурации) — реализованные объекты конфигурации". При сравнительно небольшом (от нескольких десятков до сотен) числе опций на уровне подсистем и отдельных элементов приходится анализировать до $10^6...10^9$ и более вариантов конфигурации [11]. Это обстоятельство, в свою очередь, требует особого внимания к решению вопросов автоматизации процессов инженерии требований. Проблемные вопросы производственной инженерии требований обусловлены, таким образом, не только сложностью создаваемых систем, необходимостью обеспечения контроля их целостности и гарантий качества, но и необходимостью поддержки многовариантности. Отмеченные вопросы можно охарактеризовать как ключевые для производственной инженерии требований.

Анализ показывает, что в центре внимания производственной инженерии требований должны находиться: моделирование исходной совокупности требований во взаимосвязи с архитектурой изделия и выделенными объектами конфигурации; практическое прослеживание изменений требований; определение и формализация процессов инженерии требований, адаптированных к условиям промышленного предприятия; повышение уровня их зрелости и критерии контроля запланированных результатов. Именно эти аспекты вносят решающий вклад в успех деятельности системных инженеров и позволяют обеспечить успешную автоматизацию процессов инженерии требований. К сожалению, примеры эффективного решения подобных задач применительно к ЖЦ сложных технических систем на отечественных промышленных предприятиях авторам неизвестны.

Отметим также, что в настоящее время на рынке имеется более сотни инструментальных средств, предназначенных для автоматизации процессов работы с требованиями [12]. При таком их разнообразии осознанный выбор подходящих инструментальных средств возможен только при наличии высокой квалификации как в области инженерии требований, так и в сфере гармонизации процессов реализации этих требований с другими процессами предприятия. В этих условиях отечественные предприятия, как правило, не выбирают наиболее подходящее инструментальное средство, а пытаются настроить для решения своих задач наиболее доступное из них. При этом недостаточно внимания уделяется вопросам налаживания процессов инженерии требований по существу и управления ими. Такое положение ведет к неминуемому росту проектных рисков и повышению затрат на разработку и реализацию принятых решений.

Общая схема формирования и использования требований в жизненном цикле системы

Современные стандарты системной и программной инженерии рассматривают создание систем не само по себе, а в рамках некоей программы работ или проекта, направленного на реализацию потребности заинтересованных сторон (ЗС). К их числу, прежде всего, относятся потенциальные собственники и

пользователи системы [13—15]. Заинтересованные стороны инвестируют средства в этот проект или программу и, как правило, впоследствии организуют правильную и эффективную эксплуатацию системы, ее сопровождение и развитие. Таким образом, управление требованиями к системе осуществляется на протяжении ее полного ЖЦ. Современная парадигма международной системы стандартизации, в частности ISO/IEC/IEEE, основана на рассмотрении полного жизненного цикла системы, в то время как разработанные в конце прошлого века ГОСТ 34 рассматривает в качестве продолжительности действия ТЗ период до сдачи системы заказчику, а ГОСТ 15 — до постановки изделия на производство [19, 20]. Современная международная система стандартизации учитывает, что после постановки сданной или произведенной на заводе системы (изделия) на балансовый учет заказчика или покупателя она становится активом владельца. Система при этом такой же актив, как здание, машина или любой другой актив. Его нужно эксплуатировать и обслуживать, проводить ежегодное бюджетирование этой деятельности.

На рис. 1 показана схема формирования требований и взаимосвязи между системой и различными категориями требований, учитывающая рекомендации стандарта ISO/IEC/IEEE 29148. Согласно этой схеме системы создаются предприятиями, функционирующими в определенном окружении. Предприятие при этом имеет дело одновременно с тремя категориями требований, к числу которых относятся:

- деловые требования, определяемые на уровне управления бизнесом;
- требования заинтересованных сторон, которые определяются на уровне программ/проектов и/или деловых операций;
- требования собственно к целевой системе, а также к ее элементам и к программному обеспечению (ПО).

Все три категории требований тесно связаны между собой отношениями прослеживаемости, т. е. возможностью установления и поддержки взаимосвязи между исходными и производными требованиями, их источниками и верификацией фактов выполнения требований. Именно поэтому требования к системе и к ее элементам, включая ПО, выявляются изначально на уровне среды предприятия (в том числе программы или проекта) на основании представлений о необходимых возможностях будущей системы. Последние сформированы ЗС, исходя из потребностей и особенностей бизнеса предприятия или потребностей рынка. Анализ этих возможностей с позиций их реализуемости и последующее согласование с ЗС приводит к появлению *требований к использованию системы.* В ходе декомпозиции этих требований и формирования архитектуры будущей системы разрабатываются требования, используемые в процессе реализации системы и ее элементов, включая ПО, которые также согласуются с соответствующими ЗС. В отличие от прослеживаемости требований к программному обеспечению, поддержание прослеживаемости требований к техническим системам зачастую требует построения "очень длинных" трасс в сотни взаимосвязанных вершин. Это обстоятельство приводит к необходимости создания специальных средств автоматизации анализа и документирования трасс. Для эффективного решения этой задачи необходимо создание инструментальных средств, обеспечивающих ведение и анализ требований разных видов, которые в ходе реализации проекта/программы системы хранятся в единой базе данных (репозитории требований), доступ к которому

Рис. 1. Общая схема формирования требований с учетом положений [13]

предоставляется уполномоченным специалистам. По существу репозиторий должен поддерживаться в актуальном состоянии в течение ЖЦ системы.

В целях выстраивания эффективной производственной инженерии требований промышленному предприятию полезно наладить и поддерживать три гармонизированных между собой процесса, а именно: *процесс анализа бизнеса (бизнес-процессов предприятия)*; *процесс определения нужд и требований ЗС; процесс определения требований к системе.* Эти процессы должны быть сопряжены с процессами предприятия [14] и должны являться основой для формирования документов (выделенных ниже), установленных предприятием для внутренних процессов.

В рамках процесса *анализа бизнеса* с учетом принятых концепций ЖЦ, включая концепцию деятельности предприятия, предприятие определяет деловые требования, реализация которых может решить проблему(ы) предприятия и/или позволит ему воспользоваться выявленной возможностью. Традиции составления и утверждения концепции деятельности в увязке с другими типовыми концепциями ЖЦ, определенными, например, в руководстве INCOSE [16], на отечественных промышленных предприятиях, как правило, отсутствуют. Именно поэтому отмеченному вопросу следует уделить особое внимание. Концепция деятельности используется в качестве основы, которая в рамках организации указывает направление развития и общие характеристики будущего бизнеса и систем. В рамках проекта такая концепция помогает понять его предпосылки, а для лиц, занятых инженерией требований, облегчает выявление требований ЗС. Требования, определенные в процессе анализа бизнеса, фиксируются в спецификации деловых требований (*business requirements specification — BRS*) и включаются в базу данных требований.

В рамках процесса *определения нужд и требований ЗС* ответственные за программу/проект инженеры описывают эти потребности совместно с выявленными ЗС. Они определяют совокупность сценариев использования целевой системы и ее необходимые возможности, включая особенности взаимодействия с окружением и потребности в системах обеспечения. Полнота и доказательность определения потребностей ЗС должны быть достаточны для получения разрешения на начало и финансирование новой программы/проекта в рамках процесса управления портфелем проектов. Эти результаты должны содержать и техническую информацию, необходимую для формирования запроса на предложение, если система должна быть приобретена по контракту в рамках процесса приобретения [14]. Такая информация необходима для получения разрешения на разработку и последующую реализацию системы, если она создается в ответ на запросы рынка. Кроме того, важнейшим результатом процесса является преобразование нужд ЗС в их требования. Такие требования формализуются в строго структурированные утверждения, которые могут быть проверены посредством их валидации (в рамках валидации требований высокого уровня до их реализации) и/или верификации требований к системе по отношению к требованиям ЗС высокого уровня [13]. В промышленности требования ЗС, идентифицированные и документированные на уровне программы/проекта, служащие исходными данными для определения функций целевой системы, иногда называют требованиями высокого уровня (*top level requirement — TLR*) [9].

С определением нужд и требований ЗС тесно связан переход от представления об отдельных возможностях целевой системы, соответствующих потребностям ЗС, к полному, согласованному описанию ее поведения в процессе использования по назначению, т. е. к формированию концепции использования системы. Действия на этом направлении включают определение того, что должна делать система, а именно — выделение набора функций и выявление требований к качеству их выполнения, а также обоснование требований к характеристикам и показателям ее функционирования, таким как производительность, надежность и т. п. Описание того, как система должна реализовывать функции, на этом этапе принципиально не проводится.

Кроме того, при формировании концепции использования необходимо обеспечить двустороннюю прослеживаемость между требованиями к функциям системы и источниками этих требований. В качестве возможных рекомендаций по разработке концепции использования могут быть рассмотрены положения стандарта ANSI/AIAA G-043A-2012 Американского института аэронавтики и астронавтики (*American Institute of Aeronautics and Astronautics*) [17]. Требования ЗС фиксируются в спецификации требований ЗС (*stakeholder requirements specification — StRS*) и включаются в репозиторий требований.

В рамках процесса *определения требований к системе* системные инженеры и инженеры по направлениям преобразуют представление ЗС о требуемых возможностях и функциях системы в техническое представление о решении, которое пригодно к реализации на производстве и к эксплуатации пользователем. Это техническое представление должно содержать критерии того, что, насколько хорошо и при каких условиях должна делать система, чтобы соответствовать деловым требованиям и требованиям ЗС, а также проектным, конструктивным и другим требующим учета ограничениям. Технические требования многообразны, их виды более подробно будут рассмотрены ниже. Поскольку реализация каждого из таких требований связана с определенными затратами, важно, чтобы был установлен, с одной стороны, полный, с другой стороны — минимальный набор требований к системе, обусловленных требованиями ЗС. После определения высокоуровневой совокупности требований к системе, их необходимо распределить и передать на последовательно более низкие уровни системной иерархии. Существенно, чтобы по мере выполнения процессов привязки и распределения требований поддерживалась их двусторонняя прослеживаемость вплоть до самого низкого иерархического уровня. Для этого на каждом из иерархических уровней выполнения проекта каждому (проиндексированному) требованию ставится в соответствие множество атрибутов, отражающих

связь между исходными и производными требованиями или их состояниями. Здесь приводятся предварительные предложения авторов по атрибутированию требований, которые подлежат развитию в последующих работах. Однако они частично использовались в процессе создания, сопровождения и развития программных систем довольно большого масштаба.

Атрибутирование требований направлено на решение нескольких задач. Во-первых, на то, чтобы обеспечить возможность прослеживания системных функций "снизу вверх", поскольку на достаточно низких уровнях декомпозиции такая зависимость может теряться при выборе базовых элементов реализации системы. Во-вторых, на поддержку механизмов, позволяющих маркировать каждый элемент архитектуры атрибутом принадлежности к системной функции. Целью и в этом случае является обеспечение возможности двустороннего прослеживания (прежде всего "снизу вверх", от элементов архитектуры к системным функциям). В-третьих, атрибутирование направлено на создание механизмов, позволяющих маркировать каждый элемент конфигурации атрибутами принадлежности к элементу архитектуры и плановым/достигнутым состояниям в проекте/программе/ЖЦ. В некоторых случаях приходится вводить виртуальные элементы архитектуры. В конструкторской практике граница между системными функциями при их реализации может быть размыта. Не исключены компоновки нескольких системных функций или их элементов в некоторые конструктивные элементы. Такие элементы могут быть не предусмотрены ЗС, однако могут оказаться эффективными на практике, с учетом ремонтопригодности или других критериев.

При определении требований следует также уделять особое внимание гарантиям того, что требования сформулированы надлежащим образом. Полезные предложения можно найти в Руководстве INCOSE по написанию требований [18]. В нем представлены рекомендации по написанию требований, а также рассматриваются характеристики качества отдельных требований и наборов требований, атрибуты формулировок отдельных требований и правила описания отдельных требований. Исходная совокупность полных, точных, недвусмысленных требований к системе и ее элементам формируется на ранних стадиях ее ЖЦ. Эти требования фиксируются в утвержденных и опубликованных спецификациях требований к системе (*system requirements specification — SyRS*) и вносятся в репозиторий требований, доступ к которому следует обеспечить для всех уполномоченных сторон. Спецификации *SyRS* используется далее для специфицирования требований к элементам системы, включая ПО. В последнем случае разрабатывается спецификация требований к ПО (*software requirements specification — SRS*). Таким образом, должно быть создано дерево спецификаций, позволяющее полностью определить (описать) систему и организовать ее производство.

Взаимосвязь между процессами инженерии требований и их результатами

Традиционное ТЗ (ТТЗ) является контрактным документом и выступает в качестве неотъемлемой части договора между заказчиком и поставщиком продукции [19, 20]. В силу этого обстоятельства интересы собственно разработчиков решений, связанные с возможностью прослеживания связей между особенностями использования (эксплуатации) системы и требованиями к ее материальному воплощению, а также к отдельным элементам не всегда учитываются при формулировании требований в ТЗ. Кроме того, требования, содержащиеся в ТЗ, почти всегда требуют интерпретации, в частности, выделения из состава комплексных требований единичных требований, используемых при верификации решений. Эти обстоятельства зачастую приводят к заметно-

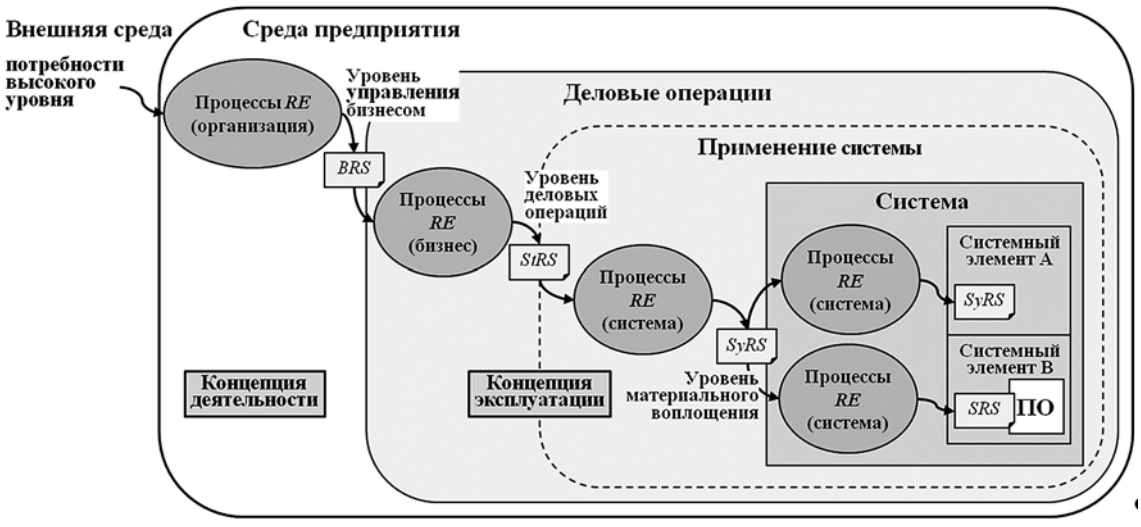

Рис. 2. Взаимосвязь между процессами инженерии требований (*RE*) и их результатами

му росту затрат на проект и сроков его реализации. На рис. 2 показана последовательность процессов инженерии требований и место соответствующих спецификаций, рекомендованные современной инженерией требований [13]. Эта схема, в отличие от упомянутой традиционной схемы ТЗ (ТТЗ), позволяет систематически использовать совокупность взаимоувязанных спецификаций требований, детализировать требования до необходимой разработчикам глубины, а также на протяжении ЖЦ системы отделять друг от друга управленческие и технические аспекты. В результате спецификации требований становятся неотъемлемой частью пакета документов. На их основе ЗС и предприятие реализуют политики управления ЖЦ продукции и налаживают процессы ЖЦ, что позволяет смягчить риски программ и проектов по созданию систем. Кроме того, использование подобного подхода дает полезную возможность независимого управления функциональными требованиями и требованиями к продукции, а также позволяет существенно упростить деятельность по установлению и поддержанию прослеживаемости требований.

Специфицированные и увязанные между собой требования составляют основу для всех работ по проектированию и производству, по испытаниям и эксплуатации системы, по ее техническому обслуживанию и прекращению использования. Как следствие, определяются стоимость и график реализации проекта. Однако работа с требованиями к системе на этом не заканчивается. По мере увеличения уровня детализации проектно-конструкторских решений, а также по мере их материализации процессы инженерии требований итеративно повторяются на каждом из этапов проекта с непрерывной обратной связью. Полученные результаты увязываются с принятыми на уровне предприятия концепциями ЖЦ, в частности, с концепциями приобретения, развертывания, использования и сопровождения [16]. Эти результаты также формируют основу для моделирования архитектуры, для выделения и учета подходящих в технологическом или техническом отношениях элементов, из которых может быть построена система, а также для учета неизбежного изменения требований на протяжении ЖЦ, что дает возможность налаживания эффективного управления конфигурацией. В итоге может быть сформирован результирующий репозиторий требований. Данные репозитория могут успешно использоваться в процессе верификации и валидации как системы в целом, так и отдельных ее элементов.

Следует отметить, что в практике производственной инженерии первостепенную важность при составлении спецификаций требований имеют обязательные требования, например, сертификационные требования, которые в сочетании с требованиями, полученными в результате анализа ТЗ, позволяют сформировать исходные (начальные) наборы требований. Использование ноу-хау из предыдущих программ расширяет эти исходные (начальные) наборы требований до уровня, достаточного для первоначальных исследований в области архитектуры целевой системы.

В таблице на примере воздушного судна и его тормозной системы показан фрагмент возможной спецификации требований к системе в целом, составленной в соответствии с рекомендациями SAE AIR 6110 [21] и содержащей указания как на источник требований, так и на его обоснование. Общие рекомендации по формированию спецификаций требований можно найти в стандартах ISO/IEC/IEEE 29148 [13] и ISO/IEC/IEEE 15289 [22].

Пример спецификации требований к воздушному судну (фрагмент)

Требование	Описание	Источник требования	Обоснование
S18-ACFT-R-0009	Самолет должен иметь средства торможения на земле согласно АП 25.735 [5]	АП 25. Раздел 25.735	Минимальный стандарт, выполнение требований которого обязательно для сертификации воздушного судна
S18-ACFT-R-0110	Самолет должен иметь автоматическое торможение	Производное требование	Технологические усовершенствования возможностей автоматической посадки согласно категории ИКАО CAT III b и исследования рынка, отчет о потребностях клиентов
S18-ACFT-R-0135	Самолет должен обеспечивать противоюзовую функцию	Производное требование	Всепогодная эксплуатация и устойчивость самолета во время пробегов по взлетно-посадочной полосе и исследования рынка, отчет о потребностях клиентов
S18-ACFT-R-0184	Самолет должен иметь тормозную систему с гидравлическим приводом	Производное требование	Исследования (отчет TS-18-XXXX) показали, что тормозные системы с гидравлическим приводом экономически целесообразней систем с электрическим приводом
S18-AC FT-R-0185	Экипаж должен иметь возможность пересилить автоматическое торможение	АП 25.25.735(c) [5] и S18-ACFT-R-0110	Экипаж сам выбирает режим управления тормозами

Виды требований, цели и ожидаемые результаты их использования в жизненном цикле систем и их программного обеспечения

Требования к целевой системе подразделяются по видам с учетом совместных позиций их использования всеми ЗС, а именно участниками процессов ЖЦ на стороне заказчика или пользователя. Прежде всего, целью разработки требований является описание совместного видения системы заинтересованными сторонами. Такое видение включает:

- обязательные или процессуальные требования, в частности, соответствие системы или процесса ее создания необходимым законам, нормам и правилам, установленным для систем данного класса (если таковые существуют);
- требования к функциям системы и ее составных частей, а также к взаимодействию со смежными системами (внешние интерфейсы) и между элементами системы (внутренние интерфейсы);
- требования к характеристикам, которые относятся к функциям, обычно устанавливаются количественно и определяют степень или уровень выполнения функции или задачи;
- нефункциональные требования, например, конструктивные, технологические, эксплуатационные и другие свойства, важные для пользователя в его представлениях об удобстве использования, а также для заказчика в его представлениях о способе производства, эксплуатации, утилизации и т. п.

Как правило, на практике все виды требований формируются итерационно, как по причине сложности самой системы, так и в силу необходимости обеспечения согласованности и непротиворечивости требований. Это кропотливая системная работа, которую выполняют совместно специалисты заказчика, генподрядчика и других ЗС. Такая работа для сложных систем сопровождается проведением НИР и НИОКР с целью исследовать лучшие решения и апробировать некоторые из них на макетах. Основные виды требований, которые формируются на этом этапе, представлены на рис. 3.

Рис. 3. Виды требований к системе

Следует заметить, что требования к функциям не только отражают представление ЗС об основных функциях и использовании (эксплуатации) будущей системы, но также специфицируют функции следующего (одного или нескольких) уровня декомпозиции основных функций с учетом их общесистемной значимости. Кроме того, функциональные требования отражают внешние связи будущей системы со смежными с ней другими системами, которые являются, как правило, источниками данных для решения задач целевой системы или получателями данных от нее. Как следует из представленного описания, *все подсистемы, входящие в состав целевой системы, в совокупности реализуют ее функции.* Как следствие, требования к таким подсистемам по существу задают ожидаемые характеристики реализуемой функции. Таким образом, эти требования могут и должны быть основой для проверки степени реализации соответствующей функции.

Нефункциональные требования отражают условия, в которых система должна работать или существовать, доступность системы, ожидаемые надежность и качество, а также человеческие факторы, например, требования к квалификации или удобству использования. Кроме показателей, которые характеризуют способность системы к выполнению функций в ожидаемых условиях применения, нефункциональные требования могут специфицировать способ (технологии) производства, режимы эксплуатации и обслуживания системы, другие аспекты существования системы на протяжении ЖЦ.

Из изложенного выше следует, что *целью* формирования требований является не только описание внешних свойств создаваемой системы для ее приемки, но и возможность использования функциональных требований для анализа степени полноты и качества реализации функций системы. Кроме того, требования являются основой для подготовки планов проверки соответствия целевой системы и подсистем, входящих в ее состав, в рамках процедуры верификации.

Отметим, что правильно построенные процессы инженерии требований позволяют на протяжении ЖЦ не только прослеживать соответствие требований архитектуры и конфигурации "сверху вниз" при декомпозиции системы, но и при определенных условиях "снизу вверх", т. е. при проверке степени реализации требований. По причине большой

размерности системы трассы прослеживания становятся очень длинными.

Важно также отметить, что юридически ответственность за реализацию требований к системе определяется условиями договора на ее разработку. По завершении действия договора прекращает действие и ТТЗ/ТЗ, а ЖЦ системы продолжается. Система подлежит производству, развертыванию, эксплуатации, сопровождению, развитию и, наконец, утилизации. На этих стадиях необходимо хранить и использовать, сопровождать и развивать описание, заменяющее ТТЗ/ТЗ, как у ЗС, так и у разрабатывающих организаций (прежде всего интеграторов). *Именно эту роль и должны играть упомянутые выше спецификации требований к системе в том состоянии, которое сформировалось к текущей стадии ЖЦ этой системы, а также требований.* Для организаций, создающих серии однотипных систем, наличие репозиториев, в которых хранится и/или поддерживается комплекс требований, повышает эффективность управления конфигурацией, создает возможности выработки типовых решений, их унификации и стандартизации и снижения совокупной стоимости ЖЦ.

Распределение требований между элементами системы

Как отмечалось выше, исходными данными для инженеров, создающих целевую систему, являются представления о ее функциях, отраженные в требованиях высокого уровня — *TLR*. В свою очередь, *TLR* увязываются с концепцией использования. Такая концепция на ранних этапах ЖЦ в текстовом или в графическом виде в контексте потребностей пользователей и других ЗС описывает эксплуатационные возможности и характеристики целевой системы в среде ее использования. Она задает основу для определения рабочего пространства, возможностей системы и ее интерфейсов, а также условий эксплуатации. Причем функции целевой системы выделяются итерационно путем логического сопоставления требований высокого уровня и представлений о поведении системы в среде функционирования, которые отражены в принятой концепции использования. В результате формируется полный перечень функций целевой системы, а также связанных с ними требований к функциям и интерфейсам этих функций, т. е. исходная функциональная конфигурация создаваемого изделия (рис. 4).

Вместе с тем представления о системе или о ее свойствах в окружающей среде, воплощенные в ее элементах и связях между ними, в принципах проектирования и развития системы, формируются в рамках инжиниринга архитектуры. Инжиниринг архитектуры предполагает использование типовых архитектурных представлений [23]. Эти представления должны быть зафиксированы на уровне предприятия. При их выявлении за основу могут быть взяты либо представления, используемые лидерами отрасли, либо библиотечные представления [24].

Следует также учитывать, что стандарт ISO/IEC/IEEE 42010 в контексте процесса системной инженерии рекомендует использовать *точку зрения декомпозиции и привязки* (*decomposition and allocation viewpoint*) [23]. Эта точка зрения предусматривает решение четырех ключевых задач:

- идентификацию требований к системе с присвоением уникального идентификатора каждому из требований и с декомпозицией (при необходимости) исходных требований на производные требования, определяющие полный набор требований к системе;
- декомпозицию системы на элементы с обязательной идентификацией интерфейсов между элементами;
- распределение (привязку) требований между элементами системы с установлением взаимного соответствия между полным набором требований и всеми элементами системы так, чтобы каждый

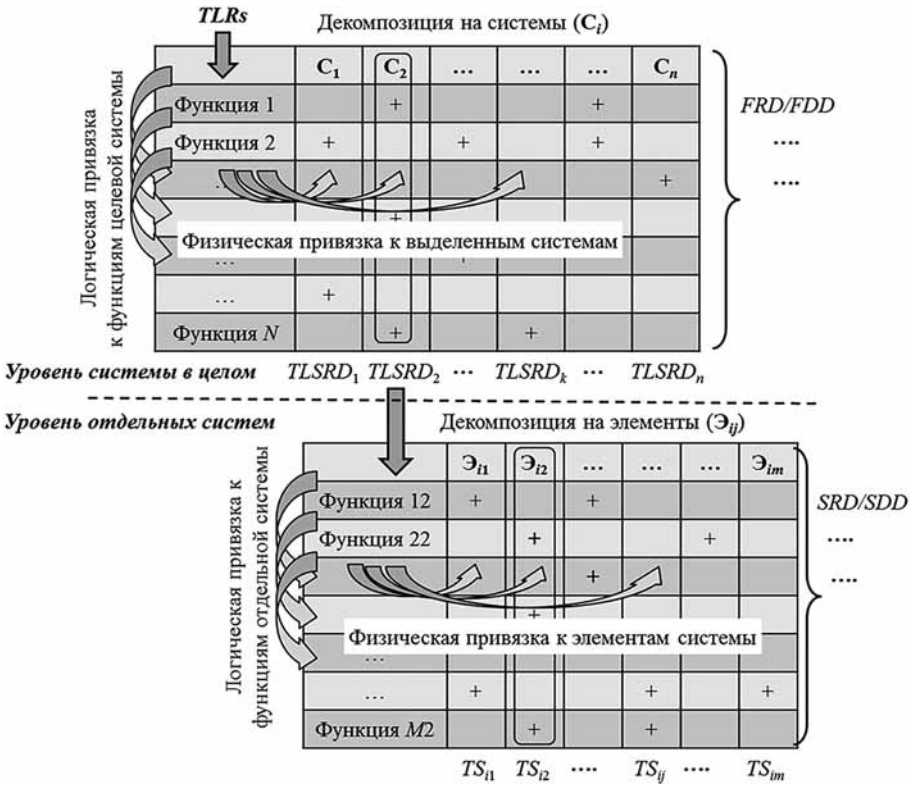

Рис. 4. Распределение требований и функций между элементами системы на различных уровнях системной иерархии

элемент удовлетворял одному или нескольким требованиям, и каждое требование было привязано по меньшей мере, к одному элементу;

- верификацию того, что все требования распределены по элементам системы.

Таким образом, при переходе от описания функционирования к описанию требований к системе и к ее составным частям в центре внимания оказывается процедура привязки этих требований и функций к элементам системы на различных уровнях системной иерархии (рис. 4). В процессе привязки выполняется надлежащая группировка функций целевой системы с их физической привязкой к системам второго иерархического уровня, а также распределение требований, предъявляемых к данным функциям. В результате формируются: спецификация функций (*function description document — FDD*); спецификация функциональных требований (*function requirement document — FRD*) и набор спецификаций высокоуровневых требований к системам, входящим в состав целевой системы (*top level system requirement document — TLSRD*). Спецификация функций *FDD* содержит сведения о системах и структурных доменах, которые в составе целевой системы участвуют в реализации функции, определенной на уровне системы в целом в соответствии с высокоуровневыми требованиями, идентифицированными и документированными на уровне программы/проекта — *TLR*. В свою очередь, спецификация функциональных требований *FRD* включает требования к специфицированным функциям, вытекающие из специфицированных требований ЗС.

Применительно к промышленной инженерии требований декомпозиция целевой системы на отдельные системы выполняется, как правило, в контексте ранее сложившихся представлений о логической архитектуре целевой системы. Далее в ее составе определяют ключевые системы, возможно, основные подсистемы, а также связи между ними. Работа с требованиями в этой ситуации строится сразу в двух направлениях "снизу вверх", т. е. от логической архитектуры к требованиям, и "сверху вниз", т. е. от логической архитектуры к физической архитектуре и элементам конфигурации. Например, в авиастроении сложившиеся представления о системах, входящих в состав воздушного судна, отражены в общепризнанном документе S1000D [25], который используется в нашей стране. В случае отсутствия исходных сведений об архитектуре целевой системы, она отдельно итерационно определяется с учетом специфицированных требований и указанных ранее архитектурных представлений и точек зрения.

Архитектура систем третьего иерархического уровня (на рис. 4 эти системы названы элементами), как правило, определяется непосредственно в процессе разработки. Результатом этой деятельности является описание архитектуры этих элементов, а также привязка функций систем второго иерархического уровня и связанных с ними требований к этим элементам (рис. 4). В итоге формируются: описание каждой системы в целом (*system description document — SDD*); спецификации требований к системам (*system requirement document — SRD*) и набор технических спецификаций элементов, входящих в состав отдельных систем (*technical specification — TS*). Спецификация *SDD* составляется для каждой из систем в составе целевой системы. Она включает описания: архитектуры системы; взаимодействия между элементами в составе системы; функций, выполняемых этой системой. Спецификация требований *SRD* содержит описание всех требований, которым должна соответствовать система для того, чтобы полностью отвечать спецификации требований к системе в целом. Техническая спецификация элемента *TS* содержит сведения, необходимые и достаточные для закупки или производства соответствующего элемента.

В случае необходимости процедура, показанная на рис. 4, может быть продолжена на четвертом уровне системной иерархии.

Разработка архитектуры и распределение требований тесно взаимосвязаны и являются итерационными процессами. В рамках каждого итерационного цикла степень конкретизации и понимания требований повышаются. Распределение системных требований между элементами аппаратного и программного обеспечения при этом становится все более корректным. Важнейшим результатом усилий по распределению требований между элементами является набор технических спецификаций, примерный состав которого описан выше.

Итоговая совокупность прослеживаемых и верифицируемых требований к характеристикам и свойствам отдельных элементов может быть основой для успешного производства системы, управления ее конфигурацией, а также всех видов проверки соответствия.

Заключение

Ключевой составляющей снижения проектных рисков в ЖЦ сложных технических систем (далее — систем) является инженерия требований. Именно она позволяет найти баланс между сложностью и гетерогенностью таких систем и возможностью поддержания необходимого уровня гарантий их качества, в том числе гарантий безопасности, обеспечения производства и сопровождения в процессе эксплуатации.

Важнейшим условием поставки на рынок качественных систем является наличие у предприятия-поставщика регламентированных и работоспособных процессов сопровождения систем на всех этапах их ЖЦ. Применительно к инженерии требований эти процессы должны обеспечивать взаимоувязанность трех видов описаний: описание совокупности требований; описание архитектуры системы; описание объектов конфигурации. За основу удобно принять стандартные технические процессы, рекомендованные в работах [13, 14], с их последующей адаптацией к условиям предприятия.

Ввиду сложности систем и большого числа участников процессов ЖЦ (с учетом поставщиков готовых решений, а также обеспечивающих и прочих процессов) неотъемлемым условием успешности эффективного использования требований является комплексная автоматизация процессов инженерии требований.

В основу такой автоматизации должны быть положены: стандартизованные на уровне поставщика целевой системы (а значит, и для его кооперации) модели процессов инженерии требований, стандартные иерархические представления систем и их элементов; шаблоны документов, формируемых на выходе типовых процессов инженерии требований (и других связанных с ними процессов ЖЦ); схемы формирования и использования требований в привязке к концепции использования системы и к другим концепциям ЖЦ, включая обеспечение четкой привязки функциональных требований к иерархической структуре создаваемой системы.

Опыт передовых предприятий показывает, что для обеспечения непрерывности перехода от контрактных документов (например, ТТЗ/ТЗ) к проекту и проектным решениям целесообразно использовать разработку совокупности обязательных спецификаций требований, включая спецификации требований заинтересованных сторон (*StRS*), спецификации требований к системе (*SyRS*), спецификации требований к ПО (*SRS*), а также опционально спецификации деловых требований (*BRS*).

Фундаментальное значение для успеха проектов создания систем имеет обеспечение и поддержание прослеживаемости требований на протяжении ЖЦ. Ключевым фактором успеха в этом случае является построение дерева функций системы, а также дерева спецификаций требований к системе и к ее элементам с непременной взаимной увязкой этих представлений между собой.

Список литературы

1. **Wiegers K., Beatty J.** Software Requirements (3rd Edition) (Developer Best Practices), Microsoft Press, 2013. 673 p.
2. **Pohl K.** Requirements Engineering: Fundamentals, Principles, and Techniques, Springer-Verlag Berlin Heidelberg, 2010. 814 p.
3. **Hull E., Jackson K., Dick J.** Requirements Engineering (4th Edition). Springer-Verlag, 2017. 239 p.
4. **Позин Б. А.** Проблемы программной инженерии на современном этапе ее развития // Программная инженерия. 2011. № 1. С. 2—6.
5. **Межгосударственный** авиационный комитет. Авиационные правила. Часть 25. Нормы летной годности самолетов транспортной категории. 2015. URL: http://www.armak-iac.org/dokumenty/aviatsionnye-pravila/
6. **Airframer:** The aerospace manufacturing directory. URL: http://www.airframer.com/default.html
7. **ISO/IEC** 15026-3:2015 Systems and software engineering. Systems and software assurance. Part 3: System integrity levels. URL: https://www.iso.org/standard/64842.html
8. **ГОСТ Р** ИСО/МЭК 15026-4-2016 Системная и программная инженерия. Гарантирование систем и программного обеспечения. Часть 4. Гарантии жизненного цикла. М.: Стандартинформ, 2016.
9. **SAE** Guidelines for Development of Civil Aircraft and Systems ARP4754A: 2010. URL: https://www.sae.org/standards/content/arp4754a/
10. **Межгосударственный** авиационный комитет. Авиационный регистр. Руководство Р-4754а по разработке воздушных судов гражданской авиации и систем. Введено в действие с 22.08.16 директивным письмом № 02-2016 Авиарегистра МАК от 22 августа 2016 г. М.: Авиарегистр МАК, 2016. 103 с.
11. **Батоврин В. К., Балашов Ю. В.** Управление конфигурацией в проектах создания сложных систем // Управление проектами и программами. 2017. № 4 (52). С. 250—263.
12. **Project** Performance International Requirements Management Tools List. PPI. 2015. URL: https://www.ppi-int.com/wp-content/uploads/2017/07/PPI-005107-5-RqtsManToolList-151008.pdf
13. **ISO/IEC/IEEE** FDIS 29148:2017 Systems and software engineering. Life cycle processes. Requirements engineering. URL: https://www.iso.org/standard/72089.html
14. **ISO/IEC/IEEE** 15288:2015 Systems and software engineering. System life cycle processes. URL: https://www.iso.org/standard/63711.html
15. **ISO/IEC/IEEE** 12207:2017 Systems and software engineering. Software life cycle processes. URL: https://www.iso.org/standard/63712.html
16. **Systems** engineering handbook. A guide for system life cycle processes and activities. Fourth edition. INCOSE-TP-2003-002-04. Hoboken, John Wiley & Sons. 2015. 305 p.
17. **American** Institute of Aeronautics and Astronautics. Guide to the Preparation of Operational Concept Documents. ANSI/AIAA G-043B—2018 URL: https://arc.aiaa.org/doi/pdf/10.2514/4.105487.001.
18. **INCOSE** Guide for Writing Requirements. INCOSE-TP-2010-006-01. Version/Revision: 1. 2012. URL: https://www.incose.org/products-and-publications/technical-publications.
19. **ГОСТ** Р 15.201—2000 Система разработки и постановки продукции на производство. Продукция производственно-технического назначения. Порядок разработки и постановки продукции на производство. М.: Стандартинформ, 2008. 17 с.
20. **ГОСТ** 34.602—89 Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. М.: Стандартинформ, 2009. 18 с.
21. **SAE** AIR 6110 Contiguous aircraft/system development process example: 2011. URL: https://www.sae.org/standards/content/air6110/
22. **ISO**/IEC/IEEE 15289:2017 Systems and software engineering. Content of life-cycle information items (documentation). URL: https://www.iso.org/standard/71950.html.
23. **ISO**/IEC/IEEE 42010:2011 Systems and software engineering. Architecture description. URL: https://www.iso.org/standard/50508.html
24. **Architecture** Viewpoint Library. URL: http://www.iso-architecture.org/viewpoints/
25. **S1000D**-B6865-01000-00 International specification for technical publications using a common source database. Issue No. 4.2. AeroSpace and Defence Industries Association of Europe — ASD. 2016. URL: http://public.s1000d.org/Pages/Home.aspx.

Requirements Engineering at a Modern Industrial Enterprise

V. K. Batovrin, batovrin@mirea.ru, MIREA — Russian Technological University, Moscow, 117454, Russian Federation, **B. A. Pozin,** bpozin@ec-leasing.ru, JSC "EC-leasing", Moscow, 117587, Russian Federation; National Research University Higher School of Economy, Moscow, 101000, Russian Federation

Corresponding author:

Batovrin Victor K., Head of Department, MIREA — Russian Technological University, Moscow, 117454, Russian Federation,
E-mail: batovrin@mirea.ru

Received on October 01, 2018
Accepted on November 01, 2018

The article discusses the key problematic issues of requirements engineering in relation to the projects implemented by industrial enterprises. The enterprise that leads the development, implementation, operation and maintenance processes throughout complex technical systems life cycle is considered. The vital need to state requirements engineering processes at a modern level within the framework of production cooperation is shown in the context of ensuring the integrity and consistency of project activities. The main problems of the absence of mature requirements engineering throughout the life cycle of a complex technical system are discussed. The estimation of the scope of such problems for the aviation industry as an example is given. It is shown that the key success factor to solve these problems is the creation of subdivisions responsible for the forming of various types of requirements and maintaining their integrity and traceability throughout the life cycle of a complex technical system, as well as complex automation of these activities. The objectives, results and content of the basic processes of requirements engineering as well as the modern standards and best practices in this area are discussed. The method for the eliciting, managing and specification of requirements during the life cycle of the technical system is proposed in that context. It is shown that it is possible to ensure two-way traceability between the most important entities: the hierarchies of functional requirements and system requirements, the system architecture and the identified configuration objects when using this method.

Keywords: *requirements engineering, requirements eliciting, requirements quality, requirements documentation, requirements traceability*

For citation:

Batovrin V. K., Pozin B. A. Requirements Engineering at a Modern Industrial Enterprise, *Programmnaya Ingeneria,* 2019, vol. 10, no. 3, pp. 114—124.

DOI: 10.17587/prin.10.114-124

References

1. **Wiegers K., Beatty J.** *Software Requirements (3rd Edition) (Developer Best Practices),* Microsoft Press, 2013. 673 p.
2. **Pohl K.** *Requirements Engineering: Fundamentals, Principles, and Techniques,* Springer-Verlag Berlin Heidelberg, 2010. 814 p.
3. **Hull E., Jackson K., Dick J.** *Requirements Engineering (4th Edition)*. Springer-Verlag, 2017, 239 p.
4. **Pozin B. A.** Problemy programmnoy inzhenerii na sovremennom etape yeye razvitiya (Software engineering problems of at the present stage of its development), *Programmnaya inzheneriya*, 2011, no. 1, pp. 2—6 (in Russian).
5. **Mezhgosudarstvennyy** aviatsionnyy komitet. *Aviatsionnyye pravila. Chast' 25. Normy letnoy godnosti samoletov transportnoy kategorii (*Aviation rules. Part 25. Airworthiness standards for aircraft of a transport category). 2015, 288 p., available at: http://www.armak-iac.org/dokumenty/aviatsionnye-pravila/ (in Russian).
6. **Airframer:** *The aerospace manufacturing directory,* available at: http://www.airframer.com/default.html
7. **ISO/IEC 15026-3:2015** Systems and software engineering. Systems and software assurance. Part 3: System integrity levels, available at: https://www.iso.org/standard/64842.html
8. **GOST R ISO / MEK 15026-4—2016** Sistemnaya i programmnaya inzheneriya. Garantirovaniye sistem i programmnogo obespecheniya. Chast' 4. Garantii zhiznennogo tsikla (Systems and software engineering. Systems and software assurance. Part 4. Assurance in the life cycle), available at: https://www.gost.ru/portal/gost/home/standarts/catalognational/ (in Russian).
9. **SAE ARP4754A:2010** Guidelines for Development of Civil Aircraft and Systems, available at: https://www.sae.org/standards/content/arp4754a/
10. **Mezhgosudarstvennyy** aviatsionnyy komitet. Aviatsionnyy registr. Rukovodstvo R-4754a po razrabotke vozdushnykh sudov i sistem (Manual R4754A on the development of aircraft and systems), Moscow, Aviaregistr MAK, 2016, 103 p. (in Russian).
11. **Batovrin V. K., Balashov Yu. V.** Upravleniye konfiguratsiyey v proyektakh sozdaniya slozhnykh sistem (Configuration management in complex systems projects), *Upravleniye proyektami i programmami*, 2017, no. (52), pp. 250—263 (in Russian).
12. **Project** Performance International Requirements Management Tools List. PPI, 2015, available at: https://www.ppi-int.com/wp-content/uploads/2017/07/PPI-005107-5-RqtsManToolList-151008.pdf (in Russian).
13. **ISO/IEC/IEEE FDIS 29148:2017** Systems and software engineering. Life cycle processes. Requirements engineering, available at: https://www.iso.org/standard/72089.html
14. **ISO/IEC/IEEE 15288:2015** Systems and software engineering. System life cycle processes, available at: https://www.iso.org/standard/63711.html
15. **ISO/IEC/IEEE 12207:2017** Systems and software engineering. Software life cycle processes, available at: https://www.iso.org/standard/63712.html
16. **Systems** engineering handbook. *A guide for system life cycle processes and activities*, Fourth edition, INCOSE-TP-2003-002-04. Hoboken, New Jersey, John Wiley & Sons, 2015, 305 p.
17. **American** Institute of Aeronautics and Astronautics. ANSI/AIAA G-043B—2018. Guide to the Preparation of Operational Concept Documents, available at: https://arc.aiaa.org/doi/pdf/10.2514/4.105487.001
18. **INCOSE** Guide for Writing Requirements. INCOSE-TP-2010-006-01. Version/Revision: 1. 2012, available at: https://www.incose.org/products-and-publications/technical-publications
19. **GOST R 15.201—2000** Sistema razrabotki i postanovki produktsii na proizvodstvo. Produktsiya proizvodstvenno-tekhnicheskogo naznacheniya. Poryadok razrabotki i postanovki produktsii na proizvodstvo. System development and delivery of products for production. (Products for industrial purposes. The order of development and delivery of products for production), Moscow, Standartinform, 2008, 17 p. (in Russian).
20. **GOST 34.602—89** Kompleks standartov na avtomatizirovannyye sistemy. Tekhnicheskoye zadaniye na sozdaniye avtomatizirovannoy sistemy. (Set of standards for automated systems. Terms of Reference for the creation of an automated system), Moscow, Standartinform, 2009, 18 p. (in Russian).
21. **SAE AIR 6110** Contiguous aircraft/system development process example: 2011, available at: https://www.sae.org/standards/content/air6110/
22. **ISO/IEC/IEEE 15289:2017** Systems and software engineering. Content of life-cycle information items (documentation), available at: https://www.iso.org/standard/71950.html.
23. **ISO/IEC/IEEE 42010:2011** Systems and software engineering. Architecture description, available at: https://www.iso.org/standard/50508.html
24. **Architecture** Viewpoint Library, available at: http://www.iso-architecture.org/viewpoints/
25. **S1000D-B6865-01000-00** International specification for technical publications using a common source database. Issue No. 4.2. Aero-Space and Defence Industries Association of Europe — ASD, 2016, available at: http://public.s1000d.org/Pages/Home.aspx.

УДК 004.93'12 DOI: 10.17587/prin.10.125-134

В. М. Староверов, канд. физ.-мат. наук, доц., e-mail: staroverovvl@yandex.ru,
Московский государственный университет имени М. В. Ломоносова

Классификация сенсорных образов с помощью тактильной информации, полученной от механорецептора*

Статья посвящена автоматическому анализу тактильных образов, полученных с помощью медицинского тактильного эндохирургического комплекса. Предложен алгоритм для классификации тактильных образов, позволяющий выявлять и классифицировать неоднородности в исследуемых тканях. Описаны результаты валидации алгоритма на базе искусственных образцов, имитирующих однородную ткань, ткань с патологическими включениями и ткань с кровеносным сосудом.

Ключевые слова: *искусственное распознавание тактильных образов, медицинский тактильный эндохирургический комплекс, библиотека сенсорных образов*

Введение

В традиционной хирургии пальпация (ощупывание тканей) играла важную роль и была обязательным этапом, например, ревизии брюшной полости [1]. С переходом к малоинвазивной хирургии получение тактильного отклика стало сложной (а в случае использования роботов практически невозможной) задачей. Для исправления ситуации рядом исследовательских групп были разработаны устройства, потенциально способные решать задачу съема и передачи тактильных данных. Насколько известно автору, единственным решением такой задачи, прошедшим сертификацию и пригодным для использования в реальных операциях, является медицинский тактильный эндохирургический комплекс (МТЭК [2]). Основная составляющая комплекса — тактильный механорецептор, оснащенный датчиками давления и передающий показания датчиков управляющему компьютеру. Полученные данные могут быть воспроизведены на специальном тактильном дисплее, а также выведены на экран.

В настоящее время для поиска патологических участков с аномальными жесткостными характеристиками хирург изучает информацию на тактильном дисплее и экране и принимает решение самостоятельно [3—5]. Отметим, что тактильное обследование может занимать несколько минут, поэтому внимание хирурга может притупиться, что приведет к пропуску патологических участков. Негативным фактором является и субъективность анализа. Как следствие, оказывается востребованной задача добавления к программному обеспечению МТЭК модуля анализа, выполняющего две основные функции:

- выявление неоднородностей, что позволит как минимум привлекать внимание хирурга к участкам, нуждающимся в более внимательном изучении, а в ряде случаев приведет к автоматическому выявлению патологических включений;
- классификация неоднородностей, что позволит, например, исключать из рассмотрения случаи "нормального" изменения жесткостных характеристик, например, наличия кровеносных сосудов.

В работе [6] авторы представили три простых алгоритма выявления неоднородностей, показавших хорошие результаты в ситуации, когда нажатие механорецептора ортогонально исследуемой ткани. В работе [7] доказано, что эти методы находятся в общем положении, т. е. ни один из них не мажорирует другой. В работе [8] эти методы были применены к нажатиям под углами, отличающимися от прямых. Оказалось, что уже при отклонении не менее 7...10° уровень ложно выявленных неоднородностей каждого из трех методов приближается к 100 %. Для исправления ситуации авторы работы [8] предложили подход, основанный на предварительном вычислении угла нажатия. Это позволило значительно поднять качество (экспериментально оцененные вероятности ошибок не превысили трех процентов). Отметим, однако, что для успешной реализации подхода требуется составление обучающей выборки для всевозможных углов нажатия, что не всегда осуществимо на практике.

В работах [8—10] описаны методы многоклассовой классификации образов, полученных с помощью

* Работа выполнена в рамках проекта Российского научного фонда № 16-11-00058 "Разработка методов и алгоритмов автоматизированного анализа медицинской тактильной информации и классификации тактильных образов".

тактильного механорецептора. Авторы работы [8] выделяли три класса: однородный образец, образец, моделирующий кровеносный сосуд, и образец, моделирующий патологическое включение. Алгоритм состоял из двух шагов: 1) определение угла нажатия (с помощью метода ближайшего соседа среди тестовой выборки нажатий, проводимых под разными углами) и 2) определение класса с использованием найденного угла (на основе метода ближайшего соседа). В работе [9] число классов равнялось шести (с содержательной точки зрения произошло расслоение "патологий" по форме и размеру), однако рассматривались только ортогональные нажатия. В качестве классификатора помимо метода ближайшего соседа использовался случайный лес. Результаты на тестовой выборке показали, что более предпочтителен метод ближайшего соседа. Отметим, что точность для некоторых "патологических" классов оказалась существенно ниже 50 %. Наконец, в работе [10] признаковое пространство из работы [9] было обогащено за счет характеристик, связанных с градиентом давления. В результате даже несмотря на рассмотрение различных углов нажатия, качество распознавания всех классов удалось поднять практически до 80 %. Отметим, что в обучающей выборке присутствовали образцы для всех используемых углов нажатия.

Таким образом, возникает необходимость исследования точности классификации в зависимости от репрезентативности покрытия углов нажатия в обучающей выборке. Кроме того, разумно учесть внутренние симметрии тактильного механорецептора. В настоящей работе предложена собственная вариация алгоритма классификации, в котором используется нестандартная мера близости в признаковом пространстве, а также совмещается метод ближайшего соседа для первичной классификации с методом голосования для принятия решения. Отметим, что данный алгоритм пригоден, в частности, и для распознавания неоднородностей.

Тактильный механорецептор и тестовая база тактильных нажатий

С содержательной точки зрения тактильный механорецептор представляет собой управляющую головку, насаженную на ручку. Диаметр управляющей головки равен 10 или 20 мм. В головке на гексагональной решетке установлено 7 (для диаметра 10 мм) или 19 (для диаметра 20 мм) датчиков давления (см., например, fig. 1, *a*, *b* [8]). Датчики передают свои показания на управляющий компьютер 100 раз в секунду. В рамках одного тактильного нажатия головка опускается на исследуемую поверхность и затем поднимается обратно.

Занумеруем сенсоры некоторым способом. В результате получим, что одна порция отправленных на компьютер показаний (назовем ее тактильным кадром) представляет собой массив $P = (p_1,..., p_{ns})$, где $ns = 7$ для прибора с головкой диаметром 10 мм и $ns = 19$ для случая 20 мм. В рамках эксперимента использовался вариант с $ns = 19$. Одно тактильное нажатие от момента начала движения головки до момента окончания движения задается последовательностью тактильных кадров $(P_1,..., P_N)$, где N — число кадров. Будем называть такое тактильное нажатие просто *нажатием*, которое, в свою очередь, состоит из *кадров*. Таким образом, каждый кадр задается массивом из *ns* значений.

Для тестирования алгоритмов классификации была расширена библиотека искусственных образов, упомянутая в работе [10]. В качестве основы использовался мягкий силикон (твердость по Шору 00-10A0) в форме прямоугольных параллелепипедов со сторонами 40, 35 и 10 мм. В некоторые образцы были добавлены жесткие включения следующих видов:

- сферический сегмент с диаметром основания 8 мм и высотой 2,4 мм, ориентированный выпуклой стороной вверх;
- сферический сегмент с диаметром основания 8 мм и высотой 2,4 мм, ориентированный выпуклой стороной вниз;
- сферический сегмент с диаметром основания 4,7 мм и высотой 1,7 мм, ориентированный выпуклой стороной вверх;
- сферический сегмент с диаметром основания 4,7 мм и высотой 1,7 мм, ориентированный выпуклой стороной вниз;
- горизонтально ориентированный участок медицинской перфузионной трубки фирмы B. Braun серии Original Perfusion Line диаметром приблизительно 2 мм, длиной 20 мм.

Таким образом, возникло шесть классов образцов (включая образец без жестких включений). Будем называть каждый класс образцов *шаблоном*, имея в виду сходный вид сенсорной информации, которую можно получить ото всех образцов одного шаблона.

Здесь следует отметить, что размер образцов и жестких включений определялся тем обстоятельством, что разрешающая способность механорецептора составляет 1...2 мм, а радиус сенсорной платформы механорецептора составляет примерно 12 мм. Поэтому размер твердых предметов внутри препарата должен колебаться от 2 до 10 мм.

Для каждого шаблона было создано не менее пяти образцов из силикона с соответствующими жесткими включениями. Образцы каждого типа исследовались тактильным механорецептором под пятью различными углами между плоскостью головки механорецептора и тестируемой плоскостью образца: 0°, 3,6°, 7,1°, 10,6° и 14°.

Для каждого типа образца и каждого угла было проведено не менее 15 нажатий, что в итоге дало не менее 75 нажатий для каждого типа и не менее 450 нажатий во всей выборке. Через некоторое время (через 20 дней) было сделано дополнительно по 10 нажатий для каждого типа и угла, чтобы внести в данные искажения, которые определяются разными условиями испытаний (разная температура, влажность и т. п.). В результате таких действий соответствующие значения стали равны 25, 125 и 750.

Тактильные кадры удобно визуализировать в виде тепловых карт. Каждому сенсору ставится в соответствие узел гексагональной решетки, изображенный в форме шестиугольника. Цвет шестиугольника кодирует давление сенсора. Зеленый цвет соответствует нулевому давлению. При увеличении давления цвет плавно переходит в синий, а далее — в красный. Дополнительно в каждом шестиугольнике проставляется числовое значение, характеризующее нормированное давление на данный сенсор. Характерные кадры, соответствующие описанным выше классам при нулевом угле, изображены на рис. 1 (см. вторую сторону обложки). Отметим, что при изменении угла нажатия представители разных классов могу стать визуально неотличимыми (см. fig. 3, *a*, *c* [8]).

Описание алгоритма классификации

Предварительными шагами алгоритма классификации являются предобработка тактильных кадров (*предотвращение зашкаливания* и *коррекции наклонного нажатия*) и формирование *библиотек сенсорных представлений шаблона* (или просто *библиотек шаблонов*) на основе обучающей выборки. Собственно, процедура классификации начинается с тиражирования кадров классифицируемого нажатия с помощью специальных *трансформаций кадров*. Затем на основе специальной *функции близости кадров* определяется ближайший к множеству растиражированных кадров из некоторого временного диапазона элемент библиотеки. Окончательное решение принимается с помощью *"голосования"* по выбранным элементам библиотеки. Далее приведено подробное описание каждого из указанных этапов процедуры.

Предобработка кадров

В рамках предобработки для каждого кадра могут применяться два метода подавления артефактов: предотвращение зашкаливания и коррекция наклонного нажатия, введенные в работе [11].

Предотвращение зашкаливания. При работе с механорецептором значения условного *сырого давления* на сенсоры механорецептора, выдаваемые прибором, преобразуются в некоторые *нормированные значения*. Для этого в процессе так называемого *тестового нажатия* механорецептором на определенный образец задается некоторое давление $P_{\max}$, больше которого, по мнению экспериментатора, наблюдаться не может. Сырое давление нормируется таким образом, чтобы нулевое давление соответствовало бы нулевому нормированному давлению, а давление $P_{\max}$ — нормированному (безразмерному) давлению, равному 255. Следует отметить, что при реальной работе с механорецептором данный процесс калибровки устройства называется процессом *создания паспорта устройства*. Создание паспорта устройства проводится независимо для каждого механорецептора в силу отличий индивидуальных параметров всех сенсоров всех механорецепторов.

В случае, если при нажатии нормированное давление на сенсор механорецептора превысит выбранное максимально допустимое значение, происходит *зашкаливание* сенсора механорецептора. Как следствие, при увеличении давления значение, которое получается с данного сенсора, ограничивается значением, равным 255.

Для борьбы с зашкаливанием используется процедура *автокоррекции силы нажатия*. Такая процедура заключается в следующей дополнительной нормировке уже нормированных значений, поступающих от механорецептора: $v_i = 255\, v'_i / \max_{i=1,\ \ldots,\ ns} v'_i$, где v'_i — нормированные значения с сенсоров для одного кадра, получаемые с механорецептора; v_i — используемые в дальнейшем в программе значения сенсоров.

Коррекция наклонного нажатия. Под *коррекцией наклонного нажатия* будем понимать преобразование, примененное к полученным значениям функции кадра, призванное некоторым образом скомпенсировать отклонение механорецептора от перпендикулярного направления к поверхности препарата при нажатии. Данное преобразование в рассматриваемом случае сводится к следующей последовательности действий:

1) методом наименьших квадратов строится линейная функция, наименее отклоняющаяся от значений в сенсорах механорецептора (с учетом их геометрического расположения); значения данной функции в точках сенсоров вычитаются из соответствующих значений функции кадра;

2) к полученной функции кадра прибавляется константа, которая вычисляется таким образом, чтобы минимальное значение исходной функции кадра не отличалось от минимального значения преобразованной функции кадра после данного шага;

3) если после предыдущих преобразований некоторые значения функции кадра окажутся больше 255, то значения функции кадра нормируются так, чтобы максимальное значение функции кадра стало бы равным 255.

Отметим, что в отдельных экспериментах те или иные методы предобработки могут отключаться.

Формирование библиотеки шаблонов

В *библиотеку шаблона* из каждого нажатия обучающей выборки заносится N_c кадров с максимальным средним квадратичным отклонением от среднего значения кадра среди всех кадров, для которых эффект зашкаливания наступил не более, чем для одного сенсора. Таким образом выбираются в определенном смысле "самые контрастные кадры" среди кадров без залипания. В работе используется $N_c = 2$.

Для полученного набора библиотек дополнительно осуществляется некоторая фильтрация, отбрасывающая заведомо неприемлемые кадры. Например, отбрасываются кадры, близкие к кадрам сразу из нескольких библиотек шаблонов.

Процедура классификации

Первый шаг в процедуре классификации — попытка учесть естественные симметрии, порождаемые тактильным механорецептором. Рассмотрение нескольких *трансформаций кадра* из тестируемого нажатия используются для компенсации угла поворота ручки механорецептора вдоль вертикальной оси между нажатием на образец из шаблона и нажатием на тестируемый образец. Отметим, что наличие в одном шаблоне нескольких образцов также дает возможность рассмотрения нескольких (случайных) углов поворота одной неоднородности. В качестве трансформаций рассматриваются движения, переводящие множество точек сенсоров кадра в это же множество. Сначала рассматриваются повороты кадра на углы, кратные 60°. Пример изображен на рис. 2.

Таким образом из одного кадра получаем шесть. Еще шесть кадров получаются любой заданной симметрией на плоскости сенсоров относительно линии, проходящей через пять сенсоров, и поворотами на углы, кратные 60°. Пример представлен на рис. 3.

Теперь для каждого тестируемого кадра мы получаем набор из 12 преобразованных кадров, для которых и проводится процедура сравнения с кадрами из библиотек шаблонов с помощью функции близости кадров. Отметим, что все описанные выше преобразования в реальности проводятся простой перенумерацией элементов массива нажатий. Поэтому не имеет смысла для увеличения быстродействия вместе с каждым кадром из библиотеки шаблона хранить кадры всех его трансформаций.

Функция близости кадров. Будем обозначать нормированные значения сенсоров в кадре до процедуры коррекции наклонного нажатия v_i (i = 1,..., ns), где ns — число сенсоров на механорецепторе. В экспериментах используются механорецепторы, для которых

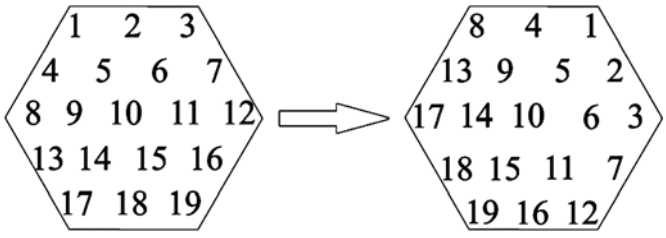

Рис. 2. Поворот кадра на 60°

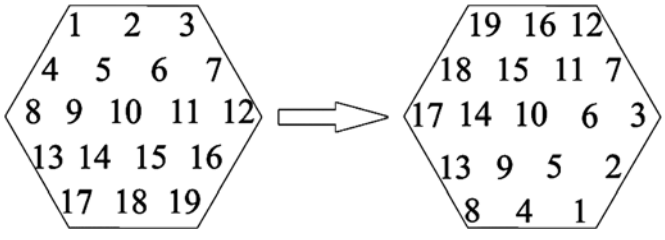

Рис. 3. Симметрия относительно оси из левого нижнего угла в верхний правый и поворот кадра на 60°

ns = 19, v — имя кадра. Для кадров v и w функция близости вводится как

$$L(v, w) = \min_{\alpha\in[\alpha_0,\,\alpha_1]} \frac{\sqrt{\sum_1^{ns}(\alpha w_i' - v_i)^2}}{\sqrt{\sum_1^{ns}(v_i')^2}},$$

где w' и v' — нормированные значения массивов кадров w и v после коррекции наклонного нажатия; α_0, α_1 — некоторые вещественные числа, являющиеся параметрами алгоритма.

Наряду с данной функцией близости для сравнения будет рассмотрена функция близости без коррекции наклонного нажатия:

$$L_0(v, w) = \min_{\alpha\in[\alpha_0,\,\alpha_1]} \frac{\sqrt{\sum_1^{ns}(\alpha w_i - v_i)^2}}{\sqrt{\sum_1^{ns}(v_i)^2}}.$$

Отметим несимметричность вхождения кадров в формулу. Этот факт определяется тем, что при проведении процесса распознавания в качестве первого кадра в паре (v_i, w_i) обычно используется кадр из библиотеки шаблонов, а в качестве второго — тестируемый кадр. Введение параметра α позволяет в заданных пределах подстраивать силу нажатия, которая используется в момент получения данных тестируемого кадра, к силе нажатия кадра из библиотеки шаблонов. Это в свою очередь позволяет использовать при сравнении большое число кадров из одного нажатия. Нормирующий множитель в знаменателе позволяет унифицировать все кадры из библиотеки по силе нажатия. Такой множитель устраняет также зависимость функции от числа сенсоров.

Теперь можно переходить, собственно, к распознаванию. Итак, при осуществлении нажатия на тестируемый образец для каждого кадра (будем его называть *тестируемым кадром*), получающегося при нажатии, рассматриваются описанные *трансформации кадра*. Каждый из получающихся после трансформации кадров сравнивается со всеми кадрами из библиотек шаблонов с помощью функции близости кадров и находится пара (трансформированный кадр, кадр из библиотек шаблонов) с наименьшим значением функции близости. Изо всех найденных пар (трансформированный кадр, кадр из библиотек шаблонов) за последний интервал времени (например, за одну секунду) ищется *выделенная пара* с наименьшим значением функции близости (кадр из библиотеки шаблонов из данной пары будем называть *выделенным кадром*). В простейшем случае рассматривается только один кадр, получающийся при нажатии на тестируемый образец, и выделенный кадр из библиотеки, ему соответствующий. Именно этот случай описывается далее в работе, хотя при

практическом использовании механорецептора принято искать выделенный кадр для набора кадров, полученных на интервале 0,5...1 с. Будем называть этот выделенный кадр *идентифицирующим кадром*, если значение функции близости между найденным выделенным кадром и трансформированным тестируемым кадром оказывается меньше заданного порогового значения. Отметим, что пороги являются параметрами алгоритма и, вообще говоря, могут отличаться для разных классов. В этом случае считается, что тестируемый кадр соответствует шаблону, задающему библиотеку шаблонов, к которой принадлежит идентифицирующий кадр.

Наконец, следует описать принципы "*голосования*", используемые при отнесении всего нажатия к определенному шаблону. Пусть в созданной библиотеке сенсорных образов находится N_l библиотек шаблонов, соответствующих различным исходным образцам (шаблонам). В настоящей работе рассматриваются случаи $N_l = 6$ (классификация неоднородностей) и $N_l = 2$ (выявление неоднородностей).

Будем считать, что данное нажатие *задает соответствие с k-м шаблоном* (*соответствует k-му шаблону*), если число идентифицирующих кадров, полученных при нажатии, принадлежащих *k*-й библиотеке шаблонов, больше числа идентифицирующих кадров, суммарно принадлежащих всем остальным библиотекам шаблонов.

Будем считать, что набор нажатий *задает суммарное соответствие с k-м шаблоном* (*суммарно соответствует k-му шаблону*), если число идентифицирующих кадров, суммарно полученных при всех данных нажатиях, принадлежащих *k*-й библиотеке шаблонов, строго больше числа идентифицирующих кадров, принадлежащих каждой иной библиотеке шаблонов. Наличие верного суммарного соответствия набора нажатий фактически свидетельствует о том, что по информации от всего набора нажатий можно приемлемо классифицировать исследуемый шаблон (с вероятностью > 50 %).

Логично считать, что описанные соответствия могут быть либо *верными* (если соответствующий *k*-й шаблон действительно является тестируемым шаблоном), либо *неверными*.

Выбор значений параметров алгоритма

Основными настраиваемыми параметрами алгоритма являются пороговые значения, увеличивая (уменьшая) которые можно увеличивать (уменьшать) вероятность соответствия выбранных кадров соответствующей исследуемому шаблону библиотеке. Увеличение *k*-го порогового значения увеличивает вероятность соответствия любого нажатия с *k*-й библиотекой, поэтому нельзя подбирать данные значения последовательно. С помощью ручного подбора всего набора пороговых значений был достигнут высокий уровень правильного распознавания шаблонов. Подбор пороговых значений осуществлялся с помощью процедуры кросс-валидации на подмножестве тестовых нажатий, соответствующих нулевому углу между головкой тактильного механорецептора и анализируемым образцом.

Описание экспериментов

Как было отмечено ранее, в исследовании рассматривались шаблоны следующего вида (далее по тексту они будут следовать в указанном порядке):

1) сферический сегмент с диаметром основания 8 мм и высотой 2,4 мм, ориентированный выпуклой стороной вверх;

2) сферический сегмент с диаметром основания 8 мм и высотой 2,4 мм, ориентированный выпуклой стороной вниз;

3) силиконовый параллелепипед без неоднородности;

4) сферический сегмент с диаметром основания 4,7 мм и высотой 1,7 мм, ориентированный выпуклой стороной вверх;

5) сферический сегмент с диаметром основания 4,7 мм и высотой 1,7 мм, ориентированный выпуклой стороной вниз;

6) горизонтально ориентированный участок медицинской перфузионной трубки диаметром 2 мм и длиной 20 мм.

При подготовке настоящей статьи работа с нажатиями происходила в следующем виде: сначала все нажатия были покадрово записаны в соответствующие файлы, только после этого рассчитывались различные статистики по полученным данным.

Для каждого шаблона были созданы две библиотеки шаблона описанным выше способом: библиотека с использованием только строго перпендикулярных нажатий к поверхности шаблона (будем называть эту библиотеку *0-библиотекой*) и библиотека с использованием всех нажатий на все образцы всех шаблонов, включая наклонные (будем называть эту библиотеку *1-библиотекой*). Исходя из приведенных выше числовых значений получаем, что каждая 0-библиотека создавалась из 25 нажатий, кроме третьей 0-библиотеки, для которой было проведено 40 нажатий, а 1-библиотека создавалась из 125 нажатий, кроме третьей 1-библиотеки, для которой было проведено 200 нажатий. Напомним, для каждого нажатия в библиотеку заносилось 2 наиболее "контрастных" кадра. Например, кадры 0-библиотеки пластиковой трубочки изображены на рис. 4 (см. вторую сторону обложки).

Изображения кадров для всех библиотек можно найти по ссылке *http://sensor.stargeo.ru/art-2018-11-26.pdf*

По ссылке *http://sensor.stargeo.ru/ArtData/arr.html* можно найти все значения массивов кадров нажатий для каждого шаблона для всех описанных выше углов. Этих данных достаточно, чтобы самостоятельно воспроизвести все расчеты, результаты которых представлены в настоящей статье.

Оценка качества распознавания на основе 0-библиотеки шаблонов

Описанный выше подход позволил осуществлять кросс-валидацию при поиске идентифицирующих

кадров в 0-библиотеке для нулевого угла наклона в следующем смысле: для каждого нажатия идентифицирующие кадры искались по всем 0-библиотекам сенсорных образов, кроме кадров, взятых из исследуемого нажатия. Отметим, что при поиске идентифицирующих кадров по всем кадрам всех библиотек прогнозируемо получается 100 %-верная идентификация нажатий во всех смыслах (проведенные тестовые расчеты подтверждают это утверждение).

В работе *http://sensor.stargeo.ru/art-2018-11-26.pdf* указано количество найденных идентифицирующих кадров в каждой из шести 0-библиотек для каждого отдельного нажатия. Приведем здесь лишь сводную табл. 1, в которой для нажатий на каждый шаблон (одна строка задает нажатия на один шаблон) укажем суммарное число идентифицирующих кадров для различных библиотек шаблонов (один столбец соответствует одной библиотеке нажатий) для случая нулевого угла наклона.

Пороговые значения для выбора идентифицирующих кадров (по которым определяется соответствие тестируемого кадра *k*-й библиотеке) в данной работе выбирались вручную по нажатиям с нулевым углом отклонения от перпендикуляра к поверхности образца. На самом деле, эти значения не одинаковы: 1) 0,07; 2) 0,06; 3) 0,1; 4) 0,04; 5) 0,05; 6) 0,09. Для нулевого угла отклонения наборы нажатий дают верные соответствия с тестируемыми шаблонами. При этом, не все отдельные нажатия соответствуют шаблонам. Как следует из данных табл. 2, число верных соответствий для заданных библиотек при нулевом угле нажатия (при полном количестве нажатий в каждой библиотеке, равно 25, кроме третьей библиотеки с полным количеством нажатий, равным 40) следующее: 1) 21; 2) 24; 3) 40; 4) 18; 5) 20; 6) 21. Эти данные свидетельствуют о существенной разделенности сенсорных образов различных образцов, с точки зрения функции близости.

Оценим качество распознавания шаблонов при различных углах наклона сначала по информации по каждому отдельному нажатию, а потом — по всем нажатиям на один образец в целом на основе построенных 0-библиотек шаблонов и выбранных выше пороговых значений.

В табл. 2 приведено число полных нажатий и число верных соответствий отдельных нажатий, в зависимости от угла наклона нажатия и номера образца.

Данные табл. 2 указывают, что с точки зрения соответствий отдельных нажатий для угла наклона механорецептора 3,5° правильно классифицируются нажатия на образцы из всех шаблонов, кроме четвертого. На самом деле этот факт не удивителен, потому что даже визуально легко спутать сенсорные образы от полусфер диаметром 11 и 6 мм. То есть из данной таблицы можно сделать следующий вывод: отдельные нажатия приемлемо классифицируют все рассматриваемые шаблоны, кроме различия полусфер диаметром 11 и 6 мм, при угле наклона механорецептора не более 3°.

Таблица 1

Число идентифицирующих кадров 0-библиотек для *k*-го шаблона. Угол к нормали = 0°

Номер шаблона *k*	Библиотека					
	1	2	3	4	5	6
1	73	1	0	13	0	0
2	0	64	0	0	1	0
3	0	0	284	0	0	0
4	19	0	0	58	8	0
5	2	6	0	2	63	0
6	0	0	0	0	0	104

Таблица 2

Число верных соответствий отдельных нажатий и число полных нажатий для каждого из шести шаблонов (для 0-библиотек)

Угол наклона, °	Шаблон					
	1	2	3	4	5	6
0	21	24	40	18	20	21
3,5	15	13	40	8	19	22
7	17	10	37	15	10	13
10,5	11	6	8	11	7	7
14	24	0	0	9	1	6
Всего нажатий	25	25	40	25	25	25

В табл. 3 приведено следующее: число верных идентифицирующих кадров для каждой группы нажатий; максимальное число неверных идентифицирующих кадров из одной библиотеки шаблонов из всех нажатий на один образец; полное число идентифицирующих кадров для каждой группы нажатий в зависимости от угла нажатия (строка соответствует заданному углу наклона нажатия) и номера шаблона (столбец соответствует одному шаблону). Объясним более подробно значения полей с данными в табл. 3. Например, тройка чисел 73/13/87 для шаблона 1 и нулевого угла нажатия получена из следующих полных данных: среди всех 25 нажатий на первый шаблон при нулевом угле нажатия было идентифицировано 73 кадра, соответствующих первой библиотеке, 1 кадр, соответствующий второй библиотеке, 0 кадров, соответствующих третьей библиотеке, 13, соответствующих четвертой библиотеке, 0, соответствующих пятой библиотеке, 0, соответствующих шестой библиотеке. В таблицу попало первое число 73, т. е. число верных соответствий кадров; 13, т. е. максимальное число неверных соответствий среди

Таблица 3

Число верных/максимально неверных/всего соответствий кадров для 0-библиотек суммарно для всех нажатий на образцы каждого шаблона

Угол наклона, °	Шаблон					
	1	2	3	4	5	6
0	73/13/87	64/1/65	284/0/284	58/19/85	63/6/73	104/0/104
3,5	51/22/80	40/6/48	232/0/232	48/38/108	64/12/82	70/0/70
7	57/13/84	16/2/20	195/0/195	73/31/111	29/18/51	23/0/23
10,5	42/28/73	11/0/11	8/0/8	55/31/103	33/23/59	17/0/17
14	96/7/104	0/1/1	0/0/0	63/111/176	3/57/60	7/1/8

всех остальных библиотек; 87, т. е. полное число идентифицированных кадров.

Данные табл. 3 свидетельствуют о том, что с точки зрения информации, полученной от данных наборов нажатий (т. е. с точки зрения суммарных соответствий всех кадров из набора нажатий), для углов наклона механорецептора $\leqslant 10^\circ$ все шаблоны классифицируются правильно. При больших углах информация от групп нажатий механорецептора становится недостоверной. Таким образом, можно сделать следующий вывод: рассмотренные наборы нажатий позволяют приемлемо классифицировать все созданные шаблоны при угле наклона механорецептора не более 10°. Следует отметить, что в работе [8] уровень распознавания на углах наклона механорецептора около 10° становится критически плохим, и без создания обучающей выборки на наклонных нажатиях подход, предложенный в работе [8], по качеству распознавания существенно уступает описанному в настоящей статье подходу.

Отметим, что соотношение первых двух чисел в каждой ячейке таблицы (число верных идентифицирующих кадров/максимальное число неверных идентифицирующих кадров из всех библиотек) характеризует качество классификации для данного угла данного шаблона. Например, если это соотношение близко к 1, то это означает, что вероятность правильной классификации не превосходит 50 %.

Оценка качества распознавания на основе 1-библиотеки шаблонов

Расширение обучающей выборки на нажатия, проводимые под всеми рассматриваемыми углами, прогнозируемо существенно улучшает качество распознавания в целом. При этом, естественно, остается актуальным вопрос о создании такой выборки в реальных (не лабораторных) условиях. Для оценки качества распознавания на основе 1-библиотек приведем таблицы, аналогичные таблицам из предыдущего подраздела. При тестировании распознавания нажатий на основе 1-библиотек использовались те же самые пороговые значения, что и для тестирования 0-библиотек.

В табл. 4 приведено число верных соответствий отдельных нажатий в зависимости от угла наклона нажатия и номера шаблона (полное число нажатий равно 25, кроме третьего шаблона, на который нажимали 40 раз).

Данные, представленные в табл. 4, показывают существенное улучшение качества распознавания в целом по сравнению со случаем использования 0-библиотек. Плохое качество распознавания третьего образца на углах более 10° свидетельствует о соответствующем качестве распознавания образца без жестких вкраплений.

В табл. 5 приведено следующее: число верных идентифицирующих кадров для каждой группы нажатий; число неверных идентифицирующих кадров, представленных в одной библиотеке шаблонов, из всех нажатий на один образец; полное число идентифицирующих кадров для каждой группы нажатий, в зависимости от угла нажатия и номера шаблона. Напомним, что теперь используем библиотеку, полученную по нажатиям с различными углами нажатия. Данная таблица построена по тому же принципу, что и табл. 3. Например, тройка чисел 69/26/101 для шаблона 1 и нулевого угла нажатия получена из следующих полных данных: среди всех 25 нажатий на первый шаблон при нулевом угле нажатия было

Таблица 4

Число верных соответствий нажатий и общее число нажатий для 1-библиотек

Угол наклона, °	Шаблон					
	1	2	3	4	5	6
0	17	24	40	22	21	24
3,5	21	21	38	23	24	25
7	18	20	36	14	17	25
10,5	17	19	1	19	18	23
14	23	8	0	20	13	21
Всего нажатий	25	25	40	25	25	25

Таблица 5

Число верных/максимально неверных/всего соответствий кадров для 1-библиотек для наборов нажатий

Угол наклона, °	Шаблон					
	1	2	3	4	5	6
0	69/26/101	67/2/69	246/0/246	84/13/110	70/11/85	117/0/117
3,5	84/28/120	66/1/68	186/0/186	106/19/141	89/5/99	123/1/124
7	75/28/123	33/2/37	152/0/152	96/31/140	55/14/79	107/2/109
10,5	90/36/131	42/2/45	1/0/1	124/25/169	62/20/87	81/0/81
14	108/16/125	13/6/21	0/0/0	156/64/225	60/14/75	65/1/66

идентифицировано 69 кадров, соответствующих первой библиотеке, 1 кадр, соответствующий второй библиотеке; 0 кадров, соответствующих третьей библиотеке, 26, соответствующих четвертой библиотеке; 5, соответствующих пятой библиотеке; 0, соответствующих шестой библиотеке. В таблицу попало первое число 69, т. е. число верных соответствий кадров; 26, т. е. максимальное число неверных соответствий среди всех остальных библиотек; 101, т. е. полное число идентифицированных кадров.

Табл. 5 сохраняет тенденцию к улучшению распознавания при переходе от 0-библиотек к 1-библиотекам.

Обнаружение неоднородностей общего вида

Рассмотрим задачу поиска неоднородностей общего вида с помощью механорецептора на основе описанного выше подхода. При решении этой задачи будем рассматривать только нажатия механорецептором под прямым углом к поверхности образца. Под неоднородностями здесь имеем в виду неоднородности по размеру и по глубине залегания похожими на неоднородности, которые используются при создании представленных в настоящей статье библиотек сенсорных образов. В исследовании, результаты которого рассматриваются в настоящей статье, использованы семь различных шаблонов: силиконовые параллелепипеды без жестких вкраплений толщиной 7 и 15 мм, а также пять ранее рассмотренных силиконовых параллелепипедов с жесткими вкраплениями:

1) сферический сегмент с диаметром основания 8 мм и высотой 2,4 мм, ориентированный выпуклой стороной вверх;

2) сферический сегмент с диаметром основания 8 мм и высотой 2,4 мм, ориентированный выпуклой стороной вниз;

3) сферический сегмент с диаметром основания 4,7 мм и высотой 1,7 мм, ориентированный выпуклой стороной вверх;

4) сферический сегмент с диаметром основания 4,7 мм и высотой 1,7 мм, ориентированный выпуклой стороной вниз;

5) горизонтально ориентированный участок медицинской перфузионной трубки диаметром 2 мм и длиной 20 мм.

В качестве цели рассматриваем классификацию получаемых сенсорных образов на два класса: шаблон без неоднородностей и шаблон с неоднородностями. Для этого созданы две библиотеки сенсорных образов. Для создания первой библиотеки использованы шаблоны без неоднородностей (40 силиконовых параллелепипедов без жестких вкраплений толщиной 7 мм), а для второй — все шаблоны с неоднородностями, кроме образцов с неоднородностями типа $k = 1$ (всего 100 образцов). Принцип создания библиотек сенсорных образов описан выше. Качество классификации будем тестировать на образцах из двух шаблонов вида: силиконовый параллелепипед без жестких вкраплений толщиной 15 мм (25 образцов) и силиконовый параллелепипед с жестким вкраплением вида $k = 1$. Отметим, что образцы, на основе которых создаются библиотеки, и тестируемые образцы не совпадают. Будем называть данное тестирование *экспериментом номер* $k = 1$. Аналогично рассмотрим эксперименты с номерами $k = 2$, 3, 4, 5. Таким образом, имеем пять экспериментов. В табл. 6 для каждого эксперимента приведено число образцов и число верных соответствий для каждого тестируемого шаблона. Пороговые значения для всех экспериментов — 0,3 и 0,12 соответственно.

Таблица 6

Обнаружение неоднородностей в двух шаблонах (число верных соответствий для нажатий) для экспериментов с номером *k*. Угол к нормали = 0°

k	Шаблон без неоднородностей		Шаблон с неоднородностями	
	Верных соответствий	Всего соответствий	Верных соответствий	Всего соответствий
1	24	25	25	25
2	24	25	25	25
3	24	25	25	25
4	24	25	25	25
5	24	25	17	25

Таблица 7

Обнаружение неоднородностей в двух шаблонах (число верно/неверно классифицированных кадров) для экспериментов с номером *k*. Угол к нормали = 0°

k	Шаблон без неоднородностей			Шаблон с неоднородностями		
	Верно классифицировано кадров	Неверно классифицировано кадров	Всего кадров	Верно классифицировано кадров	Неверно классифицировано кадров	Всего кадров
1	176	5	181	158	0	158
2	173	8	181	127	4	131
3	177	0	177	142	1	143
4	176	5	181	142	4	146
5	176	5	181	75	7	82

Данные табл. 6 указывают на хорошее качество классификации в смысле соответствий нажатий.

Суммарные число верно классифицированных кадров, неверно классифицированных кадров, полное число классифицированных кадров для пяти описанных выше экспериментов отличаются несущественно, о чем свидетельствуют данные табл. 7.

Приведенные числа доказывают хорошее качество классификации в смысле суммарных соответствий наборов нажатий.

Приведенные выше результаты служат убедительным подтверждением качества построенных библиотек сенсорных образов, использующихся для классификации образцов на образцы без неоднородностей и образцы с неоднородностями. При сравнении результатов с результатами работы [6] следует отметить, что в настоящей работе рассмотрен более широкий класс неоднородностей и качество распознавания наличия неоднородности в настоящей работе существенно выше.

Заключение

Описана процедура классификации набора сенсорных образов, которые получаются с помощью механорецептора на основе обучающей выборки по фиксированному набору образцов. С использованием данной методики строились различные библиотеки сенсорных образов для классификации сенсорных образов, полученных от шести различных образцов, представляющих собой силиконовые параллелепипеды с различными жесткими вкраплениями. В работе обоснованы ограничения на углы наклона механорецептора к поверхности образца, для которых использованный алгоритм классификации имеет приемлемое качество. Рассмотрено решение задачи классификации получаемых сенсорных образов на два класса: на образы шаблонов без жестких вкраплений и с жесткими вкраплениями. Показано, что описанная методика также может успешно применяться для решения данной задачи. Сравнение полученных результатов с результатами, представленными в работах [6, 8], показывает явные преимущества предложенной методики для решения поставленной задачи.

Список литературы

1. **Konstantinova J., Li M., Mehra G., Dasgupta P., Althoefer K., Nanayakkara T.** Behaviorial characteristics of manual palpation to localize hasd modules in soft tissues// IEEE Trans Biomed Eng, 2014, N. 61 (6). P. 1651—1659.
2. **Barmin V., Sadovnichy V., Sokolov M., Pikin O., Amiraliev A.** An original device for intraoperative detection of small indeterminate nodules // Eur. J. Cardiothorac Surg. 2014. Vol. 46. P. 1027—1031.
3. **Solodova R. F., Galatenko V. V., Nakashidze E. R., Andreytsev I. L., Galatenko A. V., Senchik D. K., Staroverov V. M., Podolskii V. E., Sokolov M. E., Sadovnichy V. A.** Instrumental tactile diagnostics in robot-assisted surgery // Med. Devices (Auckl). 2016. Vol. 9. P. 377—382.
4. **Sokolov M., Solodova R., Galatenko V., Staroverov V., Nakashidze E.** Tactile diagnostics in robotic surgery // European Journal of Surgical Oncology. 2016. Vol. 42 (9). P. 73—73.
5. **Solodova R. F., Galatenko V. V., Nakashidze E. R., Shapovalyants S. G., Andreytsev I. L., Sokolov M. E., Podolskii V. E.** Instrumental mechanoreceptoric palpation in gastrointestinal surgery // Minimally invasive surgery. 2017. Vol. 2017. P. 1—6.
6. **Solodova R. F., Staroverov V. M., Galatenko V. V., Galatenko A. V., Solodov E. V., Antonov A. P., Budanov V. M., Sokolov M. E., Sadovnichy V. A.** Automated detection of heterogeneity in medical tactile images // Stud. Health Technol. Inform. 2016. Vol. 220. P. 383—389.
7. **Рахматулин Я. И., Рухович Д. В.** Сравнение автоматических методов обнаружения неоднородностей в тактильных образах // Интеллектуальные системы. Теория и приложения. 2016. Т. 20, № 3. С. 175—179.
8. **Alexandrov D. E., Nersisyan S. A.** Identification of contact angle and heterogeneity detection in tactile images" // International journal of biology and biomedical engineering. 2018. Vol. 12. P. 186—191.
9. **Nersisyan S. A., Rakhmatulin Y. I.** Pattern recognition in low-resolution instrumental tactile imaging" // International Journal of Circuits, Systems and Signal Processing. 2017. Vol. 11. P. 306—313.
10. **Nersisyan S. A., Staroverov V. M.** Gradient estimation for hexagonal grids and its application to classification of instrumentally registered tactile images // WSEAS transactions on applied and theoretical mechanics. 2018. Vol. 13, N. 13. P. 123—129.
11. **Staroverov V. M., Galatenko V. V., Zykova T. V., Rakhmatulin Y. I., Rukhovich D. V., Podol'skii V. E.** Automated real-time correction of intraoperative medical tactile images: sensitivity adjustment and suppression of contact angle artifact // Applied Mathematical Sciences. 2016. N. 10 (57). P. 2831—2842.

Classification of Sensory Images using Tactile Information Obtained from Mechanoreceptor

V. M. Staroverov, staroverovvl@yandex.ru, M. V. Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Moscow, 119991, Russian Federation

Corresponding author:

Staroverov Vladimir M., Associate Professor, M. V. Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Moscow, 119991, Russian Federation,
E-mail: staroverovvl@yandex.ru

Received on November 20, 2018
Accepted on December 11, 2018

The article is devoted to the automatic analysis of tactile images obtained with the help of Medical Tactile Endosurgical Complex (MTEC). The author proposes an algorithm for tactile samples classification that can identify and classify the heterogeneities in the tissues studied. To test the algorithm, the author creates a library of sensor heterogeneities, which consists of silicone parallelepipeds with solid inclusions. On the basis of this library, test samples of clicks was made. In the article the decision on belonging of the tested sample to one of the pre-defined classes is based on the proximity function between the given sample and the samples from the pre-defined classes. The final decision on the corresponding of the sample to the predefined class is made on the basis of some kind of "voting" method. Testing of the algorithm is carried out on the solution of two applied problems: 1) determination of belonging of the tested sample to one of the six types of heterogeneities; 2) determination of the presence of the inhomogeneity of the undefined type in the tested sample. The article deals with heterogeneities based on artificial samples simulating homogeneous tissue, tissue with pathologic inclusions and tissue with a blood vessel. In the solving the problems the method of cross-validation is actively used in the article. Detailed results of calculations posted on the Internet at the link provided with the article.

Keywords: *Artificial tactile sensing, Medical Tactile Endosurgical Complex, library of sensory images*

Acknowledgements: *The research was supported by the Russian Science Foundation (project 16-11-00058 "The development of methods and algorithms for automated analysis of medical tactile information and classification of tactile images")*

For citation:

Staroverov V. M. Classification of Sensory Images using Tactile Information Obtained from Mechanoreceptor, *Programmnaya Ingeneria,* 2019, vol. 10, no 3, pp. 125—134.

DOI: 10.17587/prin.10.125-134

References

1. **Konstantinova J., Li M., Mehra G., Dasgupta P., Althoefer K., Nanayakkara T.** Behaviorial characteristics of manual palpation to localize hasd modules in soft tissues, *IEEE Trans Biomed Eng.,* 2014, no. 61 (6), pp. 1651—1659.

2. **Barmin V., Sadovnichy V., Sokolov M., Pikin O., Amiraliev A.** An original device for in-traoperative detection of small indeterminate nodules, *Eur. J. Cardiothorac Surg.*, 2014, vol. 46, pp. 1027—1031.

3. **Solodova R. F., Galatenko V. V., Nakashidze E. R., Andreytsev I. L., Galatenko A. V., Senchik D. K., Staroverov V. M., Podolskii V. E., Sokolov M. E., and Sadovnichy V. A.** Instrumental tactile diagnostics in robot-assisted surgery, *Med. Devices (Auckl)*, 2016, vol. 9, pp. 377—382.

4. **Sokolov M., Solodova R., Galatenko V., Staroverov V., Nakashidze E.** Tactile diagnostics in robotic surgery, *European Journal of Surgical Oncology*, 2016, vol. 42 (9), pp. 73—73.

5. **Solodova R. F., Galatenko V. V., Nakashidze E. R., Shapovalyants S. G., Andreytsev I. L., Sokolov M. E., Podolskii V. E.** Instrumental mechanoreceptoric palpation in gastrointestinal surgery, *Minimally invasive surgery*, 2017, vol. 2017, pp. 1—6.

6. **Solodova R. F., Staroverov V. M., Galatenko V. V., Galatenko A. V., Solodov E. V., Antonov A. P., Budanov V. M., Sokolov M. E., Sadovnichy V. A.** Automated detection of heterogeneity in medical tactile images, *Stud. Health Technol. Inform.*, 2016, vol. 220, pp. 383—389.

7. **Rakhmatulin Ja. I., Rukhovich D. V.** Sravnenie avtomaticheskih metodov obnarugenia neodnorodnostej v taktilnyh obrazah (Comparison of automatic methods for detecting inhomogeneities in tactile images), *Intelligent system. Theory and applications*, 2016, vol. 20 (3), pp. 175—179 (in Russian).

8. **Alexandrov D. E., Nersisyan S. A.** Identification of contact angle and heterogeneity detection in tactile images, *International Journal of Biology and Biomedical Engineering*, 2018, vol. 12, pp. 186—191.

9. **Nersisyan S. A., Rakhmatulin Y. I.** Pattern recognition in low-resolution instrumental tactile imaging, *International Journal of Circuits, Systems and Signal Processing*, 2017, vol. 11, pp. 306—313.

10. **Nersisyan S. A., Staroverov V. M.** Gradient estimation for hexagonal grids and its application to classification of instrumentally registered tactile images, *WSEAS transactions on applied and theoretical mechanics*, 2018, vol. 13, no. 13, pp. 123—129.

11. **Staroverov V. M., Galatenko V. V., Zykova T. V., Rakhmatulin Y. I., Rukhovich D. V., Podol'skii V. E.** Automated real-time correction of intraoperative medical tactile images: sensitivity adjustment and suppression of contact angle artifact, *Applied Mathematical Sciences*, 2016, no. 10 (57), pp. 2831—2842.

УДК 514.7 DOI: 10.17587/prin.10.135-144

Н. Н. Светушков, канд. техн. наук, доц., e-mail: svetushkov@mai.ru,
Московский авиационный институт (национальный исследовательский университет)

Создание двумерных геометрических объектов на основе универсальных структурных элементов — кластерных точек

Рассмотрен новый подход к созданию сложных моделей — двумерных геометрических объектов. В основе такого подхода лежат топологические характеристики модели, которые задаются пользователем в отдельных выделенных узлах — кластерных точках, что обеспечивает необходимую гибкость описания и позволяет значительно изменять внешнее представление объекта. Кластерное представление имеет достаточную степень универсальности, позволяя формировать довольно сложные геометрические модели. При этом важной особенностью такого параметрического представления является возможность простыми средствами добавлять необходимые геометрические характеристики и изменять существующие. Введенное в работе базовое понятие кластерной точки основано на возможности задания набора параметров для локального описания замкнутой геометрической области, окружающей данную точку. Объединение конечного набора этих областей позволяет создавать новый параметризованный геометрический объект как единое целое — МОК-объект (объект, построенный по модели объединенного кластера). На простейших примерах показана применимость разрабатываемого подхода при проектировании технических изделий, в частности, с его помощью возможно нарисовать профиль крыла или колесо со спицами. Представлены скриншоты довольно экзотических фигур, для создания которых другими средствами потребовались бы значительные усилия. В заключении сделан вывод о возможностях широкого практического применения кластерного подхода.

Ключевые слова: *кластерные модели, геометрическое описание, топология, алгоритмические процедуры, визуализация, программное обеспечение*

Введение

Методам геометрического моделирования для двумерного и трехмерного случаев в литературе уделяется достаточно много внимания [1—9]. Связано это в первую очередь с возможностями разработки программного обеспечения и использования специальных возможностей для формирования сложных геометрических фигур [2]. Несмотря на то, что существует большое число программных средств [4, 5, 7], предназначенных для геометрического моделирования, остаются вопросы, связанные с формированием сложных геометрических объектов, которые состоят из кусочно-гладких кривых, определенным образом сопряженных в точках излома, например, при разработке геометрических форм в автомобильной промышленности. В этом случае часто возникают задачи, связанные с необходимостью или немного модифицировать объект, или добавить в него некоторые детали и т. п. При проектировании такого рода форм пользователь может столкнуться с ситуацией, когда геометрический объект необходимо создавать заново. Например, если в модели была использована соединительная линия в виде прямой, то для ее изменения в некоторой части в виде плавного изгиба заданного радиуса придется создавать три взаимно сопряженные соединительные линии, одна из которых является окружностью. Таким образом, если учесть, что двумерный геометрический объект фактически определяется своей граничной линией, появляется необходимость в более гибком параметрическом описании этой линии. Целью этого нового описания является возможность предоставить пользователю более естественный способ внесения различных коррективов в формируемый объект, при условии, что общая геометрическая форма и основные элементы описания объекта останутся неизменными. Несмотря на то, что поставленная задача на первый взгляд кажется надуманной, при проектировании сложных технических изделий (и соответственно, при составлении большого числа плоских чертежей, которые в настоящее время изготовляются на компьютере)

ее решение значительно уменьшает сроки разработки конечного изделия. В настоящей работе автор развивает концептуальные подходы, предложенные им в работе [10], в целях более полного описания возможностей геометрического моделирования на основе структурных элементов.

Базовые положения построения кластерной точки или кластерного элемента

Построение геометрической фигуры на основе модели объединенного кластера (МОК-объекта) подразумевает, что вся область рассматриваемого объекта разбивается на отдельные подобласти, вид которых определяется конечным набором параметров. Эти параметры предназначены для описания геометрической области объекта путем задания центральной точки и четырех "несущих" точек (в двумерном случае), что в совокупности может быть названо кластерным элементом. Таким образом, кластерный элемент или кластерная точка представляет собой элементарную геометрическую область, вид которой задается пользователем путем определения и варьирования ее параметров. Заметим, что такой подход был бы неполным, если бы не возможность объединять эти элементарные подобласти в единое целое по определенным правилам. Используя эту возможность, можно конструировать довольно сложные геометрические фигуры, создать которые другими способами достаточно сложно. Кроме того, параметрическое описание отдельных частей создаваемой фигуры предоставляет довольно гибкие средства как по ее хранению, так и по модификации, при этом сохраняется общая целостность объекта.

С точки зрения использования предлагаемого подхода для технических приложений в промышленности можно отметить, что он особенно эффективен в литейной отрасли при компьютерном формирования сложных форм отливочных изделий. С помощью такого подхода можно упростить процесс создания исходных моделей, которые зачастую создаются ручным способом. То же самое относится и к автомобильной промышленности, где внешний дизайн нового изделия выполняется практически вручную с применением в качестве модельной основы различных пластичных материалов.

Параметризация элементарной геометрической области должна обеспечивать автоматическое формирование границы области и ее внутренних характеристик (например, цвет, температура и др.) на основе заданных простых алгоритмов. В этом случае пользователь имеет возможность естественным образом создавать простые объекты, используя в качестве простейших геометрических фигур прямоугольники, части круга и др. В представленном подходе на основе кластерной модели для построения границы ее области используются следующие параметры: координаты центральной точки; координаты четырех "несущих" точек, выделяющих четыре базовых направления на плоскости (вверх, вниз, влево, вправо); направления двух касательных прямых, связанных с каждой из четырех "несущих" точек. На рис. 1 продемонстрирована модель такого кластерного элемента, позволяющая построить границу элементарной геометрической подобласти (т. е. ее описать) в двумерном случае.

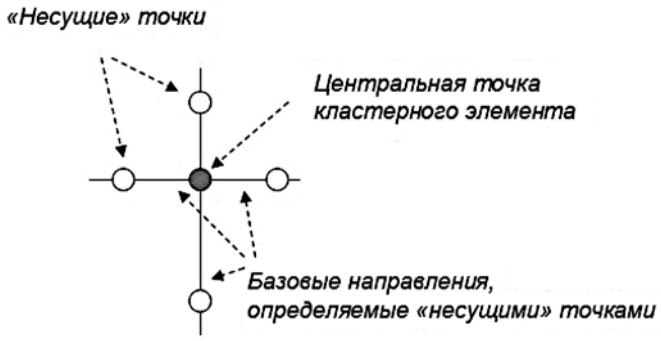

Рис. 1. Структура кластерного элемента, позволяющая параметризовать выделенную подобласть на плоскости

Таким образом, мы имеем 10 основных параметров (по две координаты для каждой из пяти точек) для описания геометрии элементарной области и 16 дополнительных (две координаты для каждого из двух направлений, связанных с четырьмя точками). Для того чтобы определить границу элементарной области, которая состоит из четырех частей, предусмотрена возможность задавать тип соединительной линии (рис. 2).

В предлагаемой модели предусмотрено использование следующих трех видов соединительных линий: отрезок; два отрезка; гладкая кривая. Эти опции кластерного элемента дают возможность представить в виде кластера достаточно широкий круг геометрических фигур.

Заметим, что дополнительная гибкость такого геометрического моделирования обеспечивается возможностью изменять не только положение центральной точки, но и положение "несущих" точек (рис. 3), что соответствующим образом видоизменяет сам кластерный элемент (геометрическую область).

Таким образом, для параметризации элементарной геометрической области добавляются еще четыре

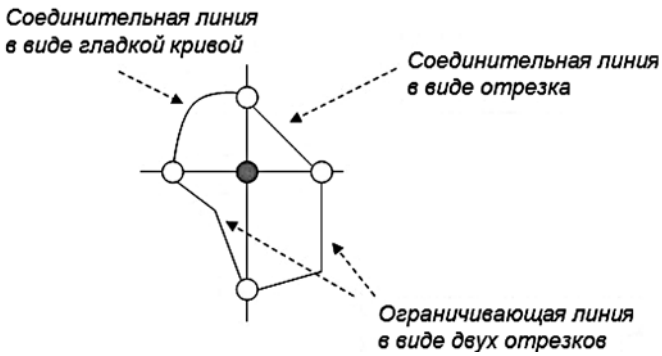

Рис. 2. Различные типы соединительных (ограничивающих) линий

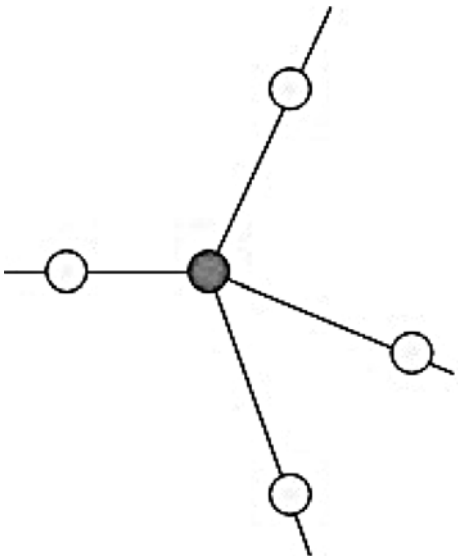

Рис. 3. Изменение базовых направлений путем смещения четырех "несущих" точек

параметра для типа соединяющих линий. Касательные линии используются для построения ограничивающей линии как состоящей из двух отрезков (см. рис. 2), так и для гладкой кривой (рис. 4).

Используя какой-либо из известных алгоритмов, можно провести гладкую кривую через две заданные точки и два известных направления (касательные прямые) в этих точках.

Таким образом, общее число параметров для описания элементарной кластерной области равно 10 + 4 + 4×4 = 30. Для современной компьютерной техники это не является критичным, так как не приводит к значительным вычислительным затратам и с точки зрения интерактивных возможностей прикладной системы геометрического моделирования не может оказать существенного влияния на ее быстродействие.

Введенное понятие кластерного элемента в определенном смысле является обобщением понятия классических геометрических фигур, таких как окружность, квадрат, прямоугольник и др. Оно позволяет единым образом осуществлять их построение и модификацию. Далее на рисунках представлены некоторые из подобных фигур. Например, квадрат получается, если для всех четырех частей кластерного элемента выбран тип соединительной линии, состоящий из двух отрезков (рис. 5, см. третью сторону обложки).

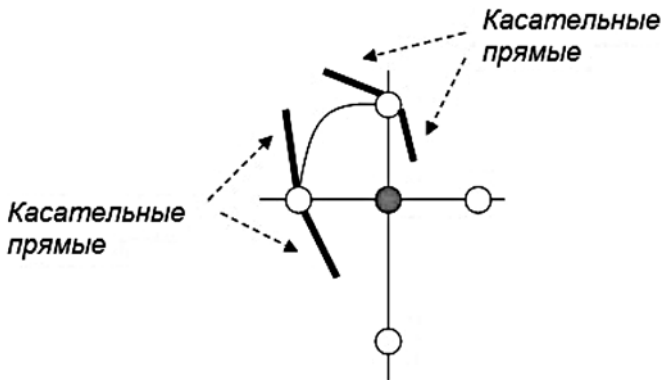

Рис. 4. Параметризация гладкой кривой путем задания касательных прямых в каждой "несущей" точке

Если в виде соединительной линии выбрать гладкую кривую, которая определяется касательными прямыми в "несущих" точках, то согласно принятому подходу получим окружность (рис. 6, см. третью сторону обложки).

Если использовать простейшую опцию, когда в качестве соединительной линии используется простой отрезок, то можно получить ромбовидную фигуру (рис. 7).

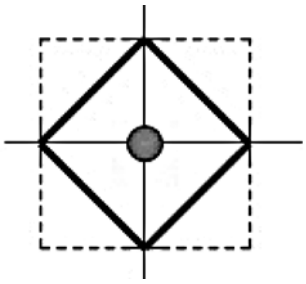

Рис. 7. Фигура ромб, которая получается при использовании простейшего типа соединительной линии в виде одного отрезка

Очевидно, что при модификации путем перемещения "несущих" точек и изменения касательных направлений будут получены более экзотические геометрические объекты. Например, если касательные линии, заданные в "несущих" точках, пересекаются в области образованного ими квадранта, то будут сформированы фигуры, показанные на рис. 8, см. третью сторону обложки.

Если касательные прямые не пересекаются в области их квадранта, то можно связать с ними следующие фигуры, показанные на рис. 9 и 10 (см. третью сторону обложки). Для их формирования вводится дополнительная разделительная точка, расположенная на соединяющем "несущие" точки отрезке, перемещение которой также влияет на вид создаваемой фигуры.

Таким образом, представленные фигуры показывают, что использование даже одного кластерного элемента позволяет сформировать не только простейшие геометрические фигуры, такие как окружность и квадрат, но и достаточно сложные, которые другими средствами создать проблематично.

Модель объединенного кластера

Отметим, что базовые положения метода построения геометрической области на основе кластерного элемента, позволяющие параметризовать определенную часть плоской фигуры, не могут обеспечить необходимой полноты и должны быть дополнены

методами объединения нескольких кластерных точек в единый многоэлементый кластерный объект.

Правила объединения должны применяться для двух или более топологически соседних кластерных точек (модель объединенного кластера) и должны давать возможность создавать сложные составные объекты, которые впоследствии могут быть использованы для геометрического моделирования реальных технических изделий. Заметим, что задание правил объединения, которые автоматически формируют конечный объект, в значительной степени упрощает действия пользователя программного средства для геометрического моделирования на основе кластерного подхода.

Для того чтобы можно было применять алгоритмы объединения, необходимо ввести понятие соседней или ближайшей кластерной точки. В этом случае можно однозначно установить связи между двумя или более объектами.

Простейшим топологически связанным объектом, для которого легко можно определить понятие ближайшего элемента, является декартово сеточное разбиение плоскости с заданным шагом, приводящее к регулярному положению кластерных элементов (центральных кластерных точек), как показано на рис. 11.

В результате исходная геометрическая фигура заменяется набором кластерных точек (рис. 12).

Для описания геометрической модели можно использовать произвольный набор кластерных точек, при условии, что известен алгоритм определения соседней точки (установленная связь) (рис. 13).

Далее описаны алгоритмы связывания или группировки кластерных точек на основе элементарной ячейки, включающей в себя четыре кластерных элемента (рис. 14).

Рассмотрим случай, когда связываются два элемента из элементарной ячейки, лежащие на одной

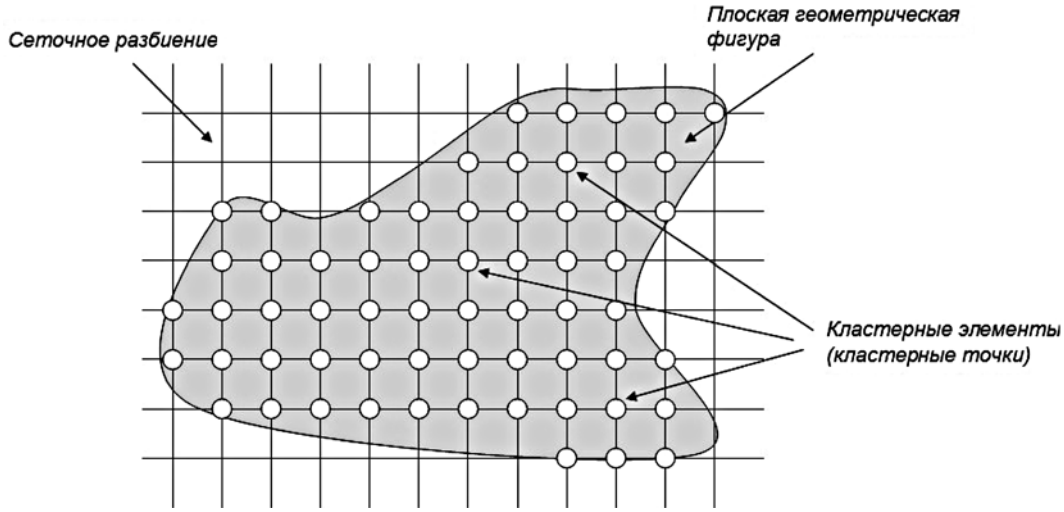

Рис. 11. Простое декартово разбиение области, которое можно использовать для формирования топологически связанной структуры кластерных точек

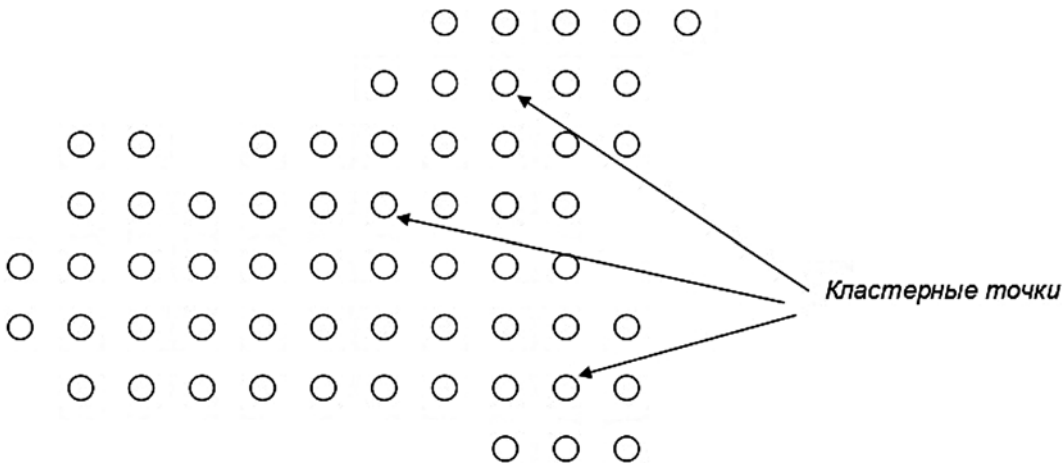

Рис. 12. Набор кластерных точек, описывающих исходную модель

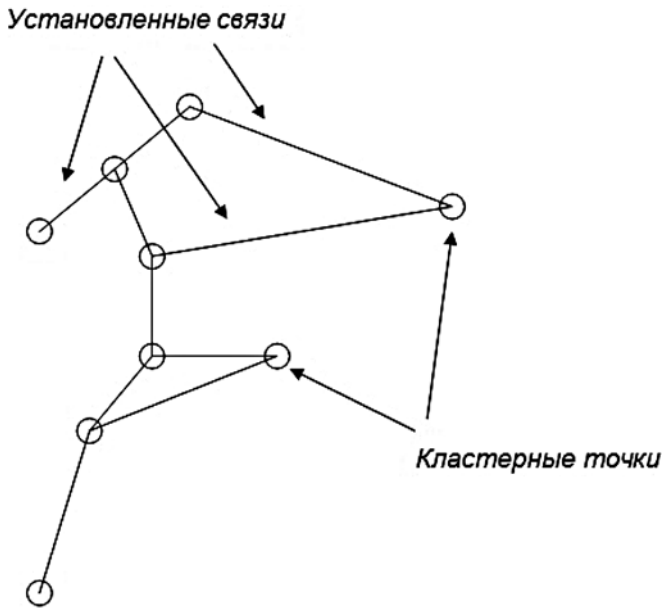

Рис. 13. Произвольный набор кластерных точек, описывающих геометрическую область

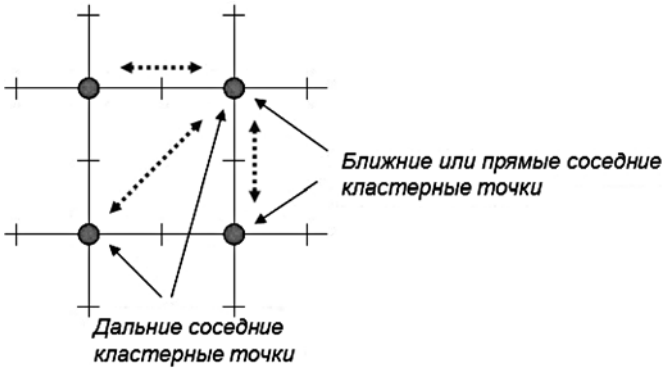

Рис. 14. Элементарная ячейка из четырех кластерных точек, в которой возможна группировка соседних элементов

грани четырехугольника, т. е. две кластерные точки являются прямыми соседними (рис. 15).

Для построения связывающей (ограничивающей) линии также используются координаты "несущих" точек и типы соединяющих линий, но только не от отдельной кластерной точки, а от двух соседних. На рис. 16 показана соединительная линия в виде отрезка, которая создается автоматически программным способом, если для обоих кластерных точек задан простейший тип соединяющей линии.

Если в качестве соединяющих линий выбран тип, определенный как два отрезка, то формируются фигуры, продемонстрированные на рис. 17, *а*.

Выбор соединяющей линии в виде гладкой кривой приводит к формированию фигур, показанных на рис. 17, *б*.

Необходимо отметить, что на рис. 17 для всех пар кластерных точек показаны две соединительные линии с одинаковым типом. Вместе с тем

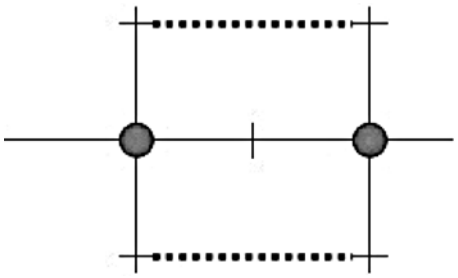

Рис. 15. Группировка двух ближних соседних точек в двухточечном кластере

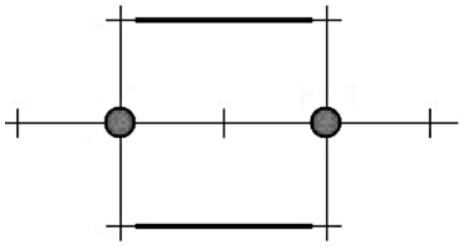

Рис. 16. Соединительная линия в двухточечном кластере в виде одного отрезка

предусмотрена возможность задавать различные типы и комбинировать варианты, когда одна из линий имеет вид отрезка, а для другая — вид гладкой кривой.

Если рассмотреть две кластерные точки, которые являются дальними соседями, то для них также можно автоматически сформировать соединительные линии (рис. 18).

Также как и для ближних соседних кластерных точек, соединительные линии могут иметь вид отдельных отрезков (рис. 19), двух отрезков (рис. 20, *а*), гладких кривых (рис. 20, *б*).

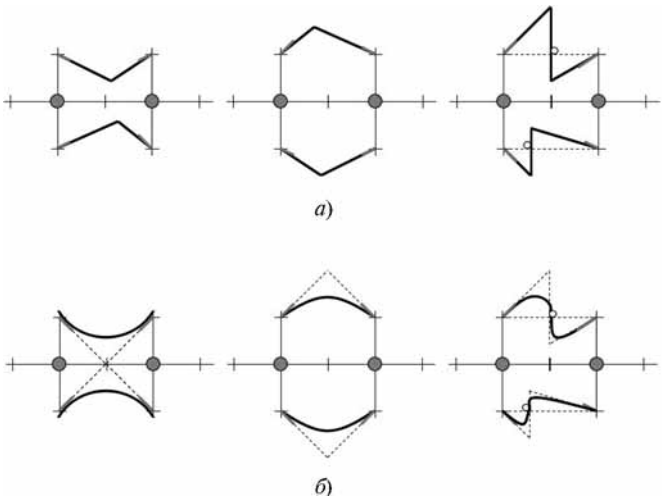

Рис. 17. Соединительная линия в двухточечном кластере в виде набора отрезков (*а*) и в виде гладкой кривой (*б*)

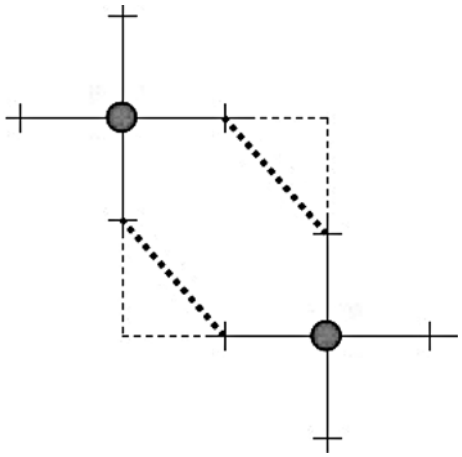

Рис. 18. Группировка двух дальних соседних точек в двухточечном кластере

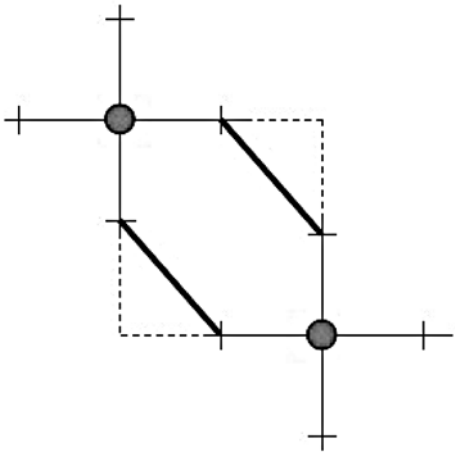

Рис. 19. Соединительная линия в двухточечном кластере в виде отдельного отрезка

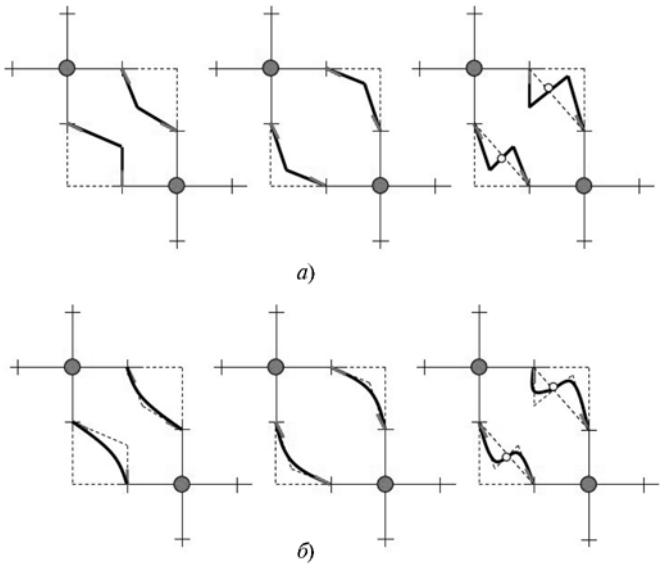

Рис. 20. Соединительные линии в двухточечном кластере в виде набора отрезков (*а*) и в виде гладких кривых (*б*)

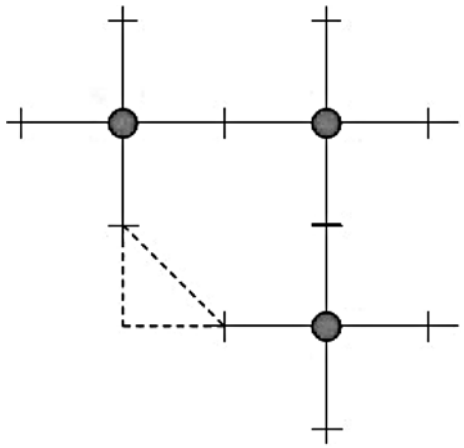

Рис. 21. Установка связей в трехточечном кластерном объекте

Группировку трех кластерных точек можно осуществить по таким же правилам, которые используют и при группировке двух кластерных точек (формирование двухточечного кластера). Трехточечный кластерный объект показан на рис. 21.

По аналогии с двухточечным объектом, соединительная линия, выделенная штриховой линией, может иметь различную форму в зависимости от заданного типа (рис. 22).

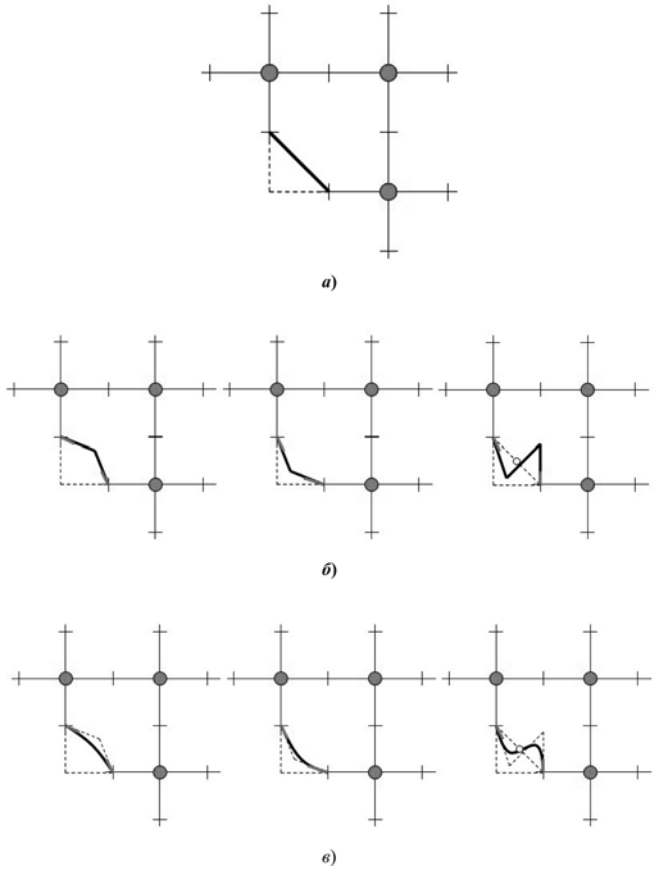

Рис. 22. Соединительная линия в трехточечном кластере в виде одного отрезка (*а*), двух отрезков (*б*) и гладкой кривой (*в*)

Очевидно, что объединение или группировка четырех кластерных точек приводит к полному заполнению четырехугольной области, и соединяющих или ограничивающих линий в этом случае не будет.

Программная реализация алгоритмов построения МОК-объектов

Для программного описания предложенного подхода был использован язык Visual C++, на котором и было создано специализированное программное обеспечение, позволяющее формировать двумерные кластерные модели. С точки зрения функциональных возможностей интерактивная программная среда имеет средства для создания сеточных областей различных размеров, для выбора и редактирования необходимых кластерных точек с помощью манипулятора "мышь", для изменения их взаимного расположения, для задания цвета элементарной геометрической области и др.

Для реализации описанных выше методов группировки кластерных точек, которые используются при создании МОК-объектов, была разработана связанная система классов на языке C++, методы которых были оформлены в виде отдельных специализированных процедур. С алгоритмической точки зрения для построения отдельного МОК-объекта на первом шаге проводили анализ всех кластерных точек для выделения соседних элементов каждой из них. Затем определяли тип группировки для соседних точек — двухточечная, трехточечная или четырехточечная. После этого применяли необходимый для данного типа группировки алгоритм объединения и выполняли построение конечной модели.

На рис. 23 показан результат работы программы при создании одноэлементного кластерного элемента, который в одном случае имеет вид окружности (*а*) (см. рис. 6), а в другом — квадрата (*в*) (см. также рис. 5). С изменением его свойств — направления касательных и типа соединяющей линии для каждой из квадрантов — были получены фигуры, представленные в правой части рис. 23 (*б*, *г*).

Еще раз отметим, что имея в распоряжении четыре направления, для которых структура кластерного элемента позволяет задавать различные типы соединительных линий, можно получить большой набор геометрических фигур на плоскости, которые в определенном смысле обобщают понятия круга и квадрата.

Заметим, что в силу того, что кластерное описание геометрической модели предполагает автоматическое (программное) построение геометрической области, пользователь должен иметь широкие возможности интерактивного задания вида границы путем перемещения "несущих" и граничных точек, изменения касательных направлений и др. На рис. 24 (см. четвертую сторону обложки) показаны различные виды пятиточечного кластерного элемента. Заданная топология определяет тип группировки кластерного

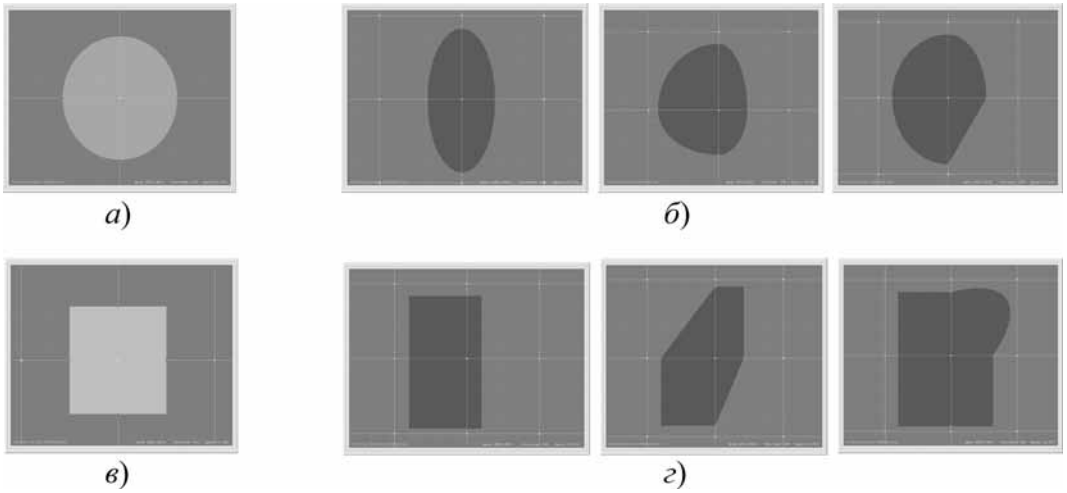

Рис. 23. Результаты работы программного комплекса при построении одноэлементного кластера:

а — в качестве соединительных линий выбраны гладкие кривые для всех квадрантов с касательными, перпендикулярными координатным прямым; *б* — фигуры, полученные модификацией окружности путем изменения направления касательных и положения несущих точек; *в* — квадрат, который формируется при выборе в качестве соединительных линий двух отрезков для каждого из квадрантов; *г* — фигуры, полученные путем модификации квадрата за счет изменения положения несущих точек и типа соединительной линии

объекта — центральная точка соединена с четырьмя точками и в свою очередь эти четыре точки соединены только с одной.

При этом, как видно на рис. 24 (см. четвертую сторону обложки), программная среда генерирует совершенно непохожие геометрические объекты, объединенные лишь общей топологией — исходный геометрический объект (рис. 24, *а*) и его различные модификации (рис. 24, *б—д*). На первом рис. 24, *а*, это обычный крест. На рис. 24, *б*, когда в кластерных элементах, определяющих концы креста, в качестве соединительных линий заданы гладкие линии, мы имеем четыре соединенных круга. Фигуры на рис. 24, *в* и *г* различаются направлением касательных линий, положением "несущих" точек и типом соединяющей линии для центральной кластерной точки. Последняя модификация (рис. 24, *д*) отличается от первой положением "несущих" точек в центральном элементе кластера.

На рис. 25 (см. четвертую сторону обложки) приведены примеры построения сложных форм на основе разработанной программной среды кластерного моделирования. Используя свойства элементарных кластеров, выбранных пользователем для формирования объекта, пользователь имеет возможность задать гладкие кривые с заданным углом наклона в различных граничных точках и провести их сопряжение друг с другом. Такая возможность позволяет построить, например, профиль крыла, беря в качестве основы четыре кластерные точки (рис. 25, *а*).

Если задать центрально симметричное расположение "несущих точек", то кластерная модель дает возможность для проектирования различных центрально-симметричных объектов, например, колеса (рис. 25, *б*).

Рис. 25, *в* и *г* демонстрируют возможности разработанной среды по созданию фрактальных объектов разнообразной внутренней структуры. Для создания похожих объектов в других средах геометрического проектирования пользователь должен провести довольно много времени перед компьютером.

Заключение

Представленный в статье алгоритм построения двумерных геометрических объектов, базирующийся на понятии кластерных точек, имеет достаточно хорошие возможности для практического применения. В частности, он может быть эффективен при создании шаблонов для дальнейшего использования в станках с ЧПУ, например, при лазерной резке и гравировке. Существенным преимуществом данного подхода является возможность локального изменения вида изделия, в том числе путем добавления новых кластерных точек, которое не требует внесения изменений в другие части единой связанной модели. Такая опция позволяет не только упростить процесс проектирования и в конечном итоге сократить время разработки нового изделия, но и обеспечить достаточную гибкость для получения критически важных параметрических характеристик.

Интуитивно понятный подход и соответственно интерфейс разработанного программного обеспечения, используемый при кластерном геометрическом проектировании, позволяют в значительной степени наращивать степень сложности конечной модели, а используя рекурсивное масштабирование — формировать геометрически сложные вложенные иерархические структуры.

Особое значение разработанный подход может играть при использовании в алгоритмах машинного зрения и распознавании изображений. Метод кластерных точек позволяет с разной степенью точности "загрублять картинку", что дает возможность вычленить из всей сцены только существенные и значимые элементы, на которых в дальнейшем может быть построен анализ изображения.

Представленные в статье кластерные модели и их полная классификация показывают, что кластерный подход и кластерная точка являются достаточно универсальным средством геометрического моделирования двумерных объектов. Такой подход позволяет естественным образом обобщить различные геометрические фигуры, а его использование предоставляет достаточную гибкостью, что является немаловажным преимуществом при создании различных геометрических объектов.

Список литературы

1. **Farin G. E., Hagen H., Noltemeier H.** Geometric modelling, Springer-Verlag, 1993. 316 p.
2. **Gintaris Cinelis,** Methods of Computer Aided Geometric Modelling of Architectural Objects, Vilniaus pedagoginio universiteto leidykla, 2011. 72 p.
3. **Sarfraz M.** (Eds.) Geometric Modeling: Techniques, Applications, Systems and Tools, 2004. 454 p.
4. **Басов К. А.** ANSYS и LMS Virtual Lab. Геометрическое моделирование. М.: ДМК Пресс, 2006. 240 с.
5. **Большаков В., Тозик В., Чагина А.** Инженерная и компьютерная графика. СПб.: БХВ-Петербург, 2013, 286 с.
6. **Голованов Н. Н.** Геометрическое моделирование. М.: Физматлит, 2002. 272 с.
7. **Сиденко Л.** Компьютерная графика и геометрическое моделирование. СПб.: Питер, 2009. 224 с.
8. **Трошин П. И.** Компьютерная геометрия и геометрическое моделирование, Казанский федеральный университет, 2015. 56 с.
9. **Жураев Т.** Основы геометрического моделирования рабочих органов мелиоративной и сельскохозяйственной техники. Lambert Academic Publishing, 2015. 168 с.
10. **Светушков Н. Н.** Кластерная модель геометрического описания сложных объектов // САПР и графика. 2010. № 3. С. 86—88.

Creating Two-Dimensional Geometric Objects Based on Universal Structural Element — Cluster Point

N. N. Svetushkov, svetushkov@mai.ru, Moscow Aviation Institute (National Research University), Moscow, 125993, Russian Federation.

Corresponding author:

Svetushkov Nikolay N., Assistant Professor, Moscow Aviation Institute (National Research University), Moscow, 125993, Russian Federation,
E-mail: svetushkov@mai.ru

Received on May 24, 2018
Accepted on December 27, 2018

The article discusses a new approach to the creation of complex models — two-dimensional geometric objects. The basis of this approach is the topological characteristics of the model, which are set by the user in separate selected nodes — cluster points, which provides the necessary flexibility of description and allows changing the external representation of the object to a considerable extent. The cluster representation has a sufficient degree of universality, allowing the formation of rather complex geometric models. At the same time, an important feature of such a parametric representation is the ability to add necessary geometric characteristics by simple means and modify existing ones. The basic concept of a cluster point introduced in the paper is based on the possibility of specifying a set of parameters for a local description of a closed geometric region surrounding a given point. Combining a finite set of these areas allows you to create a new parameterized geometric object as a whole, which was called the MUC object (an object built on the model of a unified cluster). Drawing an analogy with computer graphics, we can say that the MUC-object is a vector image of a complex geometric shape (as opposed to a raster image, which is defined by the color of each pixel on the screen). In other words, the proposed geometric description method allows you to convert a complex geometric image into a "vector" format, and thus significantly reduce the total amount of stored data (as opposed to a pixel image). The article describes a universal set of parameters relating to a separate cluster point, which allows one to construct both the simplest geometric objects and more complex ones, based on predetermined algorithms for combining them. The presented parameters allow us to construct the boundary of a geometric object in the form of a piecewise-smooth line, the form of which depends on the specified angles and types of connecting lines at each cluster point. The possible combinations of lines for the IOC object, consisting of one cluster point, as well as the combination of two and three points are listed. For the last two cases, merging algorithms are described, also based on the parameters specified at cluster points. The paper concludes that a cluster object consisting of a single point is a generalization of classical geometric figures — a square, a rectangle, a circle and an ellipse and allows considering the algorithm of their construction from a unified position. At the end of the article, the software implementation of the presented approach is briefly described, which allows us to construct not only the above-named figures, but also, as an example, a five-element MUC object. In the latter case, there are several screen shots of this object with modified parameters, which demonstrates that, despite the same topology, externally, these objects look significantly different. The simplest examples show the applicability of the developed approach in the design of technical products, in particular, with its help it is possible to draw a wing profile or a wheel with spokes. At the end of the work, screen shots of rather exotic figures are presented, which would have required considerable effort by other means. In the end, it is concluded about the possibilities of wide practical application of this approach.

Keywords: *cluster models, geometric description, topology, algorithmic procedures, visualization, software*

For citation:

Svetushkov N. N. Creating Two-Dimensional Geometric Objects Based on Universal Structural Element — Cluster Point, *Programmnaya Ingeneria,* 2019, vol. 10, no. 3, pp. 135—144.

DOI: 10.17587/prin.10.135-144

References

1. **Gerald E., Hans H., Hartmut N.** *Geometric modelling*, Springer-Verlag, 1993. 316 p.
2. **Gintaris C.** *Methods of Computer Aided Geometric Modelling of Architectural Objects,* Vilniaus pedagoginio universiteto leidykla, 2011, 72 p.
3. **Sarfraz M.** (Eds.) *Geometric Modeling: Techniques, Applications, Systems and Tools*, Kluwer Academic Publishers, 2004, 454 p.
4. **Basov K. A.** *ANSYS i LMS Virtual Lab. Geometricheskoe modelirovanie* (ANSYS and LMS Virtual Lab. Geometric Modelling), Moscow, DMK Press, 2006, 240 p. (in Russian).
5. **Bolshakov V., Tozik V., Chagina A.** *Inzhenernaya i komp'yuternaya grafika* (Engineering and Computer Graphics), Saint-Petersburg, BHV-Peterburg, 2013, 286 p. (in Russian).
6. **Golovanov N. N.** *Geometricheskoe modelirovanie* (*Geometric Modeling*), Moscow, Fizmatlit, 2002, 272 p. (in Russian).
7. **Sidenko L.** *Komp'yuternaya grafika i geometricheskoe modelirovanie* (Computer graphics and geometric modeling), Saint-Petersburg, Piter, 2009, 224 p. (in Russian).
8. **Troshin P. I.** *Kompyuternaya geometriya i geometricheskoe modelirovanie* (Computer geometry and geometric modeling), Kazan', Kazanskij federalnyj universitet, 2015, 56 p. (in Russian).
9. **Zhuraev T.** *Osnovy geometricheskogo modelirovaniya rabochih organov meliorativnoj i sel'skohozyajstvennoj tekhniki* (Fundamentals of geometric modeling of working bodies of land reclamation and agricultural equipment), Lambert Academic Publishing, 2015, 168 p. (in Russian).
10. **Svetushkov N. N.** Klasternaya model' geometricheskogo opisaniya slozhnyh ob"ektov (Cluster model of the geometric description of complex objects), *SAPR i Grafika*, 2010, no. 3, pp. 86—88 (in Russian).

ИНФОРМАЦИЯ

Начинается подписка на журнал "Программная инженерия" на второе полугодие 2019 г.

Оформить подписку можно через подписные агентства
или непосредственно в редакции журнала.

Подписные индексы по каталогу:

Пресса России — 39795

Адрес редакции: 107076, Москва, Стромынский пер., д. 4,
Издательство "Новые технологии",
редакция журнала "Программная инженерия"

Тел.: (499) 269-53-97. Факс: (499) 269-55-10. E-mail: prin@novtex.ru

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева*. Корректор *Н. В. Яшина*

Сдано в набор 10.01.2019 г. Подписано в печать 25.02.2019 г. Формат 60×88 1/8. Заказ PI319
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**