OBJECT MANAGEMENT GROUP

Версия 1.3

Модель и нотация принятия решений (DMN)

Стандарт DMN (Decision Model and Notation) был разработан группой компаний «Object Management Group» и переведен компанией ELMA на русский язык.

ELMA

МОДЕЛЬ И НОТАЦИЯ ПРИНЯТИЯ РЕШЕНИЙ (DMN)

Версия 1.3

Номер документа OMG: **dtc/19-12-06**

Дата выпуска: 9 декабря 2019

Стандартный URL документа: http://www.omg.org/spec/DMN

Нормативные машиночитаемые файлы:

https://www.omg.org/spec/DMN/20191111/DMN13.xsd
https://www.omg.org/spec/DMN/20191111/DMNDI13.xsd
https://www.omg.org/spec/DMN/20180505/DI.xsd
https://www.omg.org/spec/DMN/20180505/DC.xsd
https://www.omg.org/spec/DMN/20191007/DMN13.xmi
https://www.omg.org/spec/DMN/20191008/DMNDI13.xmi

Информативные машиночитаемые файлы:

https://www.omg.org/spec/DMN/20191111/examples.zip

ИСПОЛЬЗОВАНИЕ СПЕЦИФИКАЦИИ: УСЛОВИЯ, ПОЛОЖЕНИЯ И ПРЕДУПРЕЖДЕНИЯ

В данном документе содержится подробное описание спецификации Object Management Group в соответствии с изложенными ниже условиями, положениями и предупреждениями. Данный документ не является обязательством по реализации любых частей упомянутой спецификации в виде программных продуктов какой-либо компании. Информация, содержащаяся в этом документе, может быть изменена без предварительного предупреждения.

ЛИЦЕНЗИИ

Компании, перечисленные выше, предоставили Object Management Group, Inc. (OMG) неисключительную, безвозмездную, оплаченную, глобальную лицензию на копирование, распространение и изменение данного документа, а также на распространение его изменённых версий. Все обладатели авторских прав, перечисленные выше, согласились с тем, что использование спецификации, изложенной в данном документе, или приведение ПО в соответствие с данной спецификацией, не является нарушением авторских прав на данный документ любого из перечисленных правообладателей.

В соответствии со всеми нижеприведенными положениями и условиями, владельцы авторских прав на данную спецификацию настоящим предоставляют вам полностью оплаченную, неисключительную, непередаваемую, бессрочную, глобальную лицензию (без права предоставления сублицензии) на использование данной спецификации для создания и распространения программного обеспечения и спецификаций специального назначения, основанных на данном документе, а также на использование, копирование и распространение данной спецификации, в соответствии с Законом об авторском праве; при условии, что: (1) все копии данной спецификации будут содержать уведомление об авторском праве, приведенное выше, а также данное уведомление о разрешении на использование; (2) спецификация должна использоваться только в ознакомительных целях; не допускается копирование или размещение спецификации на любом сетевом компьютере или носителе, а также передача или продажа спецификации в коммерческих целях; и (3) не допускается внесение изменений в данную спецификацию. В случае нарушения любого из вышеперечисленных условий данное ограниченное разрешение на использование автоматически прекращает свое действие без предварительного уведомления. В случае прекращения действия разрешения, вы обязуетесь немедленно уничтожить любые копии спецификации, находящиеся в вашем распоряжении.

ПАТЕНТЫ

Обращаем внимание соразработчиков, что возможно для приведения в соответствие или применения спецификации OMG потребуется использовать разработки, охватываемые патентными правами. OMG не несет ответственности за выявление патентов, которые требуют лицензию на какую-либо спецификацию OMG, или за изучение юридической силы или объема патентов, о которых ему стало известно. Спецификации OMG носят

информационный и консультативный характер. Потенциальные пользователи несут ответственность за соблюдение патентных прав и обязуются самостоятельно защищать себя в случае их нарушения.

ОБЩИЕ УСЛОВИЯ ИСПОЛЬЗОВАНИЯ

Любое неправомочное использование данной спецификации может нарушать законы об авторском праве, законы о товарных знаках, а также правила и положения о коммуникациях. Данный документ содержит информацию, которая защищена авторским правом. Все права защищены. Никакая часть этой работы, защищенная авторским правом, не может быть воспроизведена или использована в какой-либо форме или любыми средствами: графическими, электронными или механическими, включая фотокопирование, запись, запись на пленку или системы хранения и извлечения информации, без разрешения владелец авторского права.

ОТКАЗ ОТ ГАРАНТИИ

ДАННАЯ ПУБЛИКАЦИЯ СЧИТАЕТСЯ ДОСТОВЕРНОЙ, ОНА ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», ОДНАКО МОЖЕТ СОДЕРЖАТЬ ОШИБКИ ИЛИ ОПЕЧАТКИ. OMG И КОМПАНИИ, ПЕРЕЧИСЛЕННЫЕ ВЫШЕ, НЕ ДАЮТ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ В ОТНОШЕНИИ ЭТОЙ ПУБЛИКАЦИИ, ВКЛЮЧАЯ, НО НЕ ОГРАНИЧИВАЯСЬ ЛЮБЫМИ ГАРАНТИЯМИ ЗАКОННОСТИ ПРАВА СОБСТВЕННОСТИ, ПОДРАЗУМЕВАЕМЫМИ ГАРАНТИЯМИ КОММЕРЧЕСКОЙ ЦЕННОСТИ ИЛИ ГАРАНТИЯМИ ПРИГОДНОСТИ ДЛЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ ИЛИ ИСПОЛЬЗОВАНИЯ. НИ ПРИ КАКИХ ОБСТОЯТЕЛЬСТВАХ OMG ИЛИ ЛЮБАЯ ИЗ КОМПАНИЙ, ПЕРЕЧИСЛЕННЫХ ВЫШЕ, НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ЗА ОШИБКИ, СОДЕРЖАЩИЕСЯ В ДАННОМ ДОКУМЕНТЕ ИЛИ ЗА ПРЯМЫЕ, НЕПРЯМЫЕ, СЛУЧАЙНЫЕ, ОСОБЫЕ, КОСВЕННЫЕ УБЫТКИ, РАСХОДЫ, ПОНЕСЕННЫЕ В РАСЧЁТЕ НА ИСПОЛНЕНИЕ ДОГОВОРА, ВКЛЮЧАЯ ПОТЕРЮ ПРИБЫЛИ, ДОХОДА, ДАННЫХ ИЛИ УТРАТУ ВОЗМОЖНОСТИ ЭКСПЛУАТАЦИИ, ПОНЕСЁННЫЕ ЛЮБЫМ ПОЛЬЗОВАТЕЛЕМ ИЛИ ЛЮБОЙ ТРЕТЬЕЙ СТОРОНОЙ В СВЯЗИ С ПРЕДОСТАВЛЕНИЕМ, ВЫПОЛНЕНИЕМ И ИСПОЛЬЗОВАНИЕМ ДАННОГО ДОКУМЕНТА, ДАЖЕ ЕСЛИ ЭТОТ ПОЛЬЗОВАТЕЛЬ ИЛИ ЛЮБАЯ ТРЕТЬЯ СТОРОНА БЫЛИ УВЕДОМЛЕНЫ О ВОЗМОЖНОСТИ ТАКИХ УБЫТКОВ.

Пользователь несет все риски, связанные с качеством и производительностью программного обеспечения, разработанного с использованием этой спецификации.
Этот отказ от гарантии является неотъемлемой частью лицензии, предоставленной для использования данной спецификации.

ОГРАНИЧЕНИЕ ПРАВ

Использование, дублирование или раскрытие информации правительством США регулируется ограничениями, изложенными в подпункте (c) (1) (ii) статьи «О технической информации и компьютерном программном обеспечении» Приложения к правилам МО по осуществлению закупок для федеральных нужд 252.227-7013 или в подпункте (c) (1) и (2) статьи «Об ограничении прав на коммерческое компьютерное программное обеспечение» положения 48 C.F.R. 52.227-19 или в разделе 48 C.F.R. 227-7202-2 Приложения к правилам МО по осуществлению закупок для федеральных нужд и в его последующих редакциях, или в разделе 48 C.F.R. 12.212 Правил закупок для федеральных нужд и в их последующих редакциях, если это применимо. Владельцы авторских прав на спецификации указаны выше, с ними можно связаться по адресу Object Management Group, 109 Highland Avenue, Needham, MA 02494, U.S.A.

ТОВАРНЫЕ ЗНАКИ

CORBA®, CORBA logos®, FIBO®, Financial Industry Business Ontology®, FINANCIAL INSTRUMENT GLOBAL IDENTIFIER®, IIOP®, IMM®, Model Driven Architecture®, MDA®, Object Management Group®, OMG®, OMG Logo®, SoaML®, SOAML®, SysML®, UAF®, Unified Modeling Language®, UML®, UML Cube Logo®, VSIPL®, и XMI® являются зарегистрированными товарными знаками Object Management Group, Inc.

Полный список товарных знаков см. по адресу в интернете: http://www.omg.org/legal/tm_list.htm. Все другие упомянутые продукты или названия компаний используются только для целей идентификации и могут быть товарными знаками соответствующих владельцев.

СООТВЕТСТВИЕ

Все владельцы авторских прав, перечисленные выше, признают, что компания Object Management Group (действующая сама или через своих доверенных лиц) является и всегда будет единственным органом, который может разрешать разработчикам, поставщикам и продавцам программного обеспечения использовать сертификационные знаки, товарные знаки или другие специальные обозначения для указания соответствия приведенному здесь материалу.

Разработчик программного обеспечения, созданного в соответствии с условиями настоящей лицензии, может заявлять о соответствия ПО этой спецификации только в том случае, если соответствие программного обеспечения имеет характер, полностью отвечающий применимым требованиям, указанным в спецификации. Программное обеспечение, разработанное в частичном соответствии с применимыми требованиями, является ПО, основанным на спецификации, но не является ПО, соответствующим спецификации. В случае если компания Object Management Group разработает и утвердит программу для тестирования, программное обеспечение, разработанное с использованием данной спецификации, будет считаться соответствующим спецификации только в том случае, если оно прошло тесты с удовлетворительным результатом.

СООБЩЕНИЯ ОБ ОБНАРУЖЕННЫХ ОШИБКАХ

Все спецификации OMG подлежат постоянному пересмотру и улучшению. В рамках этого процесса мы рекомендуем читателям сообщать о любых обнаруженных несоответствиях или неточностях, путем заполнения формы сообщения об ошибках на главной странице веб-сайта http://www.omg.org, в разделе «Документы», «Сообщить об ошибке/проблеме».

Оглавление

Предисловие

OMG

Компания Object Management Group, Inc (OMG), основанная в 1989 году, является некоммерческим консорциумом стандартов компьютерной индустрии с открытым членством, который выпускает и поддерживает спецификации для межоперационных, межплатформенных и многократно используемых корпоративных приложений в распределенных гетерогенных средах. Консорциум включает поставщиков информационных технологий, конечных пользователей, правительственные учреждения и работников научно-образовательной сферы.

Компании-члены OMG пишут, адаптируют и поддерживают свои спецификации в соответствии со зрелым, открытым процессом. Спецификации OMG реализуют методологию Model Driven Architecture® (MDA®), позволяя максимизировать прибыль на инвестируемый капитал благодаря концепции полного жизненного цикла, применимой к интеграции прикладных систем предприятия. Эта концепция охватывает несколько операционных систем, языков программирования, межплатформенное программное обеспечения и сетевые инфраструктуры, а также среды разработки программного обеспечения. Спецификации OMG включают: UML® (унифицированный язык моделирования ®); CORBA® (обобщённая архитектура обработчика объектных запросов); CWM ™ (обобщённая складская метамодель ™); и отраслевые стандарты для десятков вертикальных рынков.

Более подробную информацию о OMG можно найти на сайте http://www.omg.org/.

Спецификации OMG

Как уже отмечалось ранее, спецификации OMG предназначены для межплатформенного программного обеспечения, моделирования и вертикальных структур домена. Все спецификации OMG доступны на веб-сайте OMG, расположенном по адресу в интернете:

http://www.omg.org/spec

Все официальные спецификации OMG можно скачать бесплатно с нашего сайта. (Продукты, реализующие спецификации OMG могут быть приобретены у отдельных поставщиков.) Копии спецификаций в формате PostScript и PDF доступны для скачивания из Каталога спецификаций, упомянутого выше. Копии спецификаций можно также получить, обратившись в Object Management Group, Inc по адресу:

OMG Headquarters
109 Highland Avenue
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: pubs@omg.org

Некоторые спецификации OMG могут также использоваться в качестве стандартов ISO. Более подробная информация представлена на сайте http://www.iso.org

Ошибки

Все спецификации OMG подлежат постоянному пересмотру и улучшению. Авторы данной работы просят читателей сообщать о любых неточностях, несоответствиях или фразах, которые могут быть истолкованы неоднозначно, путем заполнения формы сообщения об ошибках на главной странице веб сайта http://www.omg.org, в разделе «Документы», «Сообщить об ошибке/проблеме».

1 Область действия документа

DMN была задумана как единая нотация, понятная всем бизнес-пользователям, начиная с бизнес-аналитиков, которым необходимо разработать первоначальные требования к принятию решения, а затем более подробные модели принятия решений; технических разработчиков, отвечающих за автоматизацию решений в процессах, и заканчивая представителями деловых кругов, которые будут управлять этими решениями и контролировать их. DMN – это своеобразный мост, позволяющий преодолеть разрыв между моделированием бизнес-решений и их реализацией. Нотация DMN предназначена для использования совместно со стандартной нотацией описания бизнес-процессов BPMN.

Другая цель, которую преследует данная нотация, обеспечить обмен моделями принятия решений между организациями в формате XML.

Почерпнув знания и опыт из существующего сообщества моделирования решений, авторы данной книги стремились объединить общие идеи, содержащиеся в противоречивых нотациях, предлагаемых сообществом, в единую стандартизированную нотацию.

Эта страница намеренно оставлена пустой

2 Соответствие требованиям спецификации

2.1 Уровни соответствия

Программное обеспечение может считаться соответствующим нотации **DMN 1.3** только в том случае, если ПО полностью отвечает всем требованиям соответствия, указанным в данной спецификации. Программное обеспечение, разработанное в частичном соответствии с применимыми требованиями, считается ПО, основанным на спецификации, но не является ПО, соответствующим спецификации.

Спецификация определяет три уровня соответствия: **уровень соответствия 1**, **уровень соответствия 2** и **уровень соответствия 3**.

Для ПО, претендующего на соответствие уровню 1, не требуется подтверждать соответствие Уровню 2 и 3. Для ПО, претендующего на соответствие уровню 2, не требуется подтверждать соответствие Уровню 3.

Программное обеспечение, претендующее на соответствие **уровню 1**, ДОЛЖНО отвечать всем требованиям, изложенным в разделах 6 (Требования к принятию решений), 7 (Логика принятия решений) и 8 (Таблица принятия решений) данного документа. Программному обеспечению, претендующему на соответствие **уровню 1**, не требуется интерпретировать выражения (элемент моделирования – `Expression`) в моделях принятия решений. Однако в случае если ПО, претендующее на соответствие **уровню 1,** интерпретирует выражение, эта интерпретация ДОЛЖНА соответствовать семантике выражений, описанной в разделе 7.

Программное обеспечение, претендующее на соответствие **уровню 2**, ДОЛЖНО отвечать всем требованиям, изложенным в разделах 6 (Требования к принятию решений), 7 (Логика принятия решений) и 8 (Таблица принятия решений) данного документа. Кроме того, оно должно интерпретировать выражения на языке простых выражений (S-FEEL), указанные в разделе 9.

Программное обеспечение, претендующее на соответствие **уровню 3**, ДОЛЖНО отвечать всем требованиям, изложенным в разделах 6 (Требования к принятию решений), 7 (Логика принятия решений), 8 (Таблица принятия решений) и 10 (Язык выражений) данного документа. Обратите внимание, что простой язык выражения, описанный в разделе 9, является подмножеством FEEL, и поэтому ПО, претендующее на соответствие **уровню 3,** может также претендовать на соответствие **уровню 2** (и **уровню 1**).

Кроме того, ПО, претендующее на соответствие любому из трех уровней соответствия **DMN 1.3**, ДОЛЖНО отвечать всем требованиям, изложенным в подпункте 2.2.

2.2 Общие требования к соответствию

2.2.1 Внешний вид графических элементов

Важнейшим аспектом использование **DMN** является выбор форм и иконок графических элементов, описанных в данной спецификации.

DMN преследует цель создать стандартный визуальный язык, который будет узнаваем и понятен для всех разработчиков моделей решений. ПО, которое позволяет пользователям создавать и отображать диаграммы модели решений, ДОЛЖНО использовать графические элементы, фигуры и маркеры, представленные в данной спецификации.

Допускаются изменения в размере, цвете, стиле линий и расположении текста определенных графических элементов, за исключением тех случаев, когда указано иное.

Допускаются следующие расширения диаграмм **DMN**:

- новые маркеры или индикаторы МОГУТ добавляться к указанным графическим элементам. Эти маркеры или индикаторы могут использоваться для выделения конкретного атрибута элемента **DMN** или для представления нового подтипа соответствующего понятия;

- на Диаграмму в качестве вариации артефакта МОЖЕТ БЫТЬ добавлена новая форма, однако она НЕ ДОЛЖНА противоречить форме, определенной для какого-либо другого элемента **DMN** или маркера;

- графические элементы МОГУТ БЫТЬ выделены цветом. Использование какого-либо цвета МОЖЕТ соответствовать определенной семантике, используемой для дополнения информации, которую, согласно данному документу, передают графические элементы;

- стили линий, используемые для отображения графических элементов, МОГУТ БЫТЬ изменены, однако они НЕ ДОЛЖНЫ противоречить каким-либо другим стилям линий, указанным в данной спецификации.

Расширение НЕ ДОЛЖНО изменять установленную форму какого-либо графического элемента или маркера (например, пунктирная линия не может заменяться сплошной линией, квадрат не может заменяться треугольником, закругленные края не могут стать острыми и т.д.)

2.2.2 Семантика решений

Эта спецификация описывает несколько семантических понятий, используемых для определения решений, и связывает их с графическими элементами, маркерами и соединениями.

Поскольку при применении диаграмма **DMN** интерпретируется как семантическая спецификация данной концепции, такая интерпретация ДОЛЖНА согласовываться с описанной в данном документе семантической интерпретацией.

2.2.3 Атрибуты и ассоциации

Данная спецификация определяет атрибуты и свойства семантических элементов, представленных посредством графических элементов, маркеров и соединений. Некоторые атрибуты являются обязательными, однако могут отображаться по желанию или не отображаются вообще, а некоторые атрибуты являются опциональными.

Для каждого обязательного атрибута или свойства ДОЛЖЕН БЫТЬ определен механизм, с помощью которого можно создавать и отображать значения атрибутов или свойств. Такой механизм ДОЛЖЕН позволять пользователям создавать или просматривать значения атрибутов или свойств элемента **DMN**.

На модели в ОБЯЗАТЕЛЬНОМ ПОРЯДКЕ отображаются все атрибуты и свойства, которые, согласно спецификации, должны иметь графическое отображение. Если отображение конкретного атрибута или свойства является опциональным, то при моделировании ДОПУСКАЕТСЯ использовать графическое отображение такого атрибута или свойства либо воспользоваться другими механизмами отображения.

Если атрибуты или свойства отображаются графически, то их отображение ДОЛЖНО осуществляться в соответствии со спецификацией. Если графическое отображение атрибутов и свойств не является обязательным, ПО для моделирования МОЖЕТ предусматривать использование графического отображения либо иные механизмы отображения. Используемое графическое представление НЕ ДОЛЖНО противоречить графическому представлению, определённому спецификацией для какого-либо другого элемента **DMN**.

3 Ссылки

3.1 Нормативные источники

BMM - *Business Motivation Model (BMM), Version 1.2*, OMG Document number: formal/2014-05-01, May 2014
http://www.omg.org/spec/BMM/1.2

BPMN 2.0 - *Business Process Model and Notation, version 2.0*, OMG Document Number: formal/2011-01-03, January 2011
http://www.omg.org/spec/BPMN/2.0

CQL - *Clinical Quality Language, V1.4, HL7*
https://cql.hl7.org/09-b-cqlreference.html#interval-operators-3

IEEE 754 - *IEEE 754-2008, IEEE Standard for Floating-Point Arithmetic*, International Electrical and Electronics Engineering Society, December 2008
http://www.techstreet.com/ieee/searches/5835853

ISO 8601 - ISO 8601:2004, *Data elements and interchange formats -- Information interchange -- Representation of dates and times*, International Organization for Standardization, 2004
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=40874

ISO EBNF - ISO/IEC 14977:1996, *Information technology -- Syntactic metalanguage -- Extended BNF*, International Organization for Standardization, 1996
http://standards.iso.org/ittf/PubliclyAvailableStandards/s026153_ISO_IEC_14977_1996(E).zip

Java - *The Java Language Specification, Java SE 7 Edition*, Oracle Corporation, February 2013
http://docs.oracle.com/javase/specs/jls/se7/jls7.pdf

PMML - *Predictive Model Markup Language (PMML)*, Data Mining Group, May, 2014
http://www.dmg.org/v4-2-1/GeneralStructure.html

RFC 3986 - RFC 3986: *Uniform Resource Identifier (URI): Generic Syntax*. Berners-Lee, T., Fielding, R., and Masinter, L, editors. Internet Engineering Task Force, 2005.
http://www.ietf.org/rfc/rfc3986.txt

UML - *Unified Modeling Language (UML), v2.4.1*, OMG Document Number formal/2011-08-05, August 2011
http://www.omg.org/spec/UML/2.4.1

XBASE - *XML Base (Second Edition).* Jonathan Marsh and Richard Tobin, editors. World Wide Web Consortium, 2009. http://www.w3.org/TR/xmlbase/

XML - *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, W3C Recommendation 26 November 2008
http://www.w3.org/TR/xml/

XML Schema - *XML Schema Part 2: Datatypes Second Edition*, W3C Recommendation 28 October 2004
http://www.w3.org/TR/xmlschema-2/

XPath Data Model - *XQuery 1.0 and XPath 2.0 Data Model (XDM) (Second Edition)*, W3C Recommendation 14 December 2010
http://www.w3.org/TR/xpath-datamodel/

XQuery and XPath Functions and Operators - *XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition)*, W3C Recommendation 14 December 2010
http://www.w3.org/TR/xpath-functions/XQuery

3.2 Ненормативные источники

JSON - *ECMA-404 The JSON Data Interchange Standard*, European Computer Manufacturers Association, October, 2013
http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

PRR - *Production Rule Representation (PRR)*, Version 1.0, December 2009, OMG document number formal/2009-12-01
http://www.omg.org/spec/PRR/1.0/

RIF - *RIF production rule dialect*, Ch. de Sainte Marie et al. (Eds.) , W3C Recommendation, 22 June 2010.
http://www.w3.org/TR/rif-prd/

SBVR - *Semantics of Business Vocabulary and Business Rules (SBVR)*, V1.2, OMG document number formal/2013-11-04, November 2013
http://www.omg.org/spec/SBVR/1.2/

SQL - ISO/IEC 9075-11:2011, *Information technology -- Database languages -- SQL -- Part 11: Information and Definition Schemas (SQL/Schemata*), International Organization for Standardization, 2011
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=5368

Xpath - *XML Path Language (XPath) Version 1.0*, W3C Recommendation 16 November 1999
http://www.w3.org/TR/xpath

Эта страница намеренно оставлена пустой

4 Дополнительная информация

4.1 Благодарность

Следующие компании представили данную спецификацию:

- Decision Management Solutions;
- Escape Velocity;
- FICO;
- International Business Machines;
- Oracle.

Следующие компании поддержали использование данной спецификации:

- KU Leuven;
- Knowledge Partners International;
- Model Systems;
- TIBCO.

Благодарность выражается членам основной команды, работавшим над содержанием данного документа: Мартин Чепмен, Боб Даниэль, Алан Фиш, Ларри Гольдберг, Джон Холл, Барбара фон Галле, Гэри Хеллмарк, Дейв Ингс, Кристиан де Сент-Мари, Джеймс Тейлор, Ян Вантхиенен, Пол Винсент.

Следующие люди также привнесли ценные идеи и обеспечили обратную связь, что помогло усовершенствовать содержимое и качество данного документа: Бас Янссен, Роберт Ларио, Пит Риветт.

Следующие люди и компании разработали Версию 1.1: Эли Аби-Лахуд, Университетский колледж Корк; Джастин Брант, TIBCO; Алан Фиш, FICO; Джон Холл, Rule ML Initiative; Денис Ганье, Trisotech; Гэри Холлмарк, Oracle; Элиза Кендалл, Thematix Partners LLC; Манфред Кёте, 88solutions; Фалко Менге, Camunda Services GmbH; Збигнев Мисяк, BOC Information Technologies Consulting; Сйир Нийссен, PNA Group; Михаил Попов, MITER; Пит Риветт, Adaptive; Брюс Сильвер, Bruce Silver Associates; Бастиан Штайнерт, Signavio GmbH; Тим Стивенсон, Omny Link; Джеймс Тейлор, Decision Management Solutions; Ян Вантиенен, K.U. Leuven; Пол Винсент, Knowledge Partners, Inc.

Следующие люди и компании разработали Версию 1.2: Алан Фиш, FICO; Денис Ганье, Trisotech; Гэри Холлмарк, Oracle; Элиза Кендалл, Thematix Partners LLC; Манфред Кёте, 88solutions; Фалко Менге, Camunda Services GmbH; Збигнев Мисяк, BOC Products & Services AG; Сйир Нийссен, PNA Group; Октавиан Патраской, Goldman Sachs; Брюс Сильвер, Bruce Silver Associates; Джил Ронен, Sapiens DECISION; Кэролайн Шарф, Tom Sawyer Software; Бастиан Штайнер, Signavio GmbH; Джеймс Тейлор, Decision Management Solutions; Эдсон Тирелли, Red Hat; Ян Вантиенен, K.U. Leuven; Стивен Уайт, Министерство по Делам Ветеранов.

Следующие люди и компании разработали Версию 1.3: Алан Фиш, FICO; Денис Ганье, Trisotech; Гэри Холлмарк, Oracle; Уве Кауфманн, GfSE e.V.; Элиза Кендалл, Thematix Partners LLC; Манфред Кёте, 88solutions; Роберт Ларио, Министерство по Делам Ветеранов; Фалко Менге, Camunda Services GmbH; Збигнев Мисяк, BOC Products & Services AG; Маттео Мортари, Red Hat;Сйир Нийссен, PNA Group; Октавиан Патраской, Goldman Sachs; Брюс Сильвер, Bruce Silver Associates; Джил Сигал, Sapiens Decision NA; Бастиан Штайнер, Signavio GmbH; Джеймс Тейлор, Decision Management Solutions; Эдсон Тирелли, Red Hat; Ян Вантиенен, K.U. Leuven; Стивен Уайт, Министерство по Делам Ветеранов.

4.2 Право на объект интеллектуальной собственности и патенты

Перечисленные выше лица передали данную работу OMG на условиях разумного не дискриминационного лицензирования без уплаты роялти.

4.3. Структура документа

В разделе 1 кратко формулируются цели спецификации.

Раздел 2 определяет три уровня соответствия спецификации: уровень соответствия 1, уровень соответствия 2 и уровень соответствия 3.

В разделе 3 перечислены ссылки на нормативные источники.

В разделе 4 представлена дополнительная информация, полезная для общего понимания структуры спецификации.

В разделе 5 обсуждается область применения **DMN** и вводятся основные концепции, в том числе два уровня **DMN**: уровень требований к принятию решений и уровень логики решения.

Раздел 6 определяет уровень требований к принятию решений в **DMN**: График требований к принятию решений (ГТР), а также Диаграмма требований к принятию решений (ДТР).

В разделе 7 представлены принципы, по которым логика принятия решения может быть связана с элементами ГТР, то есть, описывается, каким образом уровень требований к принятию решений и уровень логики решения связаны друг с другом.

Затем разделы 8, 9 и 10 определяют уровень логики решения в **DMN**:

- раздел 8 определяет нотацию и синтаксис таблиц решений в **DMN**;
- раздел 9 описывает S-FEEL: подмножество FEEL для поддержки таблиц решений;
- раздел 10 определяет полный синтаксис и семантику FEEL: языка выражений по умолчанию, который используется на уровне логики решений в **DMN**.

В разделе 11 приведен примеры использования **DMN** для моделирования принятия решений человеком и машиной.

Раздел 12 описывает форматы обмена и предоставляет ссылки на машиночитаемые файлы (XSD и XMI).

В Приложениях приводится ненормативная справочная информация:

- в Приложении A обсуждается взаимосвязь между **DMN** и **BPMN**;
- Приложение B содержит глоссарий терминов.

5 Введение в DMN

5.1 Контекст

Нотация **DMN** преследует цель предоставить конструкции, необходимые для моделирования решений, что позволит легко отобразить процесс принятия решений в организации при помощи точно определяемых бизнес-аналитиками диаграмм, и (опционально) автоматизировать принятие решений.

По существующим стандартам моделирования, процесс принятия решений рассматривается с двух разных точек зрения:

- модели бизнес-процессов (например, **BPMN**) могут описывать последовательность действий при принятии решений внутри бизнес-процесса путем определения конкретных задач или действий, в рамках которых происходит принятие решений;

- логика принятия решения (например, PRR, PMML) может определять конкретную логику, используемую для принятия индивидуальных решений, например, бизнес-правила, таблицы принятия решений или исполняемые аналитические модели.

Тем не менее, ряд авторов (включая членов группы, создавших данную спецификацию) обратили внимание на то, что процесс принятия решений имеет внутреннюю структуру, которую трудно воспроизвести, используя какой-либо из перечисленных подходов к моделированию. **DMN,** связывая между собой модели бизнес-процессов и логические модели принятия решений, стремится обеспечить иной подход, заключающийся в создании диаграммы требований к принятию решений:

- модели бизнес-процессов определяют процессные задачи, в рамках которых требуется принять решения;

- в диаграммах требований к принятию решений определяются решения, которые должны быть приняты в рамках упомянутых задач, их взаимосвязи и требования к логике принятия решений;

- логика принятия решения определяет необходимые решения достаточно подробно, чтобы обеспечить возможность проверки и/или автоматизации.

Диаграммы требований к принятию решений вместе с логикой принятия решений представляют собой полную модель принятия решения, которая дополняет модель бизнес-процесса, подробно определяя процесс принятия решений, выполняемый в рамках процессных задач. Отношения между этими тремя аспектами моделирования показаны на рисунке 5.1.

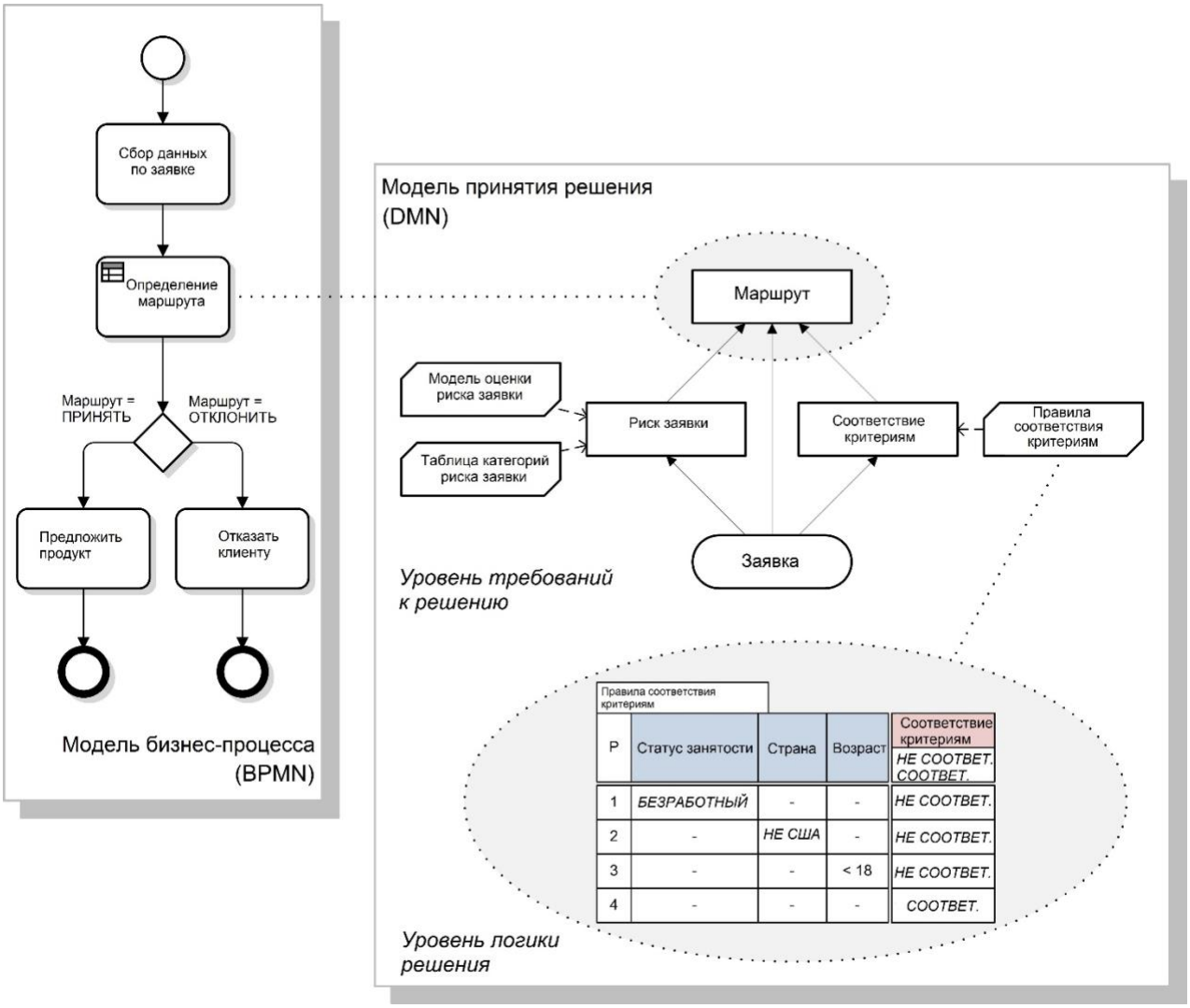


Рисунок 5.1. Аспекты моделирования

В результате набор связанных моделей позволит более детально отобразить роль бизнес-правил и аналитических моделей в бизнес-процессах, провести перекрестную проверку моделей, осуществить проектирование и автоматизацию сверху вниз, реализовать автоматическое выполнение решений. Например, система управления бизнес-процессами будет вызывать службу принятия решения, развернутую из системы управления бизнес-правилами.

Хотя на рисунке 5.1 показана взаимосвязь между моделью бизнес-процесса и моделью принятия решений, демонстрирующая взаимодействия **DMN** с другими стандартами, необходимо подчеркнуть, что **DMN** не зависит от **BPMN**. Два уровня **DMN** – требования к принятию решений и логика принятия решений – могут использоваться независимо или совместно для моделирования области принятия решений без ссылки на бизнес-процессы (см. 5.2.).

DMN предлагает конструкции, охватывающие как моделирование требований к принятию решений, так и моделирование логики принятия решений. В первом случае применяется График требований к принятию решений (ГТР), который включает в себя набор элементов и определяет правила их соединения, а также соответствующая нотация: Диаграмма требований к принятию решений (ДТР). Для моделирования логики принятия решения в **DMN** используется язык FEEL, с помощью которого определяются и создаются таблицы решений, производятся вычисления, задается логика if/then/else, простые структуры данных и внешняя логика Java и PMML и преобразуются в исполняемые выражения с формально определенной семантикой. Кроме того, **DMN** предлагает нотацию для описания логики принятия решения («табличные

выражения»), которая позволяет графически связать компоненты уровня логики решения с элементами диаграммы требований к принятию решений. Связь между этими конструкциями показана на рисунке 5.2.

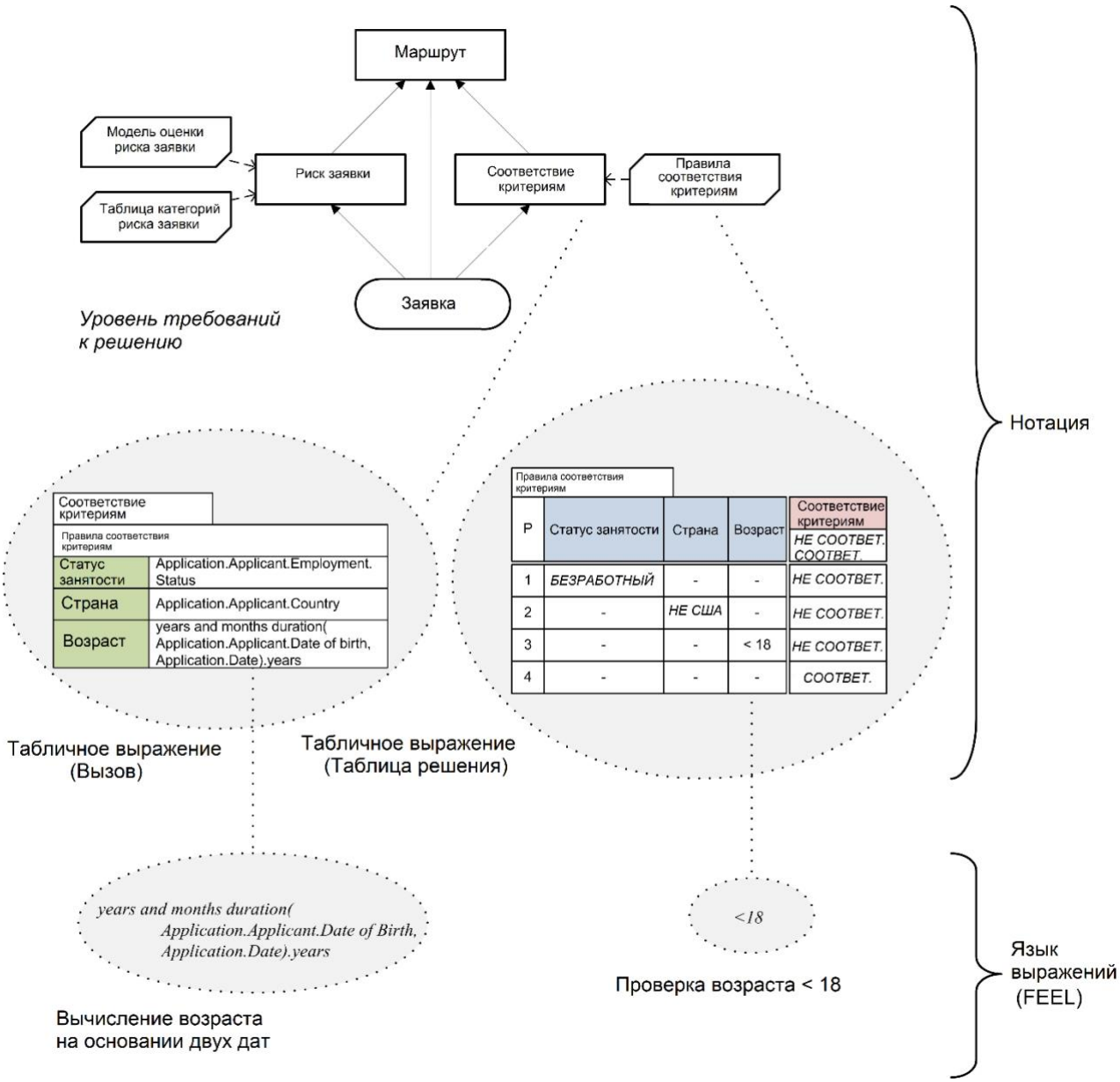


Рисунок 5.2. Конструкции DMN

5.2 Область применения DMN

Моделирование процесса принятия решений позволяет бизнес-аналитикам лучше понять и определить, каким образом организация принимает решения. Как правило, это операционные решения, принимаемые в повседневных бизнес-процессах, а не стратегические решения, для которых существует меньшее количество правил и представлений.

С учетом вышесказанного можно выделить три области применения **DMN**:

1. Моделирование процессов принятия решений людьми.
2. Моделирование требований к автоматизированному процессу принятию решений.
3. Внедрение автоматизированных решений.

5.2.1 Моделирование процессов принятия решений людьми

DMN может использоваться для моделирования решений, принимаемых персоналом внутри организации. Процесс принятия решений человеком может быть разбит на несколько составляющих и смоделирован при помощи ДТР. Решения на ДТР, вероятно, будут описаны на довольно высоком уровне, с использованием естественного языка, а не логики принятия решений.

Можно определить источники знаний, что позволит отразить на модели, каким образом люди (например, менеджеры), регулирующие органы (например, омбудсмен), документы (например, регламенты) или законодательные акты (например, законоположения) регулируют процесс принятия решений. Источники знаний могут быть связаны друг с другом, например, чтобы показать, что решение регулируется: (а) набором правил, определенных органом надзора, и (б) регламентами компании, поддерживаемыми менеджером.

Модели бизнес-знаний отображают конкретные области бизнес-знаний, которые используются при принятии решений. Это позволяет использовать **DMN** в качестве инструмента для формального определения требований к управлению знаниями. Несколько моделей бизнес-знаний могут быть связаны между собой с целью демонстрации взаимозависимости между областями знаний (подобно тому, как это сделано в существующей методике построения структуры знаний). Источники знаний могут связываться с моделями бизнес-знаний. Это делается для того, чтобы продемонстрировать, каким образом бизнес-знания управляются или поддерживаются. Например, чтобы показать, что набор бизнес-регламентов (модель бизнес-знаний) определен в регламенте компании (источник знаний).

В некоторых случаях возможно определить конкретные правила или алгоритмы принятия решений. Они могут быть смоделированы с использованием логики принятия решений (например, с помощью бизнес-правил или таблиц решений). Это позволяет описать модели бизнес-знаний на ДТР:

а) дескриптивно: чтобы зафиксировать, каким образом принимаются решения в настоящее время или в течение определенного периода;
б) директивно: чтобы определить, каким образом следует принимать решения или как они будут приниматься в будущем.

Процесс принятия решений, смоделированный в **DMN**, может сопоставляться с задачами или операциями бизнес-процесса, смоделированного в **BPMN**. На высоком уровне задача совместного принятия решений может быть сопоставлена с подмножеством решений на ДТР, представляющем общее поведение группы или отдела при принятии решений. На более детализированном уровне можно моделировать взаимозависимости между решениями, принимаемыми рядом отдельных лиц или групп, используя **BPMN**: каждый участник процесса принятия решений представлен отдельным пулом в **BPMN** и отдельной ДТР на модели решений. Решения на этих ДТР затем сопоставляются с задачами в пулах, а входные данные на ДТР сопоставляются с содержимым сообщений, передаваемых между пулами.

Таким образом, совместное использование **BPMN** и **DMN** позволяет при помощи графического языка описать несколько уровней принятия решений людьми внутри организации, начиная с операций в бизнес-процессах и заканчивая подробным описанием логики принятия решений. В этом контексте модели **DMN**

описывают совместные организационные решения, управление процессами принятия решений и необходимые бизнес-знания.

5.2.2 Моделирование требований к автоматизированному принятию решений

Моделирование требований к автоматизированному принятию решений аналогично моделированию требований к принятию решений людьми. Однако требования к автоматизированному принятию решений являются полностью директивными, а не дескриптивными, кроме того, больше внимания уделяется детальному описанию логики принятия решений.

Для полной автоматизации процесса принятия решений, логика принятия решения должна быть полной, то есть способной предоставить результат решения для любого возможного набора значений входных данных.

Тем не менее, наиболее широко используется частичная автоматизация. В этом случае прерогатива принимать определенные решения остаётся за персоналом компании.

Взаимодействие между автоматизированными процессами принятия решений и процессами принятия решений людьми отображается на модели при помощи **BPMN** и **DMN**, как это было указано выше. В этом случае разные пулы обозначают людей, ответственных за принятие решений, и автоматизированные блоки принятия решений. Иными словами, операции принятия решений распределяются между задачами в модели бизнес-процесса, где решения, принимаемые людьми, соотносятся с пользовательскими задачами, а автоматизированные операции принятия решений – с бизнес-правилами. Так, например, при исполнении автоматизированного бизнес-правила, решение может быть передано на рассмотрение человеку-рецензенту; логика решения для автоматизированной задачи должна быть задана полностью, но решение рецензента можно не указывать заранее.

После того как решения на ДТР будут сопоставлены с задачами в бизнес-процессе, можно осуществлять проверку на двух уровнях моделей. Например, можно проверить, что все входные данные на ДТР предоставляются предыдущими задачами в бизнес-процессе и что бизнес-процесс использует результаты решений только в последующих задачах или шлюзах.

При помощи **DMN** моделируются отношения между решениями и бизнес-процессами. Это позволяет определить, какие решения необходимо принять для завершения процесса и в каких конкретных задачах решения принимаются и выполняются. Не предполагается формальное сопоставление **DMN** `ItemDefinition` или **DMN** `InputData` с **BPMN** `DataObject`, однако, если такое сопоставление было реализовано, инструмент моделирования может предусматривать возможность проверки соответствия объектов.

BPMN вкупе с **DMN** позволяет определить требования к автоматизированному принятию решений и описать, каким образом в рамках одного бизнес-процесса автоматизированные решения взаимодействуют с решениями, принимаемыми людьми. Эти требования могут быть определены на любом уровне детализации. Автоматизированные инструменты моделирования могут поддерживать заданные требования благодаря трехуровневому сопоставлению моделей бизнес-процессов, ДТР и логики принятия решений.

5.2.3 Внедрение автоматизированных процессов принятия решений

Если все решения и модели бизнес-знаний полностью определены с использованием логики принятия решений, модели принятия решений становятся исполняемыми.

Одним из возможных сценариев является использование «служб решений», развернутых на базе системы управления бизнес-правилами (**BRMS**) и вызванных системой управления бизнес-процессами (**BPMS**). Служба принятия решений инкапсулирует логику решения, поддерживающую ДТР, предоставляя интерфейсы, которые соответствуют подмножествам входных данных и решениям на ДТР. Служба принятия решений вызывается набором входных данных, после чего она оценивает указанные решения и возвращает их результаты. Ограничение **DMN**, исключающее наличие побочных эффектов в логике принятия решений, означает, что службы принятия решений будут соответствовать принципам SOA, что

упростит проектирование системы. Обратите внимание, службы решений могут быть вызваны изнутри модели принятия решений, это их общее свойство с моделями бизнес-знаний.

Структура модели принятия решений, как показано на DTP, может быть использована в качестве основы для планирования проекта внедрения. В проект могут быть включены отдельные задачи, в рамках которых определяется логика принятия решения (например, обнаружения правил экспертами или создание аналитических моделей) и реализуются компоненты модели принятия решений.

Часть логики принятия решений, представляющая бизнес-знания, инкапсулированные в службы принятия решений, должна постоянно поддерживаться персоналом, ответственным за принятие решений, при помощи специальных «интерфейсов обслуживания знаний». **DMN** поддерживает эффективный дизайн и позволяет внедрять интерфейсы обслуживания знаний: любые бизнес-знания, требующие обслуживания, должны быть представлены как модели бизнес-знаний на DTP, а ответственный персонал – как источники знаний. Затем DTP определяет необходимые интерфейсы обслуживания знаний и их пользователей, а логика принятия решения определяет начальную конфигурацию поддерживаемых бизнес-знаний.

Другая часть логики принятия решения обновляется путем регулярного аналитического моделирования. Представление моделей бизнес-знаний как функций в **DMN** делает использование аналитических моделей в службах принятия решений очень простым: любая аналитическая модель, которую можно представить, как функцию, может быть непосредственно вызвана или импортирована в службу принятия решений.

5.2.4 Объединение приложений для моделирования

Три описанных выше сценария использования **DMN** не являются взаимоисключающими альтернативами; в одном большом проекте автоматизации процессов **DMN** может использоваться для реализации всех трех сценариев.

Во-первых, при моделировании процесса принятия решений в рамках существующего процесса определяется вся полнота принятия решений и области бизнес-знаний. Такой фактический анализ “as-is” является основной для улучшения процессов.

Затем процесс может быть переработан для достижения максимальной эффективности принятия решений людьми и автоматизированными системами, при этом часто между ними устанавливается взаимодействие (например, операция автоматически переадресовывается исполнителям – людям или системам поддержки принятия решений, которые консультируют пользователя или ограничивают его действия). Такая переработка включает в себя моделирование требований к принятию решений в каждой задаче процесса, а также определение ролей и обязанностей отдельных лиц или групп в организации. Полученная модель является “to-be” спецификацией процесса и координируемой процедуры принятия решений.

Сравнение моделей “as-is” и “to-be” определяет требования не только к средствам автоматизации, но и к управлению изменениями: изменения в ролях и обязанностях персонала и обучение для поддержки новых или измененных бизнес-знаний.

Наконец, модель “to-be” будет реализована как исполняемое системное программное обеспечение. При условии, что логика решения полностью определена в FEEL и/или другой внешней логике (например, с использованием внешних методов Java или моделей PMML), компоненты модели принятия решений могут быть реализованы непосредственно в качестве компонентов программного обеспечения.

DMN не предписывает какой-либо конкретной методологии для осуществления вышеуказанных действий; спецификация поддерживает исключительно модели, используемые в этих действиях.

5.3 Основные понятия

5.3.1 Уровень требований к принятию решений

Существует два общепринятых определения для слова «решение»: оно может означать процесс выбора между несколькими возможными вариантами действия или результат выбора. В этой спецификации мы придерживаемся следующего определения: **решение** – это процесс определения **выходного значения** (выбранного варианта) из числа **входных значений** с использованием логики, определяющей, каким образом это происходит. **Логика решения** может включать в себя одну или несколько **моделей бизнес-знаний**, которые инкапсулируют бизнес-ноу-хау в форме бизнес-правил, аналитических моделей или других данных. Базовая структура, на основе которой построены все модели решений, показана на рисунке 5.3.

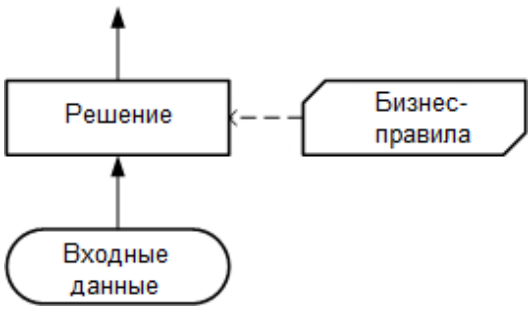


Рисунок 5.3. Базовые элементы модели принятия решений

Для удобства восприятия на многих рисунках в данной спецификации показаны решения, использующие только одну связанную модель бизнес-знаний. Однако это не является обязательным согласно **DMN**. Использование моделей бизнес-знаний для инкапсуляции логики принятия решений является вопросом стиля и методологии, поэтому решения могут быть смоделированы с использованием нескольких моделей бизнес-знаний или без таковых. Как и модели бизнес-знаний, службы решений так же можно использовать инкапсуляции логики принятия решений с целью повторного использования внутри модели принятия решения. Однако, для удобства, такие примеры будут показаны в разделе, посвященном службам решений.

Для моделей принятия решений или моделей знаний можно назначить источник компетенции, в качестве которого могут выступать, например, эксперты в соответствующей области, которые будут определять и поддерживать данные модели; или первичная документация, на основе которой проектируются модели бизнес-знаний; или ряд тестовых сценариев, которым не должны противоречить принятые решения. Они называются **источниками знаний** (см. рисунок 5.4).

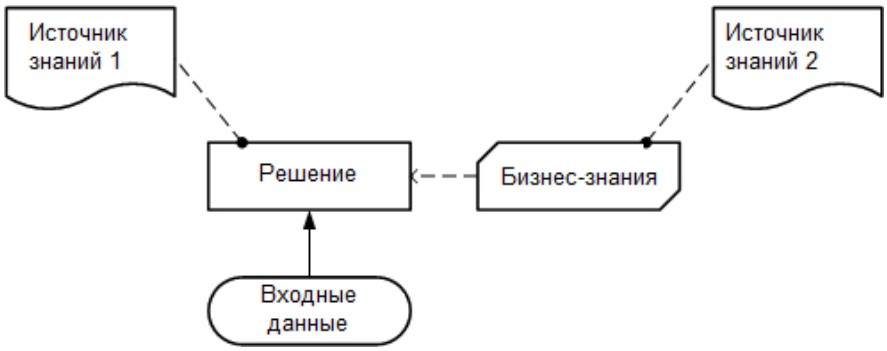


Рисунок 5.4. Источники знаний

Принято говорить, что для определения выхода решению «требуется» наличие входов. Это могут быть **входные данные** или результаты других решений (в обоих случаях, это будут скорее структуры данных, а не просто элементы данных). Если входное значение для Решения 1 включает выходное значение Решения 2, то для принятия Решения 1 требуется принять Решение 2. Следовательно, эти решения можно связать в единую схему, так называемый **График требования к принятию решений** (**ГТР**), который, в свою очередь, может быть выполнен в виде **Диаграммы требования к принятию решений** (**ДТР**). ДТР представляет собой набор решений, зависимых друг от друга, от входных данных и от моделей бизнес-знаний. Простой пример **ДТР** с двумя решениями показан на рисунке 5.5.

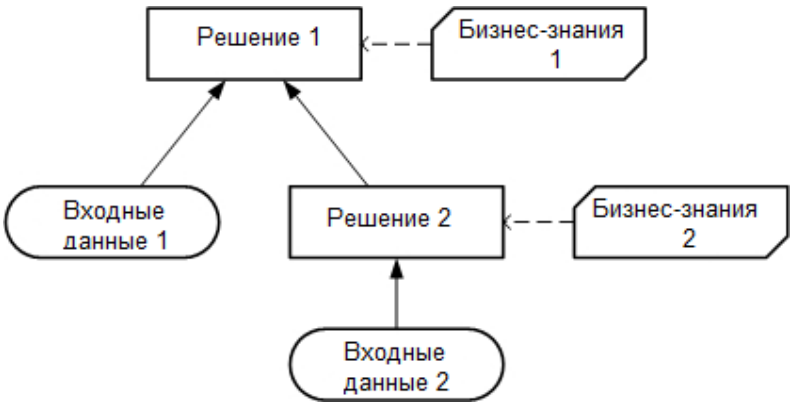


Рисунок 5.5. Простая диаграмма требований к принятию решений (ДТР)

Для принятия решения может потребоваться несколько моделей бизнес-знаний, а для модели бизнес-знаний могут потребоваться несколько других моделей бизнес-знаний, как показано на рисунке 5.6. Это позволяет, например, моделировать сложную логику принятия решений путем объединения различных областей бизнес-знаний и предоставления альтернативных вариантов логики решения в разных ситуациях.

Рисунок 5.6. Объединение нескольких моделей бизнес-знаний

Подробное описание графиков требования к принятию решений и диаграмм требований к принятию решений приводится в разделе 5.6.

5.3.2 Уровень логики решений

Компоненты уровня требований к принятию решения модели решения могут описываться, как показано выше, с использованием только бизнес-концепций. Часто бывает достаточно описать требования на этом уровне для того, чтобы провести бизнес-анализ области принятия решений, определить, какие именно бизнес-решения должны быть приняты, их взаимосвязь, выделить требуемые области бизнес-знаний и данных, источники бизнес-знаний. Используя логику принятия решения, одни и те же компоненты могут быть описаны более подробно, что позволяет в полной мере зафиксировать все бизнес-правила и вычисления и (при желании), полностью автоматизировать процесс принятия решений.

Логика принятия решения также может содержать дополнительную информацию о том, как отображать элементы в модели принятия решений. Например, элемент логики решения для таблицы принятия решений может определять, каким образом отображать правила – в виде строк или столбцов. Элемент логики решения для вычислений может определять, следует ли выравнивать термины по вертикали или по горизонтали.

Соотношение между концепциями на уровне требований к принятию решений и уровне логики решения описано ниже. Обратите внимание, что на рисунках ниже так же, как и на рисунках 1 и 2, серые эллипсы и пунктирные линии используются лишь для обозначения соответствий между концепциями на разных уровнях в целях повышения наглядности. Эти элементы не являются частью нотации **DMN**, описание которой приводится в разделах 6.2, 8.2, 10.2. Предполагается, что ПО, использующие данную нотацию, обеспечат возможность для перемещения между уровнями моделирования при помощи таких операций, как «открытие», «детализация» или «масштабирование», однако **DMN** не дает каких-либо указаний, каким образом это должно происходить.

На уровне логики решения каждое решение на ГТР определяется с помощью **выражения значения**, которое указывает, каким образом выходное значение вычисляется на основании входных значений. На этом уровне решение *считается* результатом проверки выражения. Выражение значения можно отобразить при помощи **табличного выражения**, как показано на рисунке 5.7.

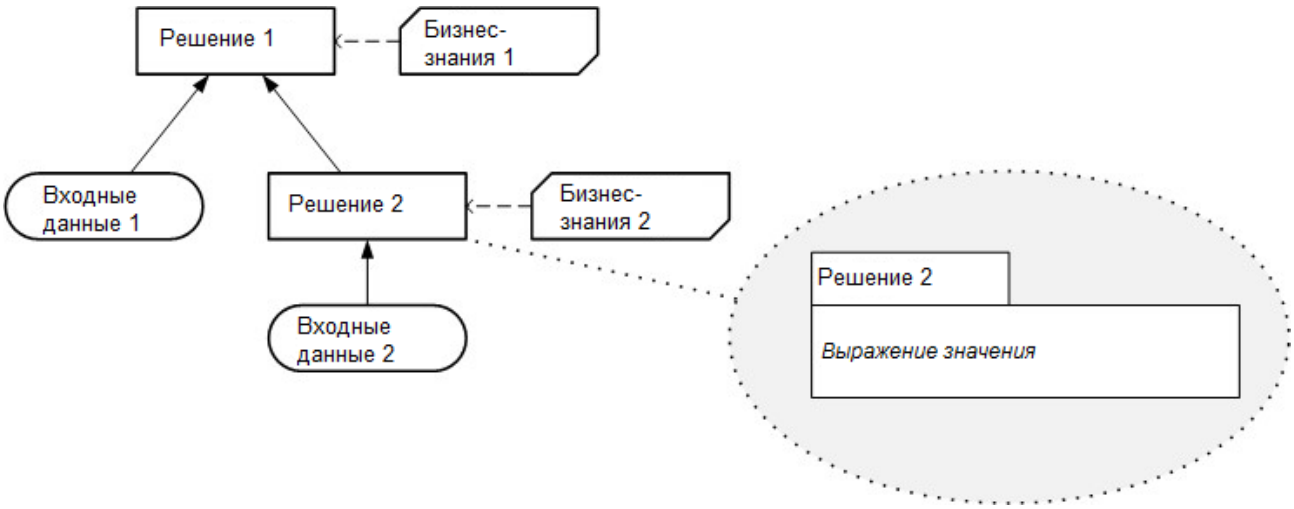


Рисунок 5.7. Решение и соответствующее выражение значения

Подобным образом на уровне логики решения модель бизнес-знаний определяется с помощью выражения значения, которое указывает, каким образом выходное значение задается, исходя из входных значений. В модели бизнес-знаний выражение значения инкапсулируется как определение функции, которая может быть вызвана из выражения значения решения. Интерпретация моделей бизнес-знаний как функций в **DMN** означает, что комбинация моделей бизнес-знаний, как на рисунке 5.6, имеет четкую семантику функционального состава. Выражение значения модели бизнес-знаний можно отобразить с помощью **табличной функции**, как показано на рисунке 5.8. Как и модель бизнес-знаний, элемент сервиса решений может быть вызван из выражения значения решения (см. раздел 5.3.3).

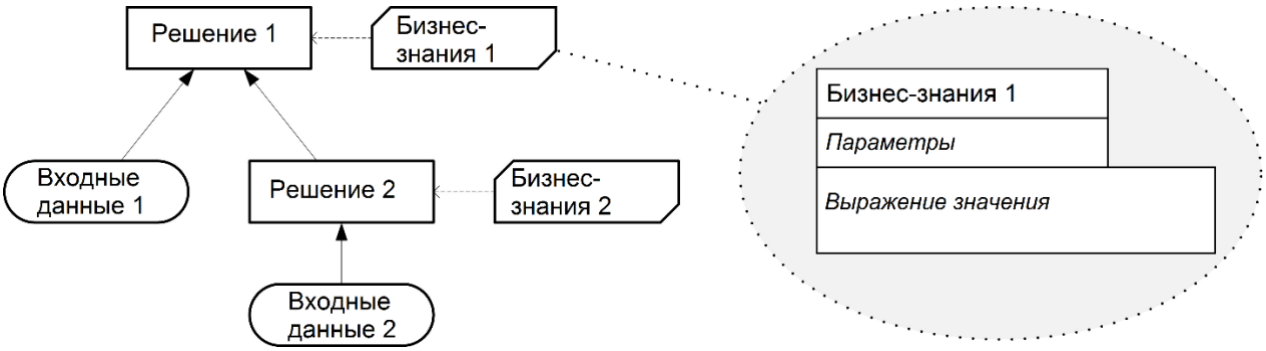


Рисунок 5.8. Модель бизнес-знаний и соответствующее выражение значения

Модель бизнес-знаний может содержать любую логику решения, которую можно представить в виде функции. Это позволит импортировать в **DMN** многие существующие стандарты моделирования логики принятия решений (например, бизнес-правила и аналитические модели). Важным форматом бизнес-знаний, особо поддерживаемым **DMN**, является Таблица принятия решений. Модель бизнес-знаний может отображаться при помощи таблицы принятия решений, как показано на рисунке 5.9.

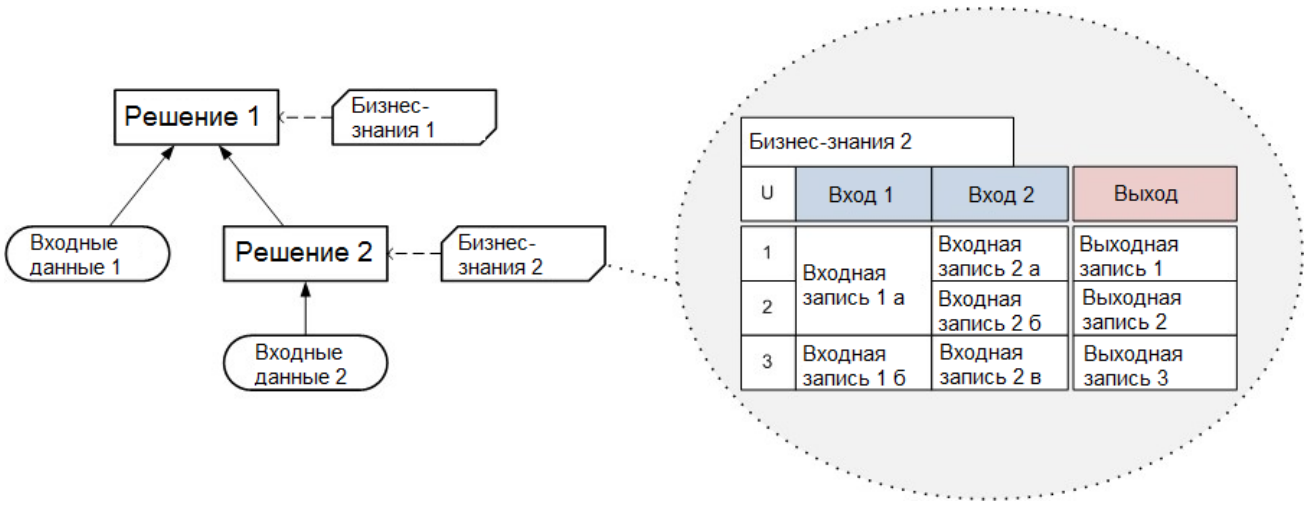


Рисунок 5.9. Модель бизнес-знаний и соответствующая Таблица принятия решений

В большинстве случаев логика решения инкапсулируется в модели бизнес-знаний, а выражение значения, связанное с решением, определяет, как вызываются модели бизнес-знаний и как результаты их вызовов объединяются для вычисления результата решения. Выражение значения решения также может указывать, как выходные значения определяются исходя из набора входных значений, без вызова модели бизнес-знаний: в этом случае с решением не связывается ни одна модель бизнес-знаний (ни на уровне требований решения, ни на уровне логики решения).

Язык выражений для определения логики принятия решения в **DMN**, охватывающий все вышеперечисленные понятия, описывается в разделе 10. Это специальный язык **FEEL**: Friendly Enough Expression Language. Таблицы принятия решений подробно описаны в разделе 8.

5.3.3 Служба принятия решений

Служба решений определяет переиспользуемую логику внутри модели принятия решений. Служба решения представляет одно или несколько решений из модели принятия решений в виде многократно используемого элемента, службы, которая может быть вызвана, например, изнутри, другим решением модели принятия решений, либо извне - задачей модели бизнес-процесса BPMN. При наличии необходимых входных данных и результатов принятия решений служба возвращает результаты решений. Любая служба принятия решений, инкапсулирующая модель решения DMN, не фиксирует состояние и не имеет побочных эффектов.

Одним из важных аспектов применения DMN является определение логики решений, автоматизируемой при помощи «служб решений». Когда служба решения вызывается извне, она может быть реализована, например, как веб-сервис. DMN не указывает, каким образом такие службы должны быть реализованы, но позволяет определить функциональность службы для модели принятия решения. Следовательно, служба решения должна быть определена в ДТР. При вызове изнутри, служба решения вызывается, как и модель бизнес-знаний, путем привязки выражений в логике вызывающего решения к параметрам в вызываемой службе.

Предположим, что клиенту требуется принять ряд решений, а служба создана для удовлетворения этого требования. Единственная функция службы принятия решений – вернуть результаты оценки этого ряда решений («выходные решения»). Службе могут быть переданы результаты решений, оцениваемых внешними клиентами («входные решения»). Служба должна инкапсулировать не только результирующие решения, но и любые решения на ГТР, прямо или косвенно заданные выходными решениями, которые не указаны во входных решениях («инкапсулированные решения»).

Интерфейсы службы принятия решений включает в себя:

- Входные данные: экземпляры всех входных данных, необходимых для инкапсулированных решений;
- Входные решения: экземпляры результатов всех входных решений;
- Выходные решения: результаты оценки всех выходных решений с использованием предоставленных входных решений и входных данных.

При наличии входных данных и входных решений, служба возвращает выходные решения.

Обратите внимание, что для работы службы принятия решений требуется указать только выходные решения и входные либо инкапсулированные решения. Остальные атрибуты (требуемые входные данные и входные или инкапсулированные решения не были указаны) затем могут быть выведены из модели принятия решения для которой служба была создана.

В качестве альтернативы, если определено больше атрибутов, чем это строго необходимо, они могут быть проверены на соответствие модели принятия решения.

На рисунке 5.10 показана служба принятия решений для модели принятия решения, которая включает в себя три решения. Выходные решения для этой службы – {Решение 1}, а входные решения – {}, то есть служба возвращает результат Решения 1 и не получает результатов каких-либо внешних решений. Поскольку для Решения 1 требуется Решение 2, которое не предоставляется службе в качестве входных данных, служба также должна инкапсулировать Решение 2. Решение 3 не требуется инкапсулировать. Поэтому инкапсулированные решения являются {Решением 1, Решением 2}. Для работы службы требуются Входные данные 1 и Входные данные 2, но не требуются Входные данные 3.

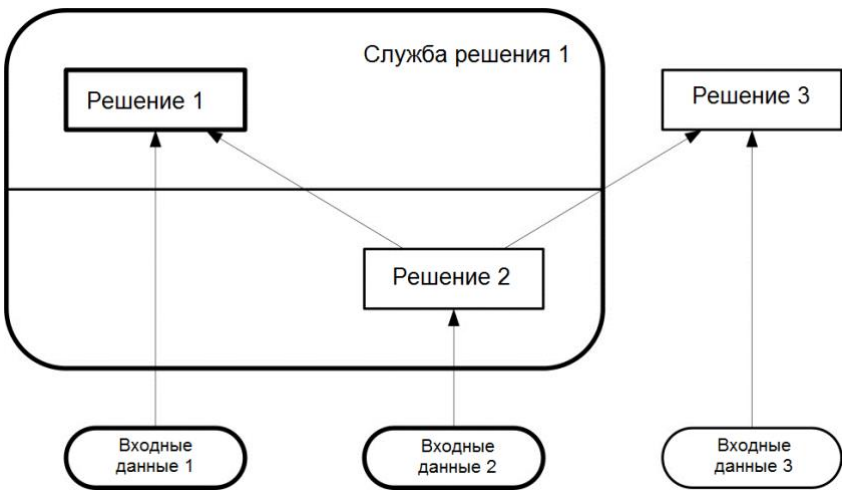


Рисунок 5.10. Служба принятия решения

Для одной и той же модели принятия решения можно использовать несколько служб принятия решений. На рисунке 5.11 показана служба принятия решения для аналогичной модели решения, выходные решения которой являются {Решением 1}, а входные решения - {Решением 2}. Инкапсулированные решения для этой службы - {Решение 1}. Для работы службы требуются входные данные 1, но не требуются входные данные 2 или входные данные 3.

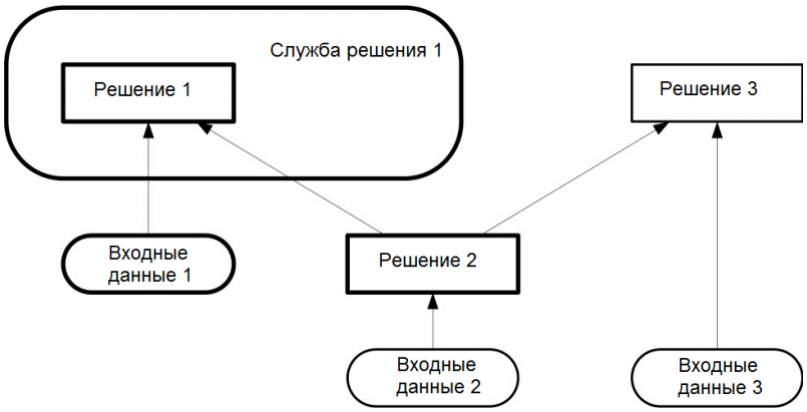


Рисунок 5.11. Служба принятия решения, для которой входным значением является входящее решение

Служба принятия решений в своей простейшей форме всегда оценивает решения, входящие в ряд выходных множеств, и возвращает их результаты.

Для обеспечения вычислительной эффективности можно представить различные улучшения этой базовой интерпретации, например,

- необязательный входной параметр, определяющий список «запрошенных решений» (подмножество минимального набора выходных данных). Только результаты запрошенных решений будут возвращены в контексте вывода;

- необязательный входной параметр, определяющий список «известных решений» (подмножество инкапсулированных наборов данных) с их результатами. Служба принятия решений не будет оценивать эти решения, но будет использовать предоставленные входные значения напрямую.

Поставщики программного обеспечения самостоятельно определяют, какие подобные опции будут включены в производимое ими ПО.

Служба принятия решений является «полной», если она содержит логику принятия решения для оценки всех инкапсулированных решений по всем возможным значениям входных данных. Запрос в службу является «корректным», если экземпляры предоставляются для всех входных решений и входных данных, обязательных для тех решений, которые должны быть оценены, т. е. (в простом случае) для всех инкапсулированных решений или (при условии наличия дополнительных параметров) для любых запрошенных решений и их обязательных вспомогательных решений, которые еще не известны.

Эта страница намеренно оставлена пустой.

6 Требования (ГТР и ДТР)

6.1 Введение

В нотации **DMN** уровень требований к решению на модели принятия решений представлен в виде графика требований к принятию решений (ГТР), который отображается при помощи одной или нескольких диаграмм требований к принятию решений (ДТР).

С помощью ГТР моделируется область принятия решения, которая содержит наиболее важные элементы принятия решений и зависимости между ними. Такими элементами моделирования являются решения, области бизнес-знаний, источники бизнес-знаний, входные данные ислужбы решений:

- элемент **Решение** обозначает процесс определения выходного значения н а основе входных значений с использованием логики принятия решения, которая может ссылаться на одну или несколько моделей бизнес-знаний;
- элемент **Модель бизнес-знания** обозначает функцию, инкапсулирующую бизнес-знания, например, бизнес-правила, таблицу решений или аналитическую модель;
- элемент **Входные данные** обозначает информацию, используемую в качестве входных данных при принятии одного или нескольких решений;
- элемент **Источник знаний** обозначает источник полномочий для модели бизнес-знаний или решения;
- Элемент **Службы решения** обозначает набор переиспользуемых решений, которые могут быть вызван как изнутри, так и извне.

Зависимости между этими элементами выражаются в виде трех видов требований: к информации, знаниям и полномочиям:

- **Требование к информации** обозначает входные данные или результат решения, используемые в качестве входа для решения;
- **Требование к знаниям** обозначает вызов модели бизнес-знаний или службы решения с помощью логики решения;
- **Требование к полномочиям** обозначает зависимость элемента ГТР от другого элемента ГТР, который выступает в качестве источника руководства или знаний.

ДТР может также содержать любое количество артефактов, представляющих аннотации к диаграмме:

- текстовая аннотация – поясняющий текст, добавленный пользователем, создавшим модель;
- ассоциация – пунктирная соединительная линия, связывающая текстовую аннотацию с элементом ГТР.
- группа – неформальный способ визуально сгруппировать элементы диаграммы.

Эти компоненты представлены в таблице 1 и более подробно описаны в разделе 6.2 «Нотация».

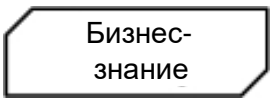
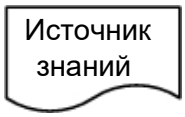
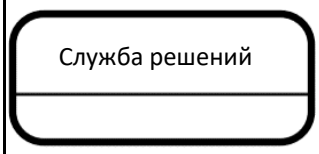
ГТР – это график, состоящий из элементов моделирования, соединенных между собой требованиями. Он является самодостаточным в том смысле, что все моделируемые требования к любому Решению на ГТР (прямые источники информации, знаний и полномочий) указываются на том же самом ГТР. Важно отличать ГТР от ДТР. График отражает решение во всей его полноте, в то время как ДТР дает представление о какой-

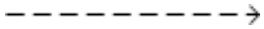
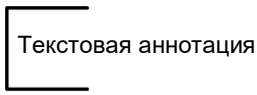
то его части, т.е. представляет собой частичное или отфильтрованное отображение. Более подробно см. 6.2.4 «Частичное отображение и скрытая информация».

6.2 Нотация

Графические обозначения для всех компонентов ДТР представлены в таблице 1 и подробно описаны ниже.

Таблица 1. Компоненты ДТР

Компонент		Описание	Графическое обозначение
Элементы	Решение	Решение – это процесс определения выходного значения из числа входных значений с использованием логики принятия решения, которая может ссылаться на одну или несколько моделей бизнес-знаний.	
	Модель бизнес-знания	Модель бизнес-знания обозначает функцию, инкапсулирующую бизнес-знания, например, бизнес-правила, таблицу решений или аналитическую модель.	
	Входные данные	Входные данные обозначают информацию, используемую в качестве входных данных при принятии одного или нескольких решений. При размещении внутри модели знаний этот элемент обозначает параметры модели знаний.	
	Источник знаний	Источник знаний обозначает источник полномочий для модели бизнес-знаний или решения.	
	Служба решения (развернутая)	Служба решения может включать в себя используемые повторно решения (не показанные в элементе, изображенном справа), которые могут быть вызваны как другим решением изнутри, так и извне, например, бизнес-процессом **BPMN**.	
	Служба решения (свернутая)	Служба решения определяет набор используемых повторно решений (которые можно спрятать, используя элмент, изображенный справа)	

Требования	Требование к информации	Требование к информации обозначает входные данные или результат решения, которые используются в качестве одного из входов решения.	
	Требование к знаниям	Требование к знаниям обозначает вызов модели бизнес-знаний с помощью логики решения.	
	Требование к полномочиям	Требование к полномочиям обозначает зависимость элемента ДТР от другого элемента ДТР, который выступает в качестве источника руководства или знаний.	
Артефакты	Текстовая аннотация	Текстовая аннотация – это поясняющий текст, добавленный пользователем, создавшим модель.	
	Ассоциация	Ассоциация связывает текстовую аннотацию с элементом ГТР, который требует объяснения или комментария.	
	Группа	Группа используется для неформальной группировки элементов и изображается в виде прямоугольника с закругленными углами, выполненного пунктирной линией	

6.2.1 Элементы ДТР

6.2.1.1 Графическое обозначение решения

Решение изображается на ДТР в виде прямоугольника. Контур фигуры выполняется, как правило, сплошной линией (см. таблицу 1). Название решения ДОЛЖНО отображаться внутри фигуры, за исключением случаев, когда его перекрывает текстовый атрибут связанного с ним элемента DMNDI:DMNLabel, который в таком случае и ДОЛЖЕН отображаться вместо названия.

Если используется опция «Перечисленные входные данные» (см. 6.2.1.3 «Входные данные»), все требования для входных данных ДОЛЖНЫ перечисляться под лейблом Решения и отделяться от него горизонтальной линией, как показано на рисунке 6.1.
Названия перечисленных входных данных ДОЛЖНЫ находиться внутри элемента ДТР.

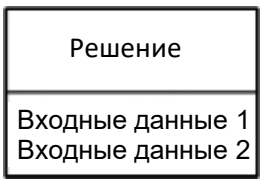


Рисунок 6.1. Решение с перечислением входных данных

Свойства Решения описаны в разделе 6.3.8 «Метамодель бизнес-контекста».

6.2.1.2 Графическое обозначение модели бизнес-знаний

Модель бизнес-знаний изображается на ДТР в виде прямоугольника с двумя срезанными краями. Контур фигуры обычно выполняется сплошной линией, как показано в таблице 1. Название модели бизнес-знаний ДОЛЖНО отображаться внутри фигуры, за исключением случаев, когда его перекрывает текстовый атрибут связанного с ним элемента DMNDI:DMNLabel, который в таком случае и ДОЛЖЕН отображаться вместо названия.

Свойства Модели бизнес-знания описаны в разделе 6.3.8 Метамодель бизнес-контекста.

6.2.1.3 Графическое обозначение Входных данных

Элемент Входные данные изображается на ДТР в виде фигуры с двумя параллельными прямыми сторонами и двумя полукруглыми краями. Контур фигуры обычно выполняется сплошной линией, как показано в таблице 1. Название элемента Входные данные ДОЛЖНО отображаться внутри фигуры, за исключением случаев, когда его перекрывает текстовый атрибут связанного с ним элемента DMNDI:DMNLabel, который в таком случае и должен отображаться вместо названия.

Альтернативный способ отображения требований к входным данным, особенно полезный при моделировании больших и сложных ДТР, заключается в том, что входные данные не отображаются как отдельные графические элементы ДТР, а вместо этого перечисляются в тех элементах принятия решений, которые используют эти данные. В данной спецификации такой способ для удобства называется перечисление входных данных.
Программы для моделирования МОГУТ предусматривать такой способ отображения входных данных. На рисунке 6.2 показаны две эквивалентные ДТР, при этом на одной ДТР изображен элемент Входные данные, а на другой проиллюстрирован вариант с использованием Перечисления входных данных. Обратите внимание, если при построении модели принятия решений элемент Входные данные не используется, НЕОБХОДИМО перечислять входные данные во всех Решениях, где эти данные используются (если только элемент не был намеренно скрыт, см. раздел 6.2.4. «Частичное отображение и скрытая информация»).

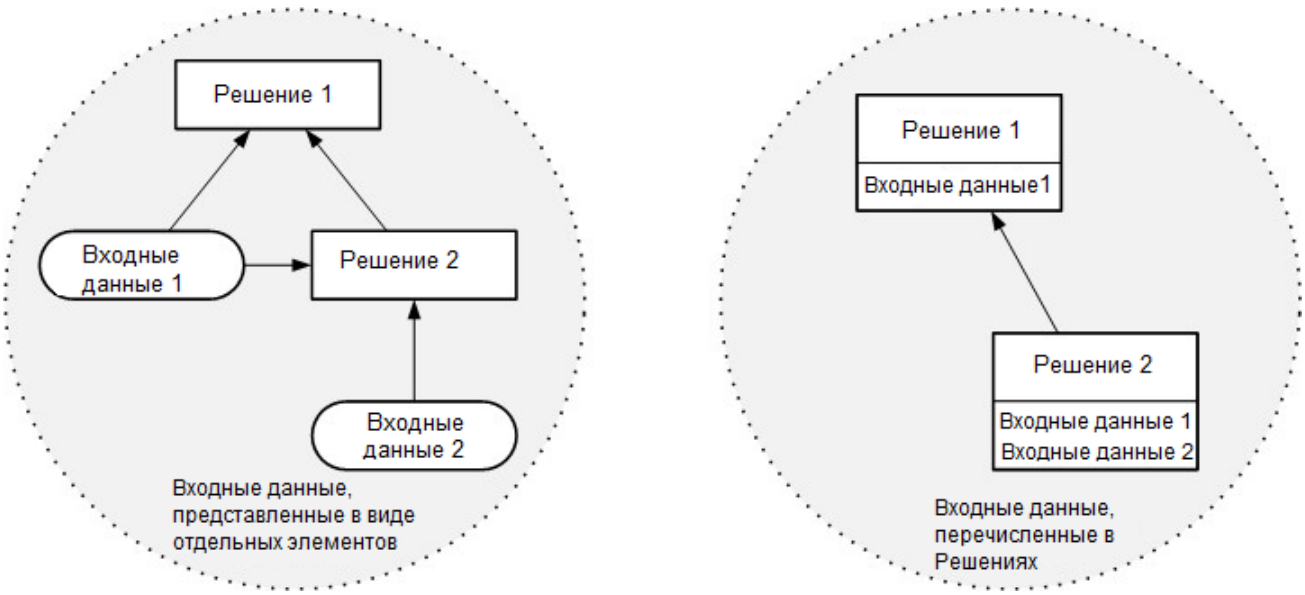


Рисунок 6.2. Перечисление входных данных

Свойства элемента Входные данные описаны в разделе 6.3.11 Метамодель входных данных.

6.2.1.4 Графическое обозначение Источника знаний

Источник знаний изображается на ДТР в виде фигуры с тремя прямыми сторонами и одной волнообразной. Контур фигуры изображается сплошными линиями, как показано в таблице 1. Название Источника знаний ДОЛЖНО отображаться внутри фигуры, за исключением случаев, когда его перекрывает текстовый атрибут связанного с ним элемента DMNDI:DMNLabel, который в таком случае и должен отображаться вместо названия.

Свойства Источника знаний описаны в разделе 6.3.12.

6.2.2 Требования ДТР

6.2.2.1 Графическое обозначение Требования к информации

Требование к информации применяется для связи элементов в направлении от Входных данных к Решению и от Решения к другим Решениям. Данный элемент демонстрирует зависимость Решения от информации, передаваемой при помощи Входных данных, или от результатов других Решений. Требование к информации также можно интерпретировать как поток данных: ДТР, отображающая только решения, входные данные и требования к информации, аналогична диаграмме потока данных, которая демонстрирует передачу информации между этими элементами во время оценки. Требования к информации корректно составленного ГТР формируют направленный ациклический граф.

Требование к информации изображается на ДТР в виде стрелки, выполненной сплошной линией, как показано в таблице 1. Стрелка рисуется в направлении потока информации, то есть к Решению, требующему информации.

6.2.2.2 Графическое обозначение Требования к знаниям

Элемент Требования к знаниям применяется для связи элементов по направлению от вызываемых элементов (Моделей бизнес-знания или Служб решения) к Решению, а также от вызываемых элементов к Модели бизнес-знаний. Они представляют собой вызов вызываемого элемента во время принятия решения. Если e это решение или Модель бизнес-знания в каком-либо ДТР, при этом e содержит требование к знаниям для вызываемого элемента b, то логика e должна содержать выражение вызова b, включая выражения для каждого из параметров b.

Требование к знаниям изображается на ДТР в виде стрелки, выполненной пунктирной линией, с наконечником без заливки, как показано в таблице 1. Стрелка рисуется в направлении потока информации (результата оценки функции), то есть к элементу, который требует бизнес-знаний.

6.2.2.3 Графическое обозначение Требования к полномочиям

Элемент Требования к полномочиям может использоваться двумя способами:

a) для связи элементов в направлении от Источника знаний к Решению, Модели бизнес-знаний и другим Источникам знаний, демонстрируя зависимость элемента ДТР от источника знаний. В этом случае Требование к полномочиям фиксирует факт соответствия набора бизнес-правил опубликованному документу (например, законодательному акту или регламентам предприятия). Также Требование к полномочиям демонстрирует, что конкретное лицо или организационная группа несут ответственность за определение логики какого-либо решения или что процесс принятия решения управляется лицом или группой. Пример такого использования Источников знаний показан на рисунке 6.3, который демонстрирует модель бизнес-знаний. Данной модели требуется два источника полномочий – регламент и законодательство, а регламент требует полномочий группы, разрабатывающей регламент;

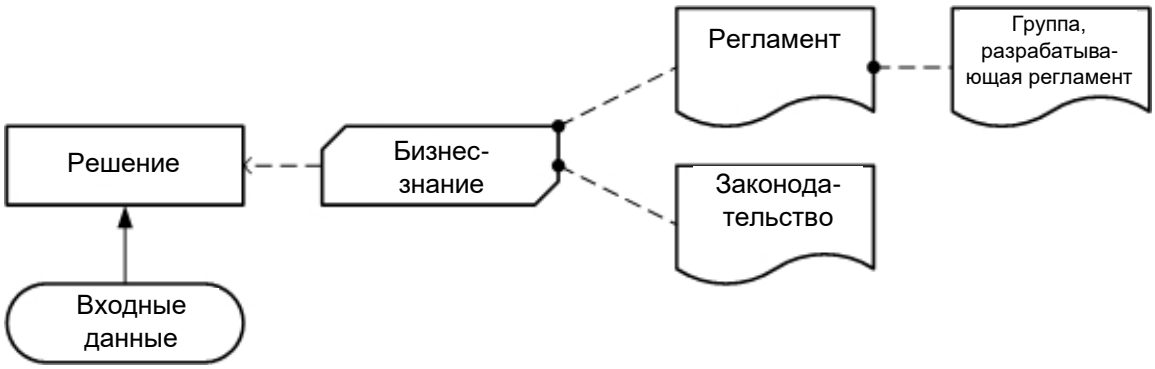


Рисунок 6.3. Источник данных, представляющий полномочия

б) для связи элементов в направлении от Входных данных к Источникам знаний. В этом случае, наряду с применением, описанным в пункте (а), Требование к полномочиям демонстрирует, каким образом Модель бизнес-знаний выводится из экземпляров Входных данных и Результатов решений. Источник знаний обычно представляет собой аналитическую модель (или процесс моделирования); Модель бизнес-знаний представляет собой исполняемую логику, созданную или зависящую от модели. Пример этого использования Источника знаний показан на рисунке 6.4, который демонстрирует модель бизнес-знаний, основанную на аналитической модели, которая в свою очередь выводится из Входных данных и Результатов зависимого решения.

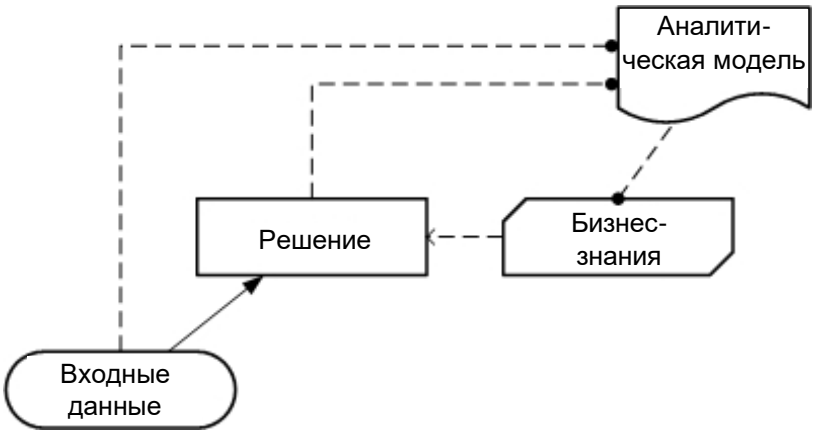


Рисунок 6.4. Источник данных, представляющий прогнозную аналитику

Однако приведенные выше рисунки являются лишь примерами. Существует множество других вариантов использования элемента Требования к полномочиям, и, поскольку Источники знаний и Требования к полномочиям не обладают семантикой выполнения, их интерпретация заведомо является неопределенной. Данная спецификация оставляет за разработчиком право определять, каким именно образом будет использоваться данный элемент.

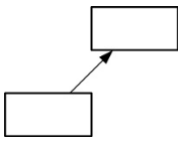
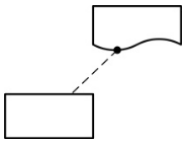
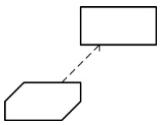
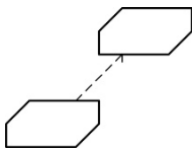
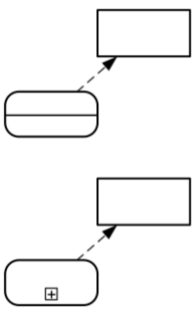
Требование к полномочиям представлено на ДТР в виде стрелки, выполненной пунктирной линией, с наконечником в виде круга со сплошной заливкой (см. таблицу 1). Стрелки изображаются по направлению от источника полномочий к управляемому элементу.

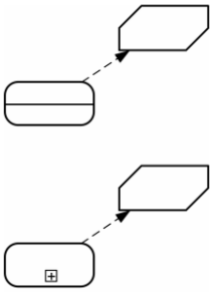
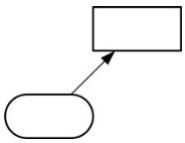
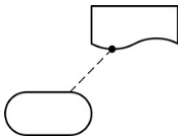
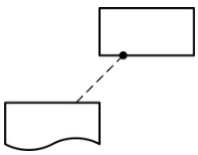
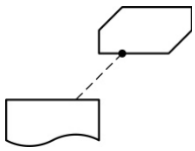
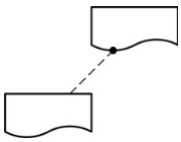
6.2.3 Правила соединения элементов

Правила, регламентирующие допустимые способы соединения элементов с требованиями на ДТР, описаны в разделе 6.2.2 «Требования ДТР» и кратко перечислены в таблице 2. Для наглядности, в качестве пояснения для каждого допустимого способа соединения приводится простая ДТР. На каждой из этих диаграмм верхний элемент («к которому направлено требование») требует нижнего элемента («из которого исходит требование»).

Обратите внимание, что Требования не могут являться входящим элементом для Входных данных, то есть Входные данные не имеют требований. Также следует отметить, что тип требования однозначно определяется типами двух связанных элементов.

Таблица 2. Правила соединения требований

Выходит от	Идет к (требуется)	Требование	Рисунок
Решение	Решение	Информация	
Решение	Источник знаний	Полномочия	
Модель бизнес-знаний	Решение	Знание	
Модель бизнес-знаний	Модель бизнес-знаний	Знание	
Служба решения	Решение	Знание	

Служба решения	Модель бизнес-знаний	Знание	
Входные данные	Решение	Информация	
Входные данные	Источник знания	Полномочия	
Источник знания	Решение	Полномочия	
Источник знания	Модель бизнес-знаний	Полномочия	
Источник знания	Источник знания	Полномочия	

6.2.4 Частичное отображение и скрытая информация

На метамодели (см. раздел 6.3 «Метамодель») отображаются свойства каждого элемента ГТР, которые обычно не представлены на ДТР, но которые дают дополнительную информацию о характере или функции элемента. Например, свойства Решения, определяющие, в каких процессах и задачах **BPMN** используется Решение. Программы для моделирования ДОЛЖНЫ предоставлять инструменты для указания и отображения таких свойств.

Для любой значительной области принятия решения можно СОЗДАВАТЬ большие и сложные диаграммы требований к решению, представляющие собой полный график требований к решению. Программы для моделирования МОГУТ предусматривать инструменты, позволяющее отображать ДТР в виде частичного или отфильтрованного представления ГТР, например, путем сокрытия категорий элементов или сворачивания отдельных областей. Элементы ГТР, чьи требования не отображены в текущем ДТР, должны быть, в связи с этим, обозначены эллипсом (...) , как показано на рисунке 11.5.

На рисунке 6.5 приводятся два примера ДТР с частичным представлением ГТР. ДТР 1 показывает только непосредственные требования одного решения; ДТР 2 демонстрирует только требования к информации и элементы, которые они соединяют.

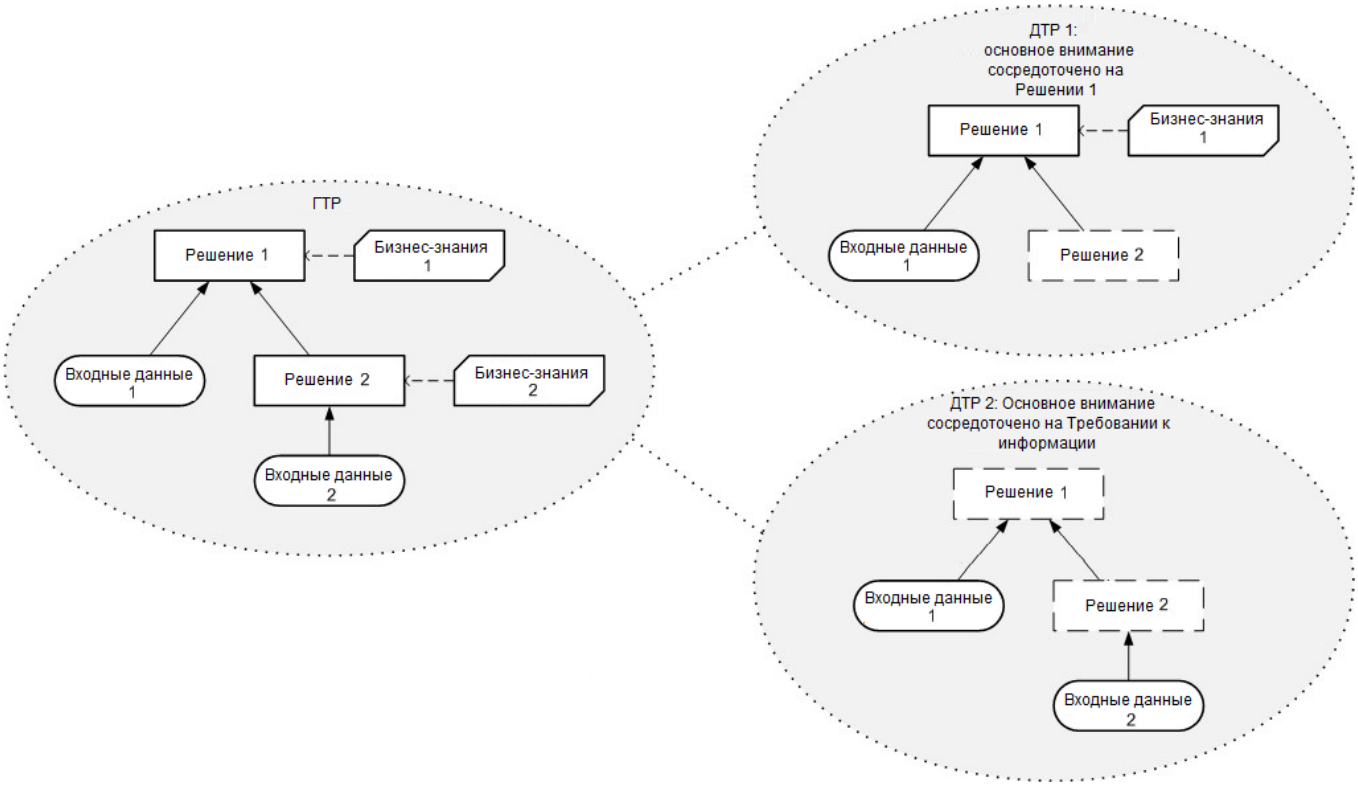


Рисунок 6.5. ДТР как частичное отображение ГТР

ДТР могут быть взаимозаменяемы через механизм обмена диаграмм, описанный в главе 13.

6.2.5 Служба решений

Служба решений изображается на ДТР в виде прямоугольника с закругленными краями. Контур фигуры выполнен полужирной сплошной линией. Название Службы решений ДОЛЖНО отображаться внутри фигуры, за исключением случаев, когда его перекрывает текстовый атрибут связанного с ним элемента DMNDI:DMNLabel, который в таком случае и должен отображаться вместо названия. Все инкапсулированные решения ДОЛЖНЫ размещаться внутри прямоугольника. Другие решения и Входные данные располагаются за его границей. Внутри прямоугольника также МОГУТ располагаться другие элементы ГТР, но они не будут являться частью определения Службы решений.

Если набор выходных решений меньше, чем набор инкапсулированных решений, Служба решений ДОЛЖНА быть разделена на две части прямой сплошной линией. Одна часть ДОЛЖНА включать только выходные решения и название Службы решений; другая часть ДОЛЖНА включать все инкапсулированные решения, которые не входят в набор выходных решений. Любая часть МОЖЕТ включать другие элементы ГТР, но они не будут являться частью определения Службы решений.

На рисунке 6.6 показана Служба решений с двумя выходными решениями; другие примеры (с одним выходным решением) показаны на рис. 5.10 и рис. 5.11.

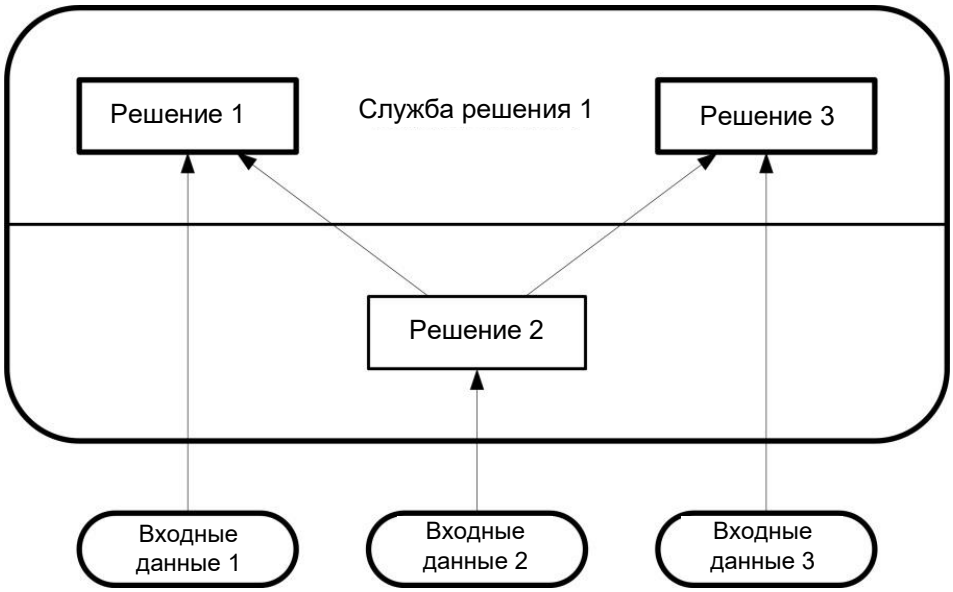


Рисунок 6.6. Обозначение службы решений

Служба решений может быть определена на одной ДТР, а затем показана в другой ДТР, в том случае, если она вызывается внутри модели принятия решений другим решением. Если Служба решения вызывается внутри модели принятия решений, она может изображаться без деталей, то есть, в «свернутом виде». На рисунке 18 показаны две диаграммы. ДТР 1 определяет Службу решений 1. На ДТР 2 изображено, как ту же Службу решений 1 вызывает Решение 5. На ДТР 2 Служба решений показана в свернутом виде.

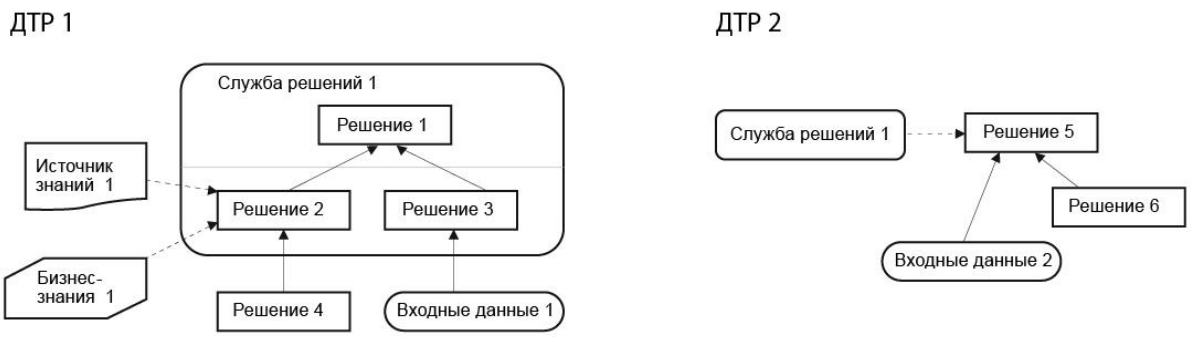


Рисунок 6.7: Служба решений в развернутом и свернутом виде

ДТР 1 на рисунке 6.7 показывает, что у Службы решений 1 есть два входа: Решение 4 и Входные данные 1. Следовательно, у Службы решений 1 есть два входных параметра с характеристиками, соответствующими Решению 4 и Входным данным 1. ДТР 2 на рисунке 18 демонстрирует, что у Решения 5 есть 2 зависимости, но по диаграмме нельзя сказать, являются ли они параметрами для вызова Службы решений 1.

Информация и требования к полномочиям, ассоциированные с Решением 2 в ДТР 1, не изображены в свернутом представлении Службы решений 1, показанной на ДТР 2.

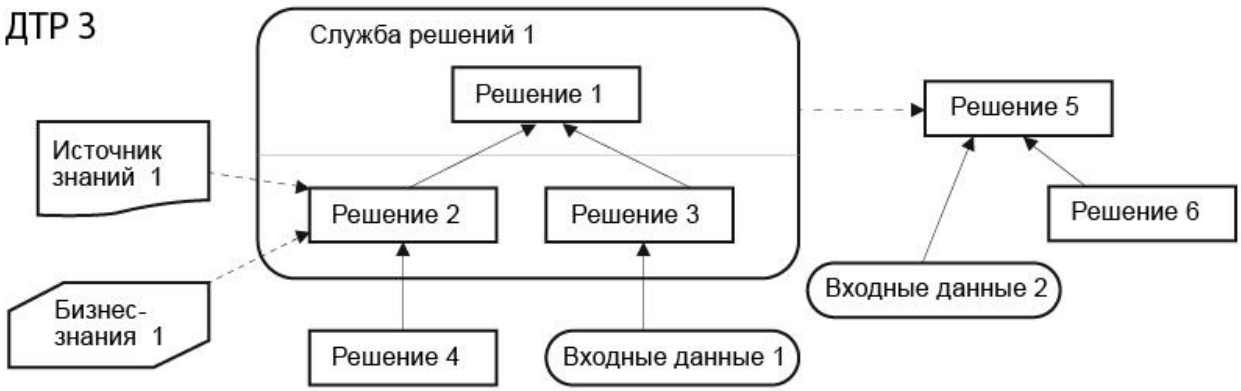


Рисунок 6.8: Служба решений, вызываемая в развернутом представлении

ДТР 1 и 2 на рисунке 6.8, а также ДТР 3 на рисунке 6.9 соответствуют одной и той же ГТР. Все они показывают различные аспекты Службы решений 1. ДТР 3 демонстрирует, как развернутое представление Службы решений 1 вызывается Решением 5.

Ограничение, накладываемое на построение служб решений на ДТР заключается в том, что одна и та же служба решений НЕ ДОЛЖНА изображаться в развернутом и свернутом виде в одной ДТР. Это следует из общего запрета на изображения одного и того же элемента DMN на одной диаграмме.

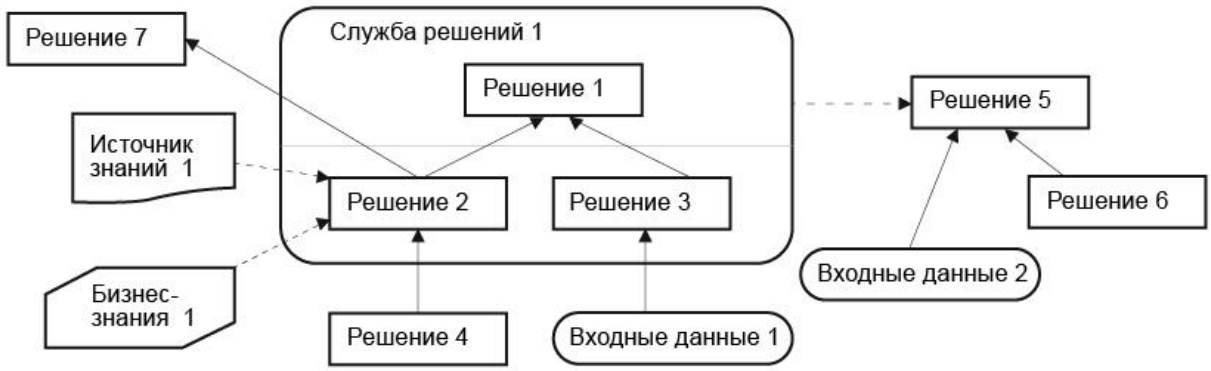


Рисунок 6.9: Служба решения как накладной элемент

Службы решений являются накладными элементами, а значит, не заключают решения внутри себя. Следовательно, многообразие связей, изображенное на рисунке 6.9, является допустимым. На этой ДТР, Решение 7 является зависимым от Решения 2.

6.3 Метамодель

6.3.1 Метамодель элементов DMN

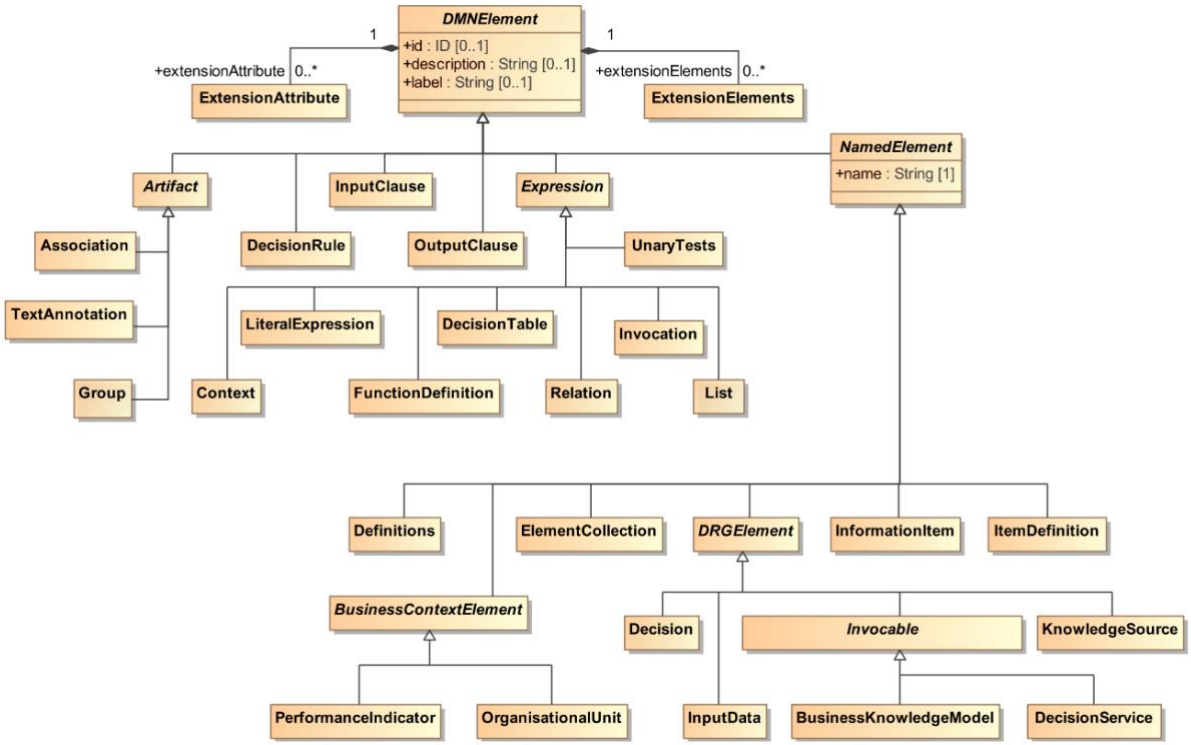


Рисунок 6.10. Диаграмма класса `DMNElement`

`DMNElement` является абстрактным суперклассом для элементов модели принятия решений. Он определяет необязательные строковые атрибуты `id`, `description` и `label`, которые наследуют другие элементы. Идентификатор `id` элемента `DMNElement` дополнительно ограничен синтаксисом XML-идентификатора (http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/datatypes.html#ID) и ДОЛЖЕН быть уникальным в рамках модели принятия решений.

`DMNElement` имеет абстрактные специализации `NamedElement` и `Expression`. `NamedElement` добавляет имя требуемого атрибута и включает абстрактные специализации `BusinessContextElement` и `DRGElement`, а также конкретные специализации `Definitions`, `ItemDefinition`, `InformationItem`, `ElementCollection` и `DecisionService`.

В таблице 3 представлены атрибуты и ассоциации модели элемента `DMNElement`.

Таблица 3. Атрибуты и ассоциации модели `DMNElement`

Атрибут	Описание
id: ID [0..1]	Необязательный идентификатор для этого элемента. ДОЛЖЕН быть уникальным в пределах содержащего его элемента `Definitions`.
description: String [0..1]	Описание данного элемента
label: String [0..1]	Альтернативное короткое описание данного элемента. Оно в первую очередь должно использоваться для элементов без именного атрибута, например, для Входного выражения. Как и атрибут описания, этот

	атрибут не имеет определенного условного обозначения и не относится ни к элементу DMNLabel, используемому в обмене диаграмм, ни к атрибуту outputLabel, принадлежащему таблице принятия решений Decision Table.
extensionElements: ExtensionElement [0..1]	Этот атрибут используется как контейнер для присоединения дополнительных элементов к любому элементу DMN (больше информации о расширяемости приводится в разделе 6.3.16 «Расширяемость»).
extensionAttributes: ExtensionAttribute [0..*]	Этот атрибут используется для присоединения именованных расширенных атрибутов и ассоциаций модели. Такая ассоциация неприменима, если используется обмен XML-схемами, поскольку механизм XSD для поддержки “anyAttribute” из других пространств имен уже удовлетворяет этому требованию (больше информации о расширяемости приводится в разделе 6.3.16 «Расширяемость»).

Таблица 4. Атрибуты и ассоциации модели `NamedElement`

Атрибут	Описание
name: string	Имя данного элемента

6.3.2 Метамодель Definitions

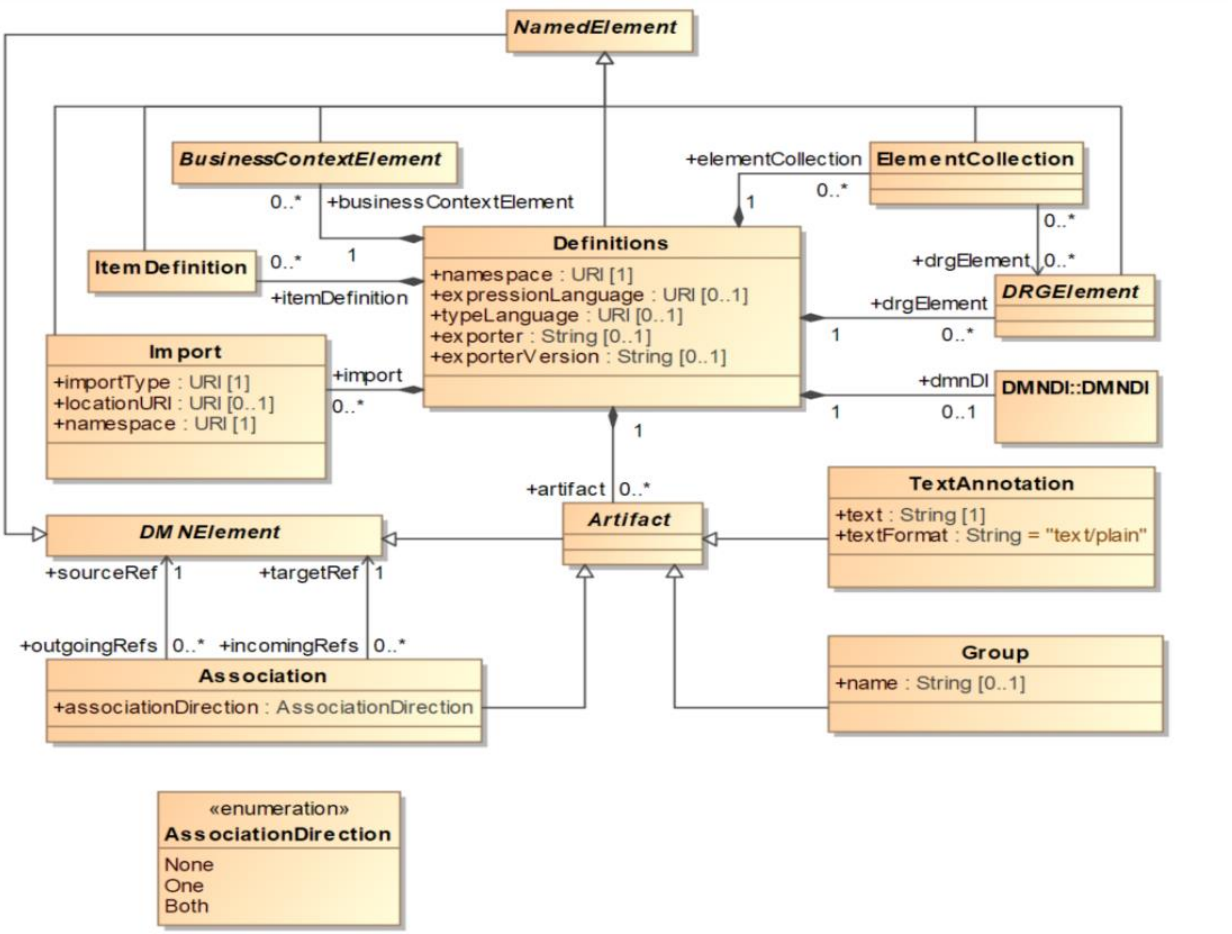


Рисунок 6.11. Диаграмма класса `Definitions`

Класс `Definitions` является внешним контейнером для всех элементов модели решения **DMN**. Он определяет область видимости и пространство имён для всех содержащихся в нем элементов. Элементы, содержащиеся в экземпляре `Definitions`, имеют собственный определенный жизненный цикл и не удаляются при удалении других элементов. Обмен файлами **DMN** всегда осуществляется с использованием одного или нескольких `Definitions`.

`Definitions` – это тип элемента `NamedElement`. Экземпляр `Definitions` наследует имя и дополнительные строковые атрибуты `id`, `description` и `label` у элемента `NamedElement`.

У экземпляра `Definitions` есть пространство имён `namespace` строкового типа. `namespace` определяет целевое пространство имён по умолчанию для элементов, входящих в `Definitions` и соответствует требованиям XML-схемы.

Экземпляр `Definitions` может задавать `expressionLanguage`, который является идентификатором URI, определяющим язык выражения по умолчанию для описания элементов в рамках `Definitions`. Это значение может быть переопределено для каждого отдельного выражения `LiteralExpression`. Язык ДОЛЖЕН быть указан в формате URI. Язык выражения по умолчанию – FEEL (раздел 10), имеет следующий URI: “ https://www.omg.org/spec/DMN/20191111/FEEL/”. Простой язык выражения S-FEEL (раздел 9), являющийся подмножеством FEEL, имеет тот же URI. **DMN** предоставляет URI для языков выражений, которые не должны интерпретироваться автоматически (например, псевдокод, который схож с FEEL, но не является таковым): «http://www.omg.org/spec/DMN/uninterpreted/20140801».

Экземпляр Definitions может задавать typeLanguage, который является идентификатором URI, определяющим язык типа по умолчанию для элементов в рамках данного экземпляра Definitions. Например, значение typeLanguage «http://www.w3.org/2001/XMLSchema» указывает, что структуры данных, определенные в рамках данного определения Definitions, по умолчанию являются типами XML-схемы. Если не указано иное, тип typeLanguage по умолчанию – FEEL. Это значение может быть переопределено для каждого отдельного ItemDefinition. typeLanguage ДОЛЖЕН указываться в формате URI (идентификатор URI для FEEL: “ /https://www.omg.org/spec/DMN/20191111/FEEL”; для того чтобы указать, что определение типа не предназначено для интерпретации, можно использовать URI "http://www.omg.org/spec/DMN/uninterpreted/20140801").

Экземпляр Definitions может задавать строковые атрибуты exporter и exporterVersion, которые дают имя инструменту и версии для создания сериализации XML. В таких стандартах, как **BPMN**, это обеспечивает обмен моделями между инструментами моделирования.

Экземпляр Definitions может включать ноль и более drgElements, которые представляют собой экземпляры DRGElement, ноль или более collections, которые являются экземплярами ElementCollection, ноль или более itemDefinitions, которые являются экземплярами ItemDefinition и ноль или более businessContextElements, которые являются экземплярами BusinessContextElement.

Definitions может содержать любое количество связанных элементов import, которые являются экземплярами Import. Атрибут Import используется для импорта элементов, определенных извне, и делает их доступными для использования в данном определении Definitions.

Элемент Definitions наследует все атрибуты и ассоциации модели от NamedElement. В таблице 5 представлены дополнительные атрибуты и ассоциации модели элемента Definitions.

Таблица 5. Атрибуты Definitions и ассоциации модели

Атрибут	Описание
namespace: anyURI [1]	При помощи данного атрибута определяется пространство имён, ассоциированное с Definitions. Данный атрибут удовлетворяет требованиям языка XML Schema.
expressionLanguage: string [0..1]	При помощи данного атрибута определяется язык выражений, используемый в LiteralExpressions в рамках данного экземпляра Definitions. По умолчанию используется FEEL (см. раздел 10). Это значение МОЖЕТ быть переопределено для каждого отдельного выражения LiteralExpression. Язык ДОЛЖЕН быть указан в формате URI.
typeLanguage: anyURI [0..1]	Данный атрибут определяет язык типа, используемый в LiteralExpressions в рамках данного экземпляра Definitions. По умолчанию используется FEEL (см. раздел 10). Это значение МОЖЕТ быть переопределено для каждого отдельного ItemDefinition. Язык ДОЛЖЕН быть указан в формате URI.
exporter: string [0..1]	Данный атрибут называет инструмент, используемый для экспорта сериализации XML.
exporterVersion: string [0..1]	Данный атрибут называет версию инструмента, используемого для экспорта сериализации XML.

itemDefinition: ItemDefinition [*]	В д атрибуте перечислены экземпляры `ItemDefinition`, которые содержатся в экземпляре `Definitions`.
drgElement: DRGElement [*]	В этом атрибуте перечислены экземпляры `DRGElement`, которые содержатся в экземпляре `Definitions`.
businessContextElement: BusinessContextElement [*]	В этом атрибуте перечислены экземпляры `BusinessContextElement`, которые содержатся в экземпляре `Definitions`.
collection: ElementCollection [*]	В этом атрибуте перечислены экземпляры `ElementCollection`, которые содержатся в данном экземпляре `Definitions`.
decisionService: DecisionService [*]	В этом атрибуте перечислены экземпляры `DecisionService`, которые содержатся в данном экземпляре `Definitions`.
import: Import [*]	Этот атрибут используется для импорта элементов, определенных извне, и делает их доступными для использования элементами в данном экземпляре `Definitions`
artifact: Artifact [0..*]	Артефакты включают текстовые аннотации, группы и ассоциации между элементами DMN.
dmnDI: DMNDI [0..1]	Этот атрибут содержит информацию об обмене диаграмм Diagram Interchange, содержащуюся в `Definitions` (полная информация об обмене диаграмм DMN Diagram Interchange приведена в главе 13).

6.3.3 Метамодель Import

Класс `Import` используется при обращении к внешним элементам, либо к экземплярам **DMN** `DRGElement`, либо к экземплярам `ItemDefinition`, содержащимся в других элементах `Definitions`, либо к элементам, не относящимся к **DMN**, таким как XML-схема или файл PMML. `Import` должен быть явно определен.

Экземпляр `Import` использует строчный атрибут `importType`, который определяет тип импорта, связанного с элементом. Например, значение “http://www.w3.org/2001/XMLSchema” указывает, что импортированный элемент является XML-схемой. Пространство имен **DMN** указывает, что импортированный элемент является элементом **DMN** `Definitions`.

Местоположение импортируемого элемента может быть указано путем сопоставления необязательного `locationURI` с экземпляром `Import`. `locationURI` является URI.

У экземпляра `Import` есть атрибут `namespace`, являющийся идентификатором URI, который в свою очередь определяет пространство имен импортируемого элемента, а также являющийся именем `name`, унаследованным от `NamedElement`, который является строкой, служащей префиксом для полностью квалифицированных имен, таких как typeRefs для импортированных `ItemDefinitions` и выражений, ссылающихся на импортированные `InformationItems`. Значение `namespace`должно быть уникальным на глобальном уровне, но имя `name` для импорта, обычно являющееся коротким и понятным для бизнеса названием, должно отличатся от названий других импортов, решений, входных данных, моделей бизнес-знаний, служб решений и определений элементов только в рамках импортируемой модели.

В таблице 6 представлены атрибуты и ассоциации модели элемента `Import`.

Таблица 6. Атрибуты и ассоциации модели элемента Import

Атрибут	Описание
importType: anyURI	Указывает стиль импорта, связанный с данным экземпляром `Import`.
locationURI: anyURI [0..1]	Определяет местоположение импортируемого элемента.
namespace: anyURI	Определяет пространство имён импортируемого элемента.

6.3.4 Метамодель Element Collection

Класс ElementCollection используется для определения именованных групп экземпляров DRGElement. В программах для моделирования `ElementCollections` могут использоваться для таких целей, как:

- определение подграфика требований одного или нескольких решений (т. е. всех элементов при закрытии требований набора);
- определение элементов, которые будут изображены на ДТР.

`ElementCollection` – это своего рода `NamedElement`, из которого экземпляр `ElementCollection` наследует имя и дополнительные строчные атрибуты `id, description и label`. Идентификатор элемента `ElementCollection` ДОЛЖЕН быть уникальным в рамках `Definitions`.

У элемента `ElementCollection` может быть любое количество связанных `drgElements`, которые являются экземплярами `DRGElement` и определяются `ElementCollection` как группа. Обратите внимание, что элемент `ElementCollection` должен ссылаться на экземпляры `DRGElement`, а не содержать их. Экземпляры `DRGElement` могут содержаться только в элементах `Definitions`.

`ElementCollection` наследует все атрибуты и ассоциации модели от `NamedElement`. В таблице 7 представлены дополнительные атрибуты и ассоциации модели элемента `ElementCollection`.

Таблица 7. Атрибуты и ассоциации модели `ElementCollection`

Атрибут	Описание
drgElement: DRGElement [*]	Данный атрибут перечисляет экземпляры DRGElement, которые группируются элементом ElementCollection.

6.3.5 Метамодель DRG Element

`DRGElement` является абстрактным суперклассом для всех элементов **DMN**, которые содержатся в `Definitions` и которые имеют графическое представление на ДТР. Все элементы модели решения **DMN**, которые не содержатся непосредственно в элементе `Definitions` (в частности, все три вида требований, привязки, разделы и правила принятия решений, импорт и цель), ДОЛЖНЫ содержаться в экземпляре `DRGElement` или в элементе модели, который в свою очередь содержится в экземпляре `DRGElement`, рекурсивно.

Специализациями `DRGElement` являются `Decision`, `InputData`, `Invocable` и `KnowledgeSource`. `Invocable` затем специализируется в `BusinessKnowledge` и `DecisionService`.

`DRGElement` – это специализация `NamedElement`, у которого `DRGElement` наследует имя `name` и дополнительные атрибуты `id, description` и `label`. Идентификатор `id` элемента `DRGElement` ДОЛЖЕН быть уникальным в рамках содержащего его экземпляра `Definitions`.

Диаграмма требований к решению (DTP) представляет собой схематическое изображение одного или нескольких экземпляров DRGElement, связанных требованиями к информации, знаниям и полномочиям. Экземпляры DRGElement представлены в виде вершин на диаграмме, ребра представляют собой экземпляры InformationRequirement, KnowledgeRequirement или AuthorityRequirement (см. 6.3.12 "Метамодель Требования к информации", 6.3.13 "Метамодель Требования к знаниям" и 6.3.14 "Метамодель требования к полномочиям"). Правила соединения элементов описаны в разделе 6.2.3 "Правила соединения элементов").

DRGElement наследует все атрибуты и ассоциации модели от NamedElement. Дополнительные атрибуты и ассоциации модели для элемента DRGElement не определены.

6.3.6 Метамодель Artifact

Элементы Artifacts используются для предоставления дополнительной информации о модели принятия решений. **DMN** поддерживает два стандартных элемента Artifacts: ассоциация Association и текстовая аннотация Text Annotation. Ассоциации Association могут использоваться для связывания Artifacts с любым DMNElement.

6.3.6.1 Ассоциация

Ассоциация Association используется для связывания информации и артефактов с графическими элементами **DMN**. Текстовые аннотации Text Annotation и другие Artifacts могут быть связаны с графическими элементами. Стрелка элемента Association указывает направление потока (например, данные), когда это необходимо.

Элемент Association наследует атрибуты и ассоциации модели DMNElement (см. Таблицу 3). В таблице 8 представлены дополнительные атрибуты и ассоциации модели для Association.

Таблица 8. Атрибуты и ассоциации модели для Association

Атрибут	Описание
associationDirection: AssociationDirection = None {None | One | Both}	Данный атрибут определяет, будет ли использоваться стрелка для того, чтобы задать направленность Association. Значением по умолчанию является None (без стрелки). Значение «One» означает, что стрелка будет находиться на целевом объекте. Значение «Both» означает, что стрелка будет использоваться на обоих концах линии ассоциации Association.
sourceRef: DMNElement[1]	Определяет элемент DMNElement, от которого направлена ассоциация Association.
targetRef: DMNElement[1]	Определяет элемент DMNElement, к которому направлена ассоциация Association.

6.3.6.2 Группа

Группа - это Artifact, механизм для неформальной визуальной группировки элементов диаграммы. **Группы** часто используются для того, чтобы выделить некоторые участки Диаграммы, не добавляя при этом дополнительных ограничений на исполнение. Такой выделенный (сгруппированный) участок Диаграммы может быть отделен для формирования отчета или проведения анализа. **Группы** не влияют на исполнение Решений.

Будучи Artifact, Группа не является DRGElement и, следовательно, не может входить или выходить из Требований информации, Требований знания или Требования полномочий. Она может входить в или

выходить только Ассоциации. Элемент `Group` наследует атрибуты и модельные связи от артефакта `Artifact`. В таблице 9 показаны дополнительные атрибуты и модельные связи для группы `Group`.

Таблица 9: Модельные связи `Group`

Атрибут	Описание
name: String[0..1]	Описательное имя элемента.

6.3.6.3 Текстовая аннотация

Текстовые аннотации – это механизм для разработчика модели, позволяющий предоставить дополнительную текстовую информацию пользователю, читающему диаграммы **DMN**.

Элемент `TextAnnotation` наследует атрибуты и ассоциации модели `DMNElement` (см. Таблицу 3). В таблице 10 представлены дополнительные атрибуты для `TextAnnotation`.

Таблица 10: Атрибуты `TextAnnotation`

Атрибут	Описание
text: string	Данный атрибут представляет собой текст, который разработчик модели желает сообщить читателю диаграммы.
textFormat: string = “text/plain”	Этот атрибут определяет формат текста. Его значение ДОЛЖНО указываться в формате mime-типа. Значением по умолчанию является “text/plain.”

6.3.7 Метамодель Decision

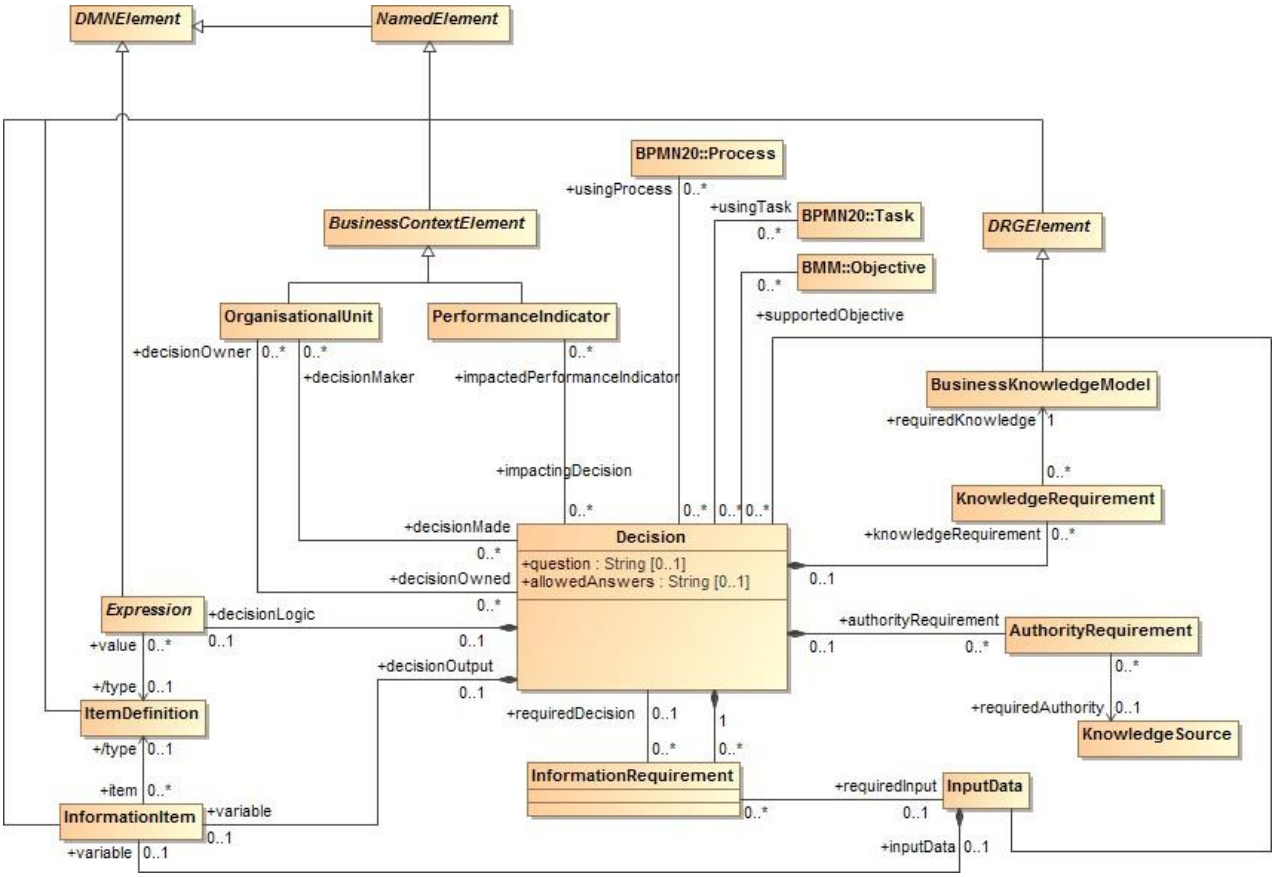


Рисунок 6.12. Диаграмма класса Decision

Класс `Decision` используется для моделирования решения.

`Decision` представляет собой конкретную специализацию `DRGElement` и наследует имя `name` и необязательные атрибуты `id`, `description` и `label` от `NamedElement`. Имя `name`, принадлежащее `Invocable`, должно отличатся от имен других вызываемых элементов, входных данных, решений или импорта в модели принятия решений.

Кроме того, у него могут быть строковые атрибуты `question` и `allowedAnswers`. Необязательный атрибут `description` должен содержать краткое описание процесса принятия решений, воплощенного в `Decision`. Необязательный атрибут `question` должен содержать вопрос, сформулированный на естественном языке, который характеризует `Decision` таким образом, что результат решения является ответом на вопрос. Необязательный атрибут `allowedAnswers` должен содержать сформулированный на естественном языке допустимый ответ на вопросы, требующие ответа Да/Нет, список допустимых значений, диапазон числовых значений и т. д.

На ДТР экземпляр `Decision` представлен таким элементом диаграммы, как **решение**.

В элемент `Decision` входят необязательный атрибут `decisionLogic`, представляющий собой экземпляр `Expression`, а также ноль и более `informationRequirement`, `knowledgeRequirement`, и `authorityRequirement`, которые являются экземплярами `InformationRequirement`, `KnowledgeRequirement` и `AuthorityRequirement` соответственно.

Кроме того, Decision определяет элемент InformationItem, представляющий собой результат решения. InformationItem может включать необязательный атрибут typeRef, который ссылается на определение ItemDefinition или другое определение типа, задающее тип данных возможных результатов Decision.

Подграфиком требований элемента Decision является ориентированный граф, состоящий из самого элемента Decision, требований informationRequirements, knowledgeRequirements и совокупности подграфиков требований каждого элемента requiredDecision или requiredKnowledge: то есть подграфиком требований Decision является закрытие informationRequirement, requiredInput, requiredDecision, knowledgeRequirement и ассоциаций requiredKnowledge, начиная с данного элемента Decision.

Синтаксис экземпляра Decision – то есть модели решения – считается корректным только в том случае, если все элементы informationRequirement или knowledgeRequirement имеют правильную синтаксическую структуру. Как следствие, наличие данного условия предполагает, что подграфик элемента Decision ДОЛЖЕН быть ациклическим, то есть элемент Decision НЕ ДОЛЖЕН прямо или косвенно ссылаться самого себя.

Помимо таких логических компонентов Decision, как требования к информации, логика принятия решений и т. д., модель решения может также документировать бизнес-контекст для принятия решения (см. раздел 6.3.8 "Метамодель бизнес-контекста" и рисунок 21 ниже).

Бизнес-контекст для экземпляра Decision определяется его связью:

- с любым количеством supportedObjectives, которые являются экземплярами Objective, согласно OMG BMM;
- с любым количеством impactedPerformanceIndicators, которые являются экземплярами PerformanceIndicator;
- с любым количеством decisionMaker и decisionOwner, которые являются экземплярами OrganisationalUnit.

Кроме того, экземпляр Decision может ссылаться:

- на любое количество usingProcess, которые, согласно OMG **BPMN 2.0**, являются экземплярами Process, использующими элемент Decision;
- на любое количество usingTask, которые, согласно OMG **BPMN 2.0,** являются экземплярами Task, использующими элемент принятия Decision.

Decision наследует все атрибуты и ассоциации моделей от DRGElement. В таблице 11 представлены дополнительные атрибуты и ассоциации модели класса Decision.

Таблица 11. Атрибуты и ассоциации модели класса Decision.

Атрибут	**Описание**
question: string [0..1]	Сформулированный на естественном языке вопрос, который характеризует Decision таким образом, что результат Decision является ответом на вопрос.
allowedAnswers: string [0..1]	Сформулированный на естественном языке ответ для вопросов, требующих ответа Да/Нет, списка допустимых значений, диапазона числовых значений и т. д.
variable: InformationItem	Экземпляр InformationItem, в котором хранится результат решения.
typeRef: String[0..1]	Указатель на спецификацию типа данных возможных результатов решения либо

	`ItemDefinition`, либо базовый тип в указанном `expressionLanguage`, либо импортированный тип.
decisionLogic: Expression [0..1]	Экземпляр `Expression`, представляющий логику решения.
informationRequirement: InformationRequirement [*]	В этом атрибуте перечислены экземпляры `InformationRequirement`, которые являются частью данного решения.
knowledgeRequirement: KnowledgeRequirement [*]	В этом атрибуте перечислены экземпляры `KnowledgeRequirement`, которые являются частью данного решения.
authorityRequirement: AuthorityRequirement [*]	В этом атрибуте перечислены экземпляры `AuthorityRequirement`, которые являются частью данного решения.
supportedObjective: BMM::Objective [*]	В этом атрибуте перечислены экземпляры BMM:: Objective, которые поддерживаются данным решением.
impactedPerformanceIndicator: PerformanceIndicator [*]	В этом атрибуте перечислены экземпляры PerformanceIndicator, на которые влияет данное решение.
decisionMaker: OrganisationalUnit [*]	Экземпляры OrganisationalUnit, которые принимают это решение.
decisionOwner: OrganisationalUnit [*]	Экземпляры OrganisationalUnit, которые являются владельцами данного решения.
usingProcesses: **BPMN**::process [*]	В этом атрибуте перечислены экземпляры **BPMN**:: process, требующий принятия этого решения.
usingTasks: **BPMN**::task [*]	В этом атрибуте перечислены экземпляры **BPMN**::task, которые принимают это решение.

6.3.8 Метамодель Business Context Element

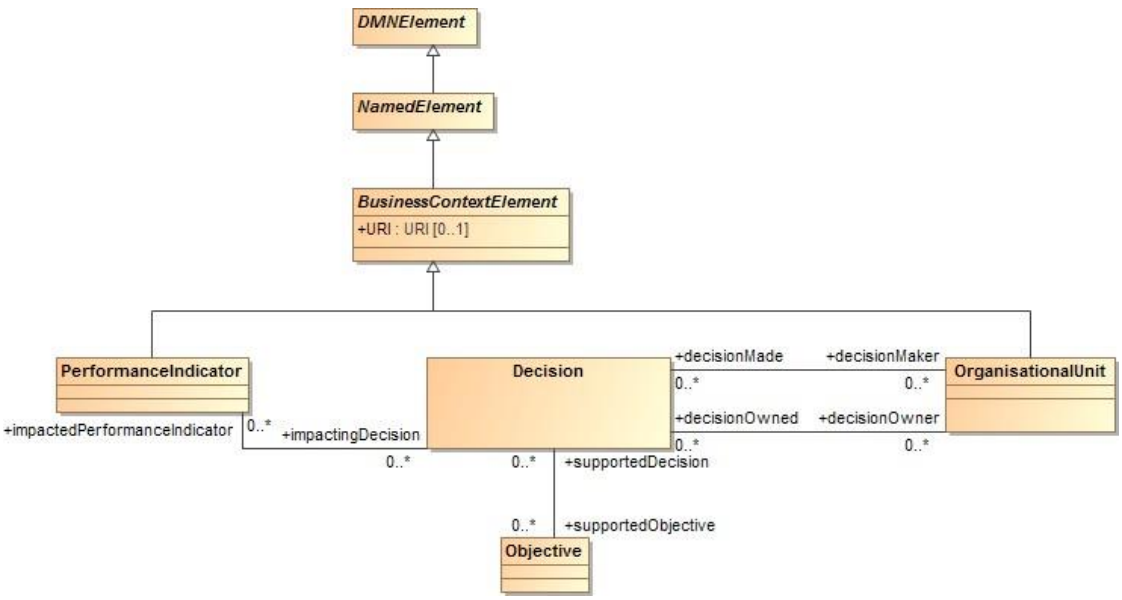


Рисунок 6.13: Диаграмма класса BusinessContextElement

Абстрактный класс BusinessContextElement и его конкретные специализации PerformanceIndicator и OrganizationUnit используются как заполнители до момента получения определения из других метамоделей OMG, таких как OMG OSM.

BusinessContextElement является специализацией элемента NamedElement, от которого он наследует имя name и дополнительные атрибуты id, description и label.

Кроме того, экземпляры BusinessContextElements могут иметь URI, который является универсальным идентификатором ресурсов URI, и

- экземпляром PerformanceIndicator, ссылающимся на любое количество impactingDecision. impactingDecision являются элементами Decision, влияющими на решение;
- экземпляром OrganisationalUnit, ссылающимся на любое количество decisionMade и decisionOwned, которые являются элементами Decision. Данные элементы моделируют решения, принимаемые организационной единицей или принадлежащие ей.

BusinessContextElement наследует все атрибуты и ассоциации моделей от NamedElement. В таблице 12 представлены дополнительные атрибуты и ассоциации модели класса BusinessContextElement.

Таблица 12. Атрибуты и ассоциации модели класса BusinessContextElement.

Атрибут	Описание
URI: anyURI [0..1]	URI данного BusinessContextElement.

PerformanceIndicator наследует все атрибуты и ассоциации моделей от BusinessContextElement. В таблице 13 представлены дополнительные атрибуты и ассоциации модели класса PerformanceIndicator.

Таблица 13. Атрибуты и ассоциации модели класса PerformanceIndicator.

Атрибут	Описание
impactingDecision: Decision [*]	Данный атрибут перечисляет экземпляры Decision, которые влияют на PerformanceIndicator.

OrganisationalUnit наследует все атрибуты и ассоциации моделей от BusinessContextElement. В таблице 14 представлены дополнительные атрибуты и ассоциации модели класса OrganisationalUnit.

Таблица 14. Атрибуты и ассоциации модели класса OrganisationalUnit.

Атрибут	Описание
decisionMade: Decision [*]	Данный атрибут перечисляет экземпляры решения Decision, принятые OrganisationalUnit.

decisionOwned: Decision [*]	Данный атрибут перечисляет экземпляры решения `Decision`, которыми `OrganisationalUnit` владеет.

6.3.9 Метамодель Business Knowledge Model

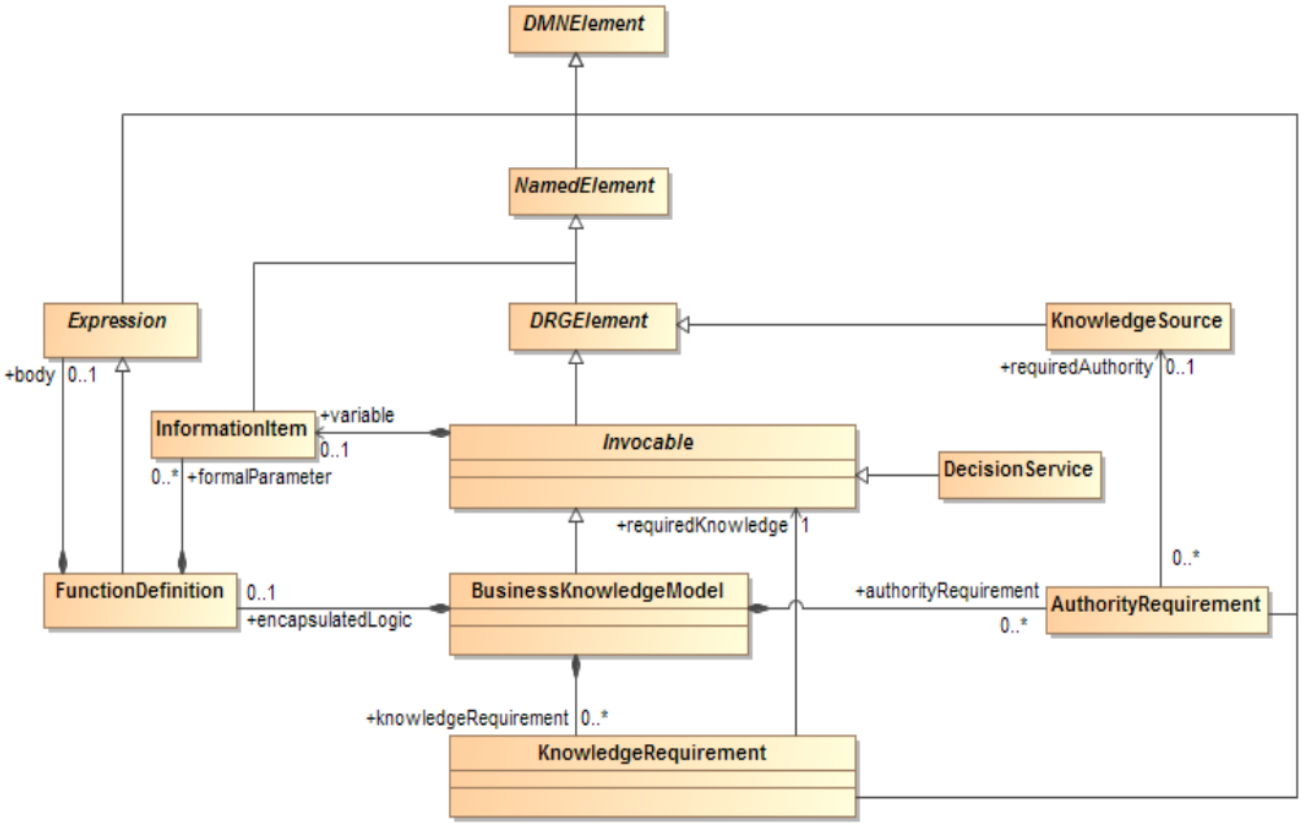


Рисунок 6.14: диаграмма класса BusinessKnowledgeModel

Модель бизнес-знаний имеет абстрактную часть, которая представляет повторно используемую вызываемую логику решения, а также конкретную часть, согласно которой логика решения должна быть единым определением табличной логики FEEL. Служба решений так же является вызываемым элементом и может быть вызвана в качестве требуемого знания из других решений или моделей бизнес-знания.

Класс `Invocable` используется для моделирования вызываемого элемента, а класс `BusinessKnowledgeModel` используется для моделирования модели бизнес-знаний.

`Invocable` – это специализация `DRGElement`, которая наследует имя `name` и дополнительные атрибуты `id, description` и `label` от `NamedElement`. Имя `Invocable` должно отличатся от имен других вызываемых элементов, входных данных, решений или импорта в модели принятия решений. `BusinessKnowledgeModel` – это специализация `Invocable`, от которого она автоматически наследует атрибут `variable`.

Элемент `BusinessKnowledgeModel` может иметь ноль или более `knowledgeRequirement`, которые являются экземпляром `KnowledgeRequirement`, и ноль или более `authorityRequirement`, которые являются экземплярами `AuthorityRequirement`. Эти элементы описаны ниже.

Подграфиком требований элемента `BusinessKnowledgeModel` является ориентированный граф, состоящий из самого элемента `BusinessKnowledgeModel`, требований `knowledgeRequirements` и совокупности подграфиков требований каждого элемента `requiredKnowledge`, на который ссылаются `knowledgeRequirements`.

Синтаксис экземпляра `BusinessKnowledgeModel` считается **корректным** только в том случае, если элементы `knowledgeRequirement` не используются в `BusinessKnowledgeModel` или имеют правильную синтаксическую структуру. Как следствие, наличие данного условия предполагает, что подграфик элемента `BusinessKnowledgeModel` ДОЛЖЕН быть ациклическим, то есть элемент `BusinessKnowledgeModel` НЕ ДОЛЖЕН прямо или косвенно ссылаться на самого себя.

На уровне логики решения элемент `BusinessKnowledgeModel` содержит элемент `FunctionDefinition`, который является экземпляром `Expression`, содержащим ноль и более параметров, являющихся экземплярами `InformationItem`. Функция `FunctionDefinition`, содержащаяся в элементе `BusinessKnowledgeModel`, является многократно используемым модулем логики решения, который представлен данным `BusinessKnowledgeModel`.

Элемент `Invocable` также содержит `InformationItem`, который содержит вызываемую ссылку на абстрактные бизнес-знания и позволяет `Decision` вызывать `InformationItem` по имени. Имя этого `InformationItem` ДОЛЖНО совпадать с именем элемента `Invocable`. `Invocable` наследует все атрибуты и ассоциации модели от `DRGElement`. В таблице 15 представлены дополнительные атрибуты и ассоциации модели класса `Invocable`. В таблице 16 представлены дополнительные атрибуты и ассоциации модели класса `BusinessKnowledgeModel`.

Таблица 15. Дополнительные атрибуты и ассоциации модели класса `Invocable`.

Атрибут	Описание
variable: `InformationItem`	Этот атрибут определяет переменную, связанную с функцией, определяемой `FunctionDefinition`, что позволяет логике принятия решения вызывать функцию по имени.

Таблица 16. Дополнительные атрибуты и ассоциации модели класса `BusinessKnowledgeModel`.

Атрибут	Описание
encapsulatedLogic: `FunctionDefinition [0..1]`	Функция, которая инкапсулирует логику, инкапсулированную данным экземпляром `BusinessKnowledgeModel`.
knowledgeRequirement: `KnowledgeRequirement [*]`	В этом атрибуте перечислены экземпляры `KnowledgeRequirement`, которые входят в данный экземпляр `BusinessKnowledgeModel`.
authorityRequirement: `AuthorityRequirement [*]`	В этом атрибуте перечислены экземпляры `AuthorityRequirement`, которые составляют `BusinessKnowledgeModel`.

6.3.10 Метамодель Decision service

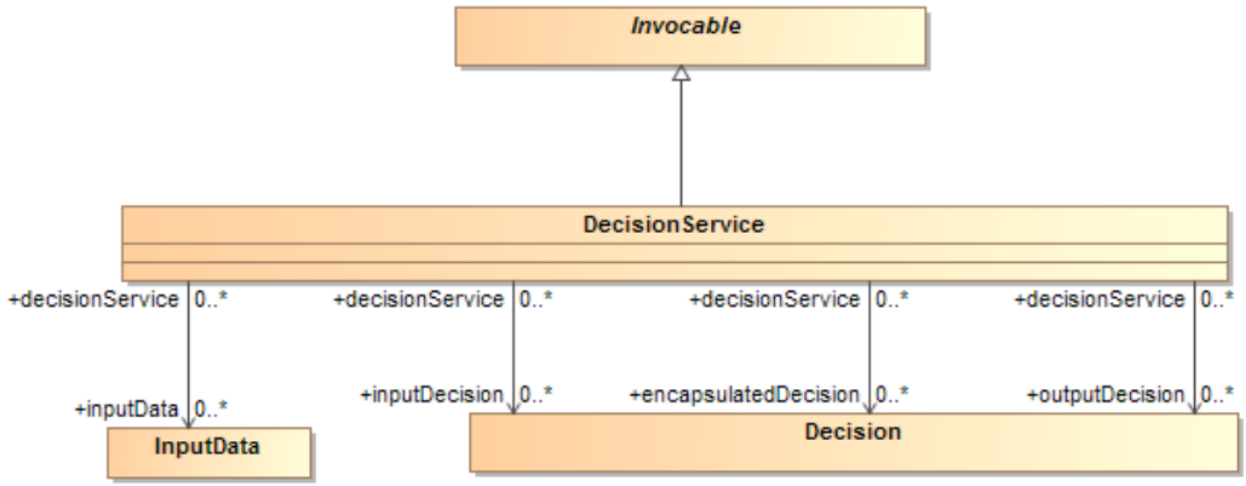


Рисунок 6.15: диаграмма класса DecisionService

Класс `DecisionService` используется для того, чтобы определить именные службы решений в рамках модели принятия решений, содержащейся в экземпляре `Definitions`.

`DecisionService` – это вид элемента `Invocable`, от которого экземпляр `DecisionService` наследует имя `name` и опциональный идентификатор `id`, описание, атрибуты лейбла `label`, являющиеся Строками, а также переменную `variable`, которая является `InformationItem`. Идентификатор `id` элемента `DecisionService` ДОЛЖЕН быть уникальным в рамках экземпляра `Definitions`. Имя `name` переменной `variable` и имя `name` службы `DecisionService` ДОЛЖНЫ быть одинаковыми. Это имя может быть использовано для вызова `DecisionService` из логики другого решения или модели бизнес-знаний.

Элемент `DecisionService` имеет одно или несколько ассоциированных `outputDecisions`, которые являются экземплярами решения `Decision` и которые должны быть на выходе из `DecisionService`, иными словами, это Решения, чьи результаты должна вернуть Служба решений, будучи вызванной.

Элемент `DecisionService` имеет ноль или несколько инкапсулированных `encapsulatedDecisions`, которые являются экземплярами решения `Decision` и которые должны быть инкапсулированы службой `DecisionService`, иными словами, это Решения, чьи результаты должна проверить Служба решений, будучи вызванной.

Элемент `DecisionService` имеет ноль или несколько `inputDecisions`, которые являются экземплярами решения `Decision` и которые `DecisionService` требует в качестве входа, иными словами, это Решения, чьи результаты будут предоставлены Службе решений, когда она будет вызвана.

Элемент `DecisionService` имеет ноль или несколько `inputData`, которые являются экземплярами `InputData` и которые `DecisionService` требует в качестве входа, иными словами, это Решения, чьи результаты будут предоставлены Службе решений, когда она будет вызвана.

Атрибуты `encapsulatedDecisions`, `inputDecisions` и `inputData` являются опциональными. Как минимум один из атрибутов `encapsulatedDecisions` и `inputDecisions` ДОЛЖЕН быть указан.

Подграфиком требований элемента `DecisionService` является ориентированный граф, состоящий из самого элемента `DecisionService`, а также совокупности подграфиков требований каждого элемента `Decision`, на которые ссылаются `encapsulatedDecisions` и `outputDecisions`.

Синтаксис экземпляра `DecisionService` считается корректным только в том случае, если его подграфик требований является ациклическим, то есть элемент `DecisionService` НЕ ДОЛЖЕН требовать самого себя прямо или косвенно.

`DecisionService` наследует все атрибуты и модельные связи от `Invocable`. В таблице 17 представлены дополнительные атрибуты и модельные связи элемента `DecisionService`.

Таблица 17: Атрибуты и модельные связи элемента **`DecisionService`**

Атрибут	**Описание**
outputDecisions: `Decision [1..*]`	В этом атрибуте перечислены экземпляры решения `Decision`, которые должны быть на выходе службы `DecisionService`.
encapsulatedDecisions: `Decision [0..*]`	При наличии, этот атрибут содержит список экземпляров решения `Decision`, которые должны быть инкапсулированы в службе `DecisionService`
inputDecisions: `Decision [0..*]`	При наличии, этот атрибут содержит список экземпляров решения `Decision`, которые должны быть на входе службы `DecisionService`
inputData: `InputData [0..*]`	При наличии, этот атрибут содержит список экземпляров `InputData`, которые должны быть на входе службы `DecisionService`

6.3.11 Метамодель Input Metadata

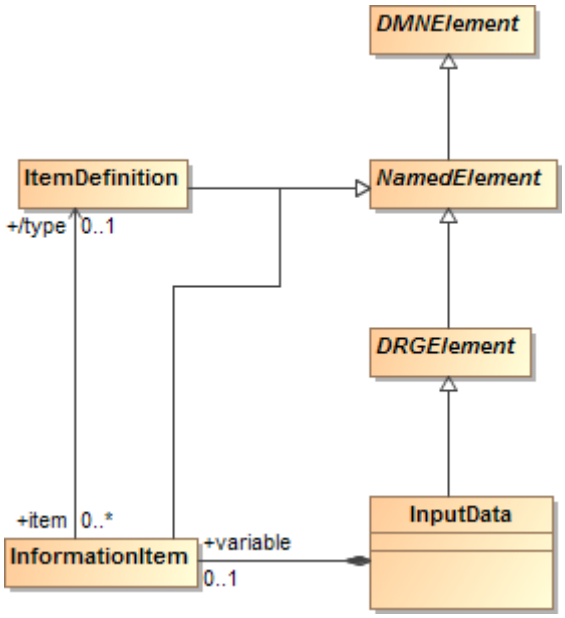


Рисунок 6.16: диаграмма класса InputData

DMN 1.3 использует класс `InputData` для моделирования входов решения, значения которых определены вне модели принятия решения.

`InputData` – это конкретная специализация `DRGElement`, наследующая имя `name` и дополнительные атрибуты `id`, `description` и `label` от `NamedElement`. Имя `InputData` должно быть отличным от имени любого другого решения, входных данных, модели бизнес-знаний, службы решений или импорта в модели принятия решений.

Экземпляр `InputData` определяет `InformationItem`, который хранит его значение. В этом случае `InformationItem` может включать в себя `typeRef` (`ItemDefinition`, базовый тип в указанном `expressionLanguage`, либо импортированный тип, который определяет тип данных, представленный `InputData`).

На ДТР экземпляр `InputData` представлен элементом диаграммы **Входные данные**. Элемент `InputData` не имеет **подграфика требований** и всегда имеет **правильный синтаксис**.

`InputData` наследует все атрибуты и ассоциации модели от `DRGElement`. В таблице 18 представлены дополнительные атрибуты и ассоциации моделей класса `InputData`.

Таблица 18: Атрибуты и ассоциации модели InputData

Атрибут	Описание
variable: InformationItem	Экземпляр `InformationItem`, который сохраняет результат `InputData`.
typeRef: String[0..1]	Указатель на спецификацию типа данных возможных значений `InputData`. Это может быть `ItemDefinition`, либо базовый тип в заданном `expressionLanguage`, либо импортированный тип.

6.3.12 Метамодель Knowledge Source

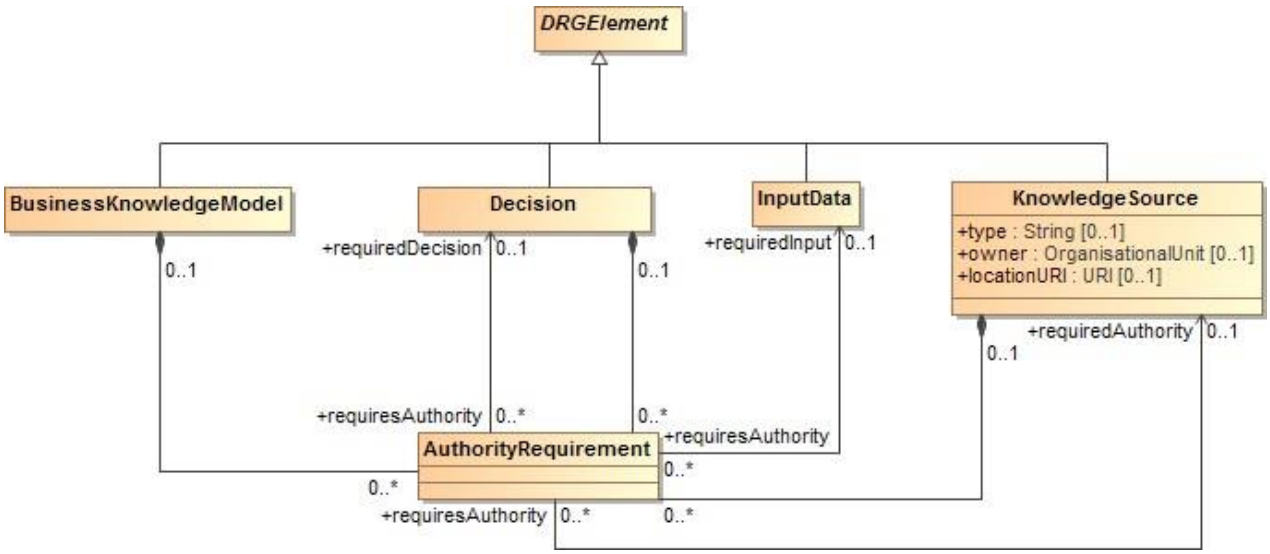


Рисунок 6.17. Диаграмма класса KnowledgeSource

Класс `KnowledgeSource` используется для моделирования достоверных источников знаний в модели принятия решений.

На ДТР экземпляр `KnowledgeSource` представлен элементом диаграммы **Источник знаний**.

`KnowledgeSource` – это конкретная специализация элемента `DRGElement` и, следовательно, элемента `NamedElement`, от которого `KnowledgeSource` наследует имя `name` и дополнительные атрибуты `id`, `description` и `label`. Кроме того, у `KnowledgeSource` есть `locationURI`, который является уникальным идентификатором ресурсов (URI). У `KnowledgeSource` есть атрибуты `type` (строка) и `owner` (экземпляр `OrganisationalUnit`). Атрибут `type` предназначен для определения достоверного источника знаний, например, регламента компании, правил, аналитического заключения.

Элемент `KnowledgeSource` может включать или не включать `authorityRequirement`, которые являются экземплярами `AuthorityRequirement`.

`KnowledgeSource` наследует все атрибуты и ассоциации модели от `DRGElement`. В таблице 19 представлены атрибуты и ассоциации модели класса `KnowledgeSource`.

Таблица 19: Атрибуты и ассоциации модели `KnowledgeSource`

Атрибут	Описание
locationURI: anyURI [0..1]	URI, в котором находится данный `KnowledgeSource`. `locationURI` ДОЛЖЕН указываться в формате URI.
type: string [0..1]	Тип `KnowledgeSource`
owner: OrganisationalUnit [0..1]	Владелец `KnowledgeSource`
authorityRequirement: AuthorityRequirement [*]	Данный атрибут определяет экземпляры `AuthorityRequirement`, которые дополняют `KnowledgeSource`.

6.3.13 Метамодель Information Requirement

Класс `InformationRequirement` используется для моделирования **Требования к информации**, представленного простой стрелкой на ДТР. `InformationRequirement` является специализацией `DMNElement`, от которого он наследует дополнительные атрибуты `id`, `description` и `lable`.

Элемент `InformationRequirement` является компонентом элемента `Decision` и связывает `Decision` с элементом `requiredDecision`, который является экземпляром `Decision` или `requireInput`, который в свою очередь является экземпляром `InputData`.

Элемент `InformationRequirement` ссылается на экземпляр `Decision` или `InputData`, который определяет переменную `variable`. Эта переменная `variable`, являющаяся экземпляром `InformationItem`, представляет элемент `InformationRequirement` на уровне логики решения.

Обратите внимание, что элемент `InformationRequirement` должен ссылаться на экземпляр `Decision` или `InputData`, который он связывает с элементом `Decision`, но не содержать его: экземпляры `Decision` или `InputData` могут содержаться только в элементах `Definitions`.

Синтаксис экземпляра `InformationRequirement` считается **корректным**, только если верно следующее:

- `InformationRequirement` ссылается на элемент `requiredDecision` или `requiredInput`, но не на оба эти элемента одновременно;
- адресуемые элементы `requiredDecision` или `requiredInput` имеют правильный синтаксис;
- элемент `Decision`, содержащий экземпляр `InformationRequirement`, не находится в подграфике требований адресуемого элемента `requiredDecision`, если `InformationRequirement` ссылается на него;
- адресуемые элементы `requiredDecision` или `requiredInput` определены в той же модели принятия решений или в импортируемой модели принятия решений.

В таблице 20 представлены атрибуты и ассоциации модели элемента `InformationRequirement`.

Таблица 20. Атрибуты и ассоциации модели `InformationRequirement`

Атрибут	Описание
requiredDecision: Decision [0..1]	Экземпляр `Decision`, который данный элемент `InformationRequirement` связывает с содержащим его элементом `Decision`.
requiredInput: InputData [0..1]	Экземпляр `InputData`, который данный элемент `InformationRequirement` связывает с содержащим его элементом `Decision`.

6.3.14 Метамодель Knowledge Requirement

Класс KnowledgeRequirement используется для моделирования **Требования к знаниям**, представленного пунктирной стрелкой на ДТР. KnowledgeRequirement является специализацией DMNElement, от которого он наследует дополнительные атрибуты id, description и lable.

Элемент KnowledgeRequirement является компонентом элемента Decision или BusinessKnowledgeModel и связывает Decision или BusinessKnowledgeModelс элементом requiredKnowledge, который является экземпляром BusinessKnowledgeModel.

Обратите внимание, что элемент KnowledgeRequirement должен ссылаться на экземпляр Invocable, который он связывает с элементом Decision или BusinessKnowledgeModel, но не содержать его: экземпляры BusinessKnowledgeModel могут содержаться только в элементах Definitions.

Синтаксис экземпляра KnowledgeRequirement считается **корректным** только если верно следующее:

- KnowledgeRequirement ссылается на requiredKnowledge;
- адресуемый элемент requiredKnowledge имеет правильный синтаксис;
- если элемент KnowledgeRequirement содержится в экземпляре BusinessKnowledgeModel, то такой элемент BusinessKnowledgeModel не находится в подграфике требований адресуемого элемента requiredKnowledge;
- адресуемый элемент requiredKnowledge определен в той же модели принятия решений или в импортируемой модели принятия решений

В таблице 21 представлены атрибуты и ассоциации модели элемента KnowledgeRequirement.

Таблица 21. Атрибуты и ассоциации модели KnowledgeRequirement

Атрибут	Описание
requiredKnowledge: Invocable	Экземпляр Invocable, который данный элемент KnowledgeRequirement связывает с содержащим его элементом Decision или KnowledgeRequirement.

6.3.15 Метамодель Authority Requirement

Класс AuthorityRequirement используется для моделирования **Требования к полномочиям**, изображаемого на ДТР в виде стрелки, выполненной пунктирной линией с залитым кругом на конце. AuthorityRequirement является специализацией DMNElement, от которого он наследует дополнительные атрибуты id, description и lable.

Элемент AuthorityRequirement может быть компонентом элементов Decision, BusinessKnowledgeModel или KnowledgeSource и связывать Decision, BusinessKnowledgeModel или KnowledgeSource:

- с элементом requiredAuthority, который является экземпляром KnowledgeSource;
- с элементом requiredDecision, который является экземпляром Decision;
- с элементом requiredInput, который является экземпляром InputData.

Обратите внимание, что элемент `AuthorityRequirement` должен ссылаться на экземпляры `KnowledgeSource`, `Decision` или `InputData`, которые он связывает с соответствующим элементом, но не содержать их: экземпляры `KnowledgeSource`, `Decision` или `InputData` могут содержаться только в элементах `Definitions`.

В таблице 22 представлены атрибуты и ассоциации модели элемента `AuthorityRequirement`.

Таблица 22. Атрибуты и ассоциации модели `AuthorityRequirement`

Атрибут	Описание
requiredAuthority: `KnowledgeSource` [0..1]	Экземпляр `KnowledgeSource`, который данный элемент `AuthorityRequirement` связывает с содержащим его элементом `KnowledgeSource`, `Decision` или `BusinessKnowledgeModel`.
requiredDecision: `Decision` [0..1]	Экземпляр `Decision`, который данный элемент `AuthorityRequirement` связывает с содержащим его элементом `KnowledgeSource`.
requiredInput: `InputData` [0..1]	Экземпляр `InputData`, который данный элемент `AuthorityRequirement` связывает с содержащим его элементом `KnowledgeSource`.

6.3.16 Расширяемость

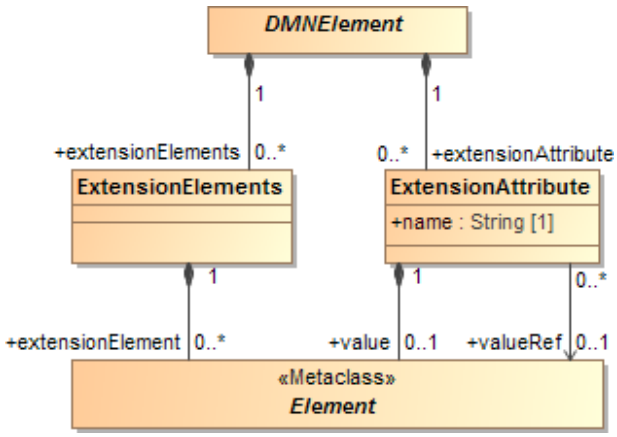


Рисунок 6.18. Диаграмма класса Расширяемость

Метамодель **DMN** предусматривает возможность расширения. Это позволяет соразработчикам **DMN** расширять указанную метамодель, но оставаться в рамках совместимости с **DMN**. Набор расширений позволяет соразработчикам **DMN** присоединять дополнительные атрибуты и элементы к стандартным существующим элементам **DMN**. Такой подход позволяет создавать более взаимозаменяемые модели, поскольку стандартные элементы остаются неизменными и понятными для других соразработчиков **DMN**. Во время обмена только дополнительные атрибуты и элементы МОГУТ быть потеряны.

Расширение **DMN** может быть выполнено с использованием двух разных элементов:

1. ExtensionElements.

2. ExtensionAttribute.

`ExtensionElements` – это контейнер для добавления произвольных элементов из других метамоделей к любому элементу **DMN**.
`ExtensionAttribute` позволяет указать имя добавленных элементов. Таким образом соразработчики **DMN** могут интегрировать любую метамодель в метамодель **DMN** и повторно использовать уже существующие элементы модели.

6.3.16.1 ExtensionElements

Элемент `ExtensionElements` – это контейнер, позволяющий добавить элементы других метамоделей к любому `DMNElement`. В таблице 23 представлены атрибуты и ассоциации модели `ExtensionElements`.

Таблица 23: Атрибуты и ассоциации модели ExtensionElements

Атрибут	**Описание**
extensionElement: Element [0..*]	Содержащийся Элемент. Эта связь неприменима, если используется обмен XML-схемами, поскольку механизм XSD для поддержки «любых» элементов из других пространств имен уже удовлетворяет этому требованию.

6.3.16.2 ExtensionAttribute

Элемент `ExtensionAttribute` содержит Элемент или ссылку на Элемент из другой метамодели. У элемента `ExtensionAttribute` также есть имя, которое определяет роль или цель связанного элемента. Этот тип не применяется, когда используется обмен XML-схемами, поскольку механизм XSD для поддержки «anyAttribute» из других пространств имён уже удовлетворяет этому требованию. В таблице 24 представлены ассоциации модели для элемента `ExtensionAttribute`.

Таблица 24: Атрибуты и ассоциации модели `ExtensionAttribute`

Атрибут	**Описание**
name: string	Имя атрибута расширения.
value: Element [0..1]	Содержащийся Элемент. Данный атрибут НЕ ДОЛЖЕН использоваться с `valueRef`.
valueRef: Element [0..1]	Ссылка на связанный Элемент. Данный атрибут НЕ ДОЛЖЕН использоваться с `value`.

6.4 Примеры

Примеры ДТР приведены в разделе 11.1.3.

7 Согласование логики принятия решений с требованиями принятия решений

7.1 Введение

Раздел 6 описывает, как моделируется структура принятия решения. Для этого используется уровень требований к решениям, или, иными словами, график требований к решениям, представленный в виде одной или нескольких диаграмм требований к решениям. Однако то, каким образом результат решения выводится из набора входов, пошагово моделируется на уровне логики решения. В этом разделе представлены принципы, которые позволяют связать логику принятия решения с элементами ГТР. Подробное описание логики принятия решения (таблицы принятия решений и выражения FEEL) приводится в разделах 8, 9 и 10.

Уровень логики решения, присущей модели решения DMN, состоит из одного или нескольких выражений значений. Элементы логики решения, моделируемые как выражения значений, включают такие табличные выражения, как таблицы решений, вызовы, и литеральные (текстовые) выражения, например, *возраст> 30:*

- **Литеральное выражение** представляет логику принятия решения в виде текста, описывающего, каким образом выходное значение выводится из входных значений. Язык выражения может (но не обязан) быть формальным или исполняемым. Примеры литеральных выражений включают в себя простое описание логики решения на естественном языке, логическое предложение первого порядка, компьютерную программу Java и документ PMML. В разделе 10 описывается исполняемый язык выражений, называемый **FEEL**. В разделе 9 описывается подмножество FEEL (S-FEEL), которое является языком по умолчанию для литеральных выражений в таблицах решений DMN (раздел 8);

- **Таблица решений** – это табличное представление логики решения на основе дискретизации возможных значений входов решения. Логика принятия решения организованна в виде правил, которые связывают дискретизированные входные значения с дискретными выходными значениями (см. Раздел 8);

- **Вызов –** это табличное представление того, каким образом логика принятия решения, обозначенная моделью бизнес-знания или службой решений, вызывается решением или другой моделью бизнес-знаний. Вызов также может быть представлен как литеральное выражение, однако табличное представление, как правило, более доступно для понимания.

Далее в тексте табличные представления логики решения будут называться *табличными выражениями* (boxed expressions).

Все три уровня соответствия **DMN** включают в себя все приведенные выше выражения. На уровне соответствия 1 литеральные выражения не интерпретируются и, следовательно, не определены. На уровне соответствия 2 литеральные выражения ограничены S-FEEL. В разделе 10 указаны дополнительные табличные выражения, доступные на третьем уровне соответствия DMN.

Логика принятия решений добавляется к модели решения путем включения компонента выражения значения в некоторые элементы моделирования на ГТР:

- на уровне логики решение – это та часть логики, которая определяет, каким образом на основе входных данных дается ответ на заданный вопрос. Как следствие, каждый элемент **решения** в модели принятия решения может включать выражение значения, которое описывает, каким образом результат решения выводится из требуемых входных значений. При этом может вызываться модель бизнес-знаний;

- на уровне логики, модель бизнес-знаний является той частью логики решения, которая определяется как функция, что позволяет повторно использовать эту часть логики в различных

решениях. Как следствие, каждый элемент **модели бизнес-знаний** может включать выражение значения, которое является телом этой функции.

Другой компонент уровня логики решения это **переменная.** Переменные предназначены для хранения значений Decisions и InputData для использования в выражениях значений. InformationRequirements определяет переменные в области видимости ссылаясь на Decisions и InputData, таким образом выражения значений могут ссылаться на эти переменные. Переменные связывают требования к информации на ГТР с выражениями значений на уровне логики решения:

- с точки зрения логики принятия решения, требование к информации является требованием, необходимым для того, чтобы присвоить значение, полученное извне, свободной переменной в логике принятия решения, что позволит оценить решение. Как следствие, каждое требование к информации в модели принятия решения указывает на Decisions или InputData, которые, в свою очередь, определяют переменную, представляющую связанные входные данные в выражении решения;

- переменные, которые используются в теле функции, определенной элементом модели бизнес-знаний на ГТР, должны быть привязаны к источникам информации в каждом значении, требующем решений. Как следствие, каждая **модель бизнес-знаний** включает в себя ноль или более переменных, которые являются параметрами функции.

Третьим ключевым элементом уровня логики решения являются определения элемента (**item definitions**). Они описывают типы и структуры элементов данных в модели принятия решений. **Элементы входных данных** на ГТР, а также **переменные** и **выражения значений** на уровне логики решения, могут ссылаться на связанное определение элемента, которое описывает тип и структуру данных, ожидаемых на входе, присвоенных переменной или являющихся результатом оценки выражения.

Обратите внимание, что **источники знаний** не представлены на уровне логики решения. Источники знаний являются частью документации логики решения, но не самой логикой решения.

Зависимости между решениями, необходимыми источниками информации и моделями бизнес-знаний представлены на ГТР в виде «требования к информации» и «требования к знаниям». Они определяют каким образом выражения значений соотносятся с упомянутыми элементами.

Как было сказано выше, каждое требование к информации на уровне ГТР связано с переменной на уровне логики решения. Каждая переменная, на которую ссылается выражение решения, должна быть ассоциирована с требуемым решением, требуемым входными данными или требуемым знанием. Кроме того, каждая переменная, ассоциированная с требуемыми решениями, входными данными или знанием, должна быть также указана в выражении решения:

- если для принятия решения требуется другое решение, выражение значения требуемого решения присваивает значение переменной, с тем чтобы оценить требующее решение. Это общий механизм в **DMN** для составления решений на уровне логики решения;

- если для принятия решения требуются входные данные, значение переменной присваивается значению источника данных, прикрепленного к входным данным во время выполнения. Это общий механизм в **DMN** для создания требований к данным для принятия решения.

Входные переменные логики принятия решения не должны использоваться вне выражения значений или вне выражений значения компонентов. Элемент решения определяет лексическую область входных переменных для логики принятия решения. Чтобы избежать конфликтов имён и неоднозначности, имя переменной должно быть уникальным в пределах своей области. При сопоставлении элементов ГТР с FEEL имя переменной совпадает с именем (возможно, квалифицированным) связанных с ними входных данных или решением, что гарантирует его уникальность.
При сопоставлении элементов ГТР с FEEL все решения и входные данные на ГТР определяют *контекст*, который является литеральным выражением, представляющим логику и область решения. Элементами

«требований к информации» в решении являются *записи контекста* в связанном контексте, где *ключ* – это имя переменной, которую определяет требование к информации, а *выражение* – это *контекст*, связанный с требуемым элементом решения или входными данными, на которые ссылается требование к информации. Выражение значения, задающее логику решения, является *выражением* в *записи контекста*, которое определяет результат *контекста*.

Точно так же элемент «модель бизнес-знаний» определяет лексическую область своих параметров, то есть входных переменных тела.

В FEEL конструкция «литеральное выражение – область», представляющая собой логику, связанную с моделью бизнес-знаний, являются *определением функции* (см. 10.3.2.11 «Семантика функции»), где *формальные параметры* – это имена параметров в элементе «модель бизнес-знаний», а *выражение* представляет собой выражение значения, которое является телом элемента «модель бизнес-знаний».

Если элемент «модель бизнес-знаний» требует одну или несколько моделей бизнес-знаний, такой элемент должен иметь явное выражение значения, которое описывает, как вызываются требуемые модели бизнес-знаний и как объединяются или обрабатываются их результаты.

На уровне логики решения решение вызывает требуемую модель бизнес-знаний путем оценки выражения значения модели бизнес-знаний с параметрами, привязанными к собственному входному значению. Как это может быть достигнуто, зависит от того, каким образом логика решения распределяется между решением и моделями бизнес-знаний:

- если элемент решения требует более одного элемента бизнес-знаний, его выражение значения должно быть литеральным выражением, которое указывает, каким образом вызываются элементы модели бизнес-знаний и как их результаты объединяются в результат решения;

- если решение не требует каких-либо моделей бизнес-знаний, его выражение значения должно быть литеральным выражением или таблицей решения, которая определяет всю логику принятия решения для вычисления выходов решения на основе его входов;

- подобным образом, если элемент решения требует только один элемент «модель бизнес-знаний», но логика решения подробно описывает логику требуемой модели бизнес-знаний, у элемента решения должно быть литеральное выражение, которое определяет, как выражение значения модели бизнес-знаний вызывается и каким образом его результат обрабатывается для получения результата решения;

- во всех других случаях (то есть, когда решение требует только одну модель бизнес-знаний и не описывает логику) выражение значения элемента решения может быть выражением значения вызова типа. В выражении значения вызова типа необходимо указать только привязки параметров модели бизнес-знаний к входным данным решений: результатом решения является результат, возвращаемый выражением значения модели бизнес-знаний для значений, переданных его параметрам.

Привязка параметра модели бизнес-знаний является выражением значения, которое указывает, как значение, переданное этому параметру, выводится из значений входных переменных вызывающего решения.

7.2 Нотация

7.2.1 Выражения

В данном разделе мы определяем нотацию для логики решения, представленной в виде **табличных выражений**. Нотация позволяет разбить модель логики решения на небольшие составляющие, которые могут быть связаны с артефактами ГТР. ДТР вкупе с табличными выражениями образуют исчерпывающий, преимущественно графический язык, который полностью определяет модели принятия решений.

Помимо общего определения табличных выражений, в этом разделе приводится описание двух видов табличных выражений:

- **Табличные литеральные выражения;**
- **Табличные вызовы.**

Табличные выражения для таблицы принятия решений описывается в разделе 8. В разделе 10 приводятся другие типы табличных выражений для FEEL.

Табличные выражения определяются рекурсивно, т.е. табличные выражения могут содержать другие табличные выражения. Верхне-уровневые табличные выражения соответствуют логике решения одного артефакта ГТР. У такого табличного выражения ДОЛЖНА быть ячейка имени, которая содержит имя артефакта ГТР. Ячейка имени может располагаться сверху, как показано на рисунке 7.1:

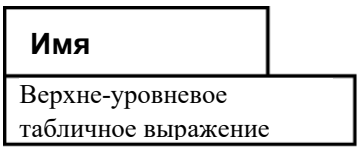


Рисунок 7.1: Табличные выражения

Кроме того, ячейка имени и ячейка выражения могут быть отделены друг от друга пробелом и соединены с левой стороны с помощью линии, как показано на рисунке 7.2:

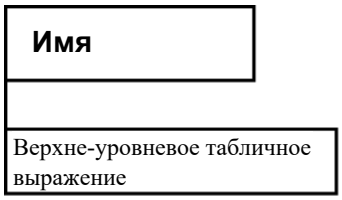


Рисунок 7.2. Табличное выражение с раздельными полями имени и выражения

Имя – единственная визуальная ссылка, указывающая на связь между элементами ДТР и табличными выражениями. Ожидается, что графические инструменты будут поддерживать соответствующие графические ссылки, например, при нажатии на фигуру решения откроется Таблица принятия решений. Разработчики ПО для моделирования сами определяют, каким образом табличное выражение визуально соединяется с элементом ДТР.

7.2.2 Табличное литеральное выражение

В табличном выражении литеральное выражение представлено текстом. Однако для улучшения удобочитаемости табличных литеральных выражений используются два типа условных обозначений: типографские строковые литералы и типографские литералы даты и времени.

7.2.2.1 Типографские строковые литералы

Строковый литерал, такой как «ОТКАЗАТЬ», может быть представлен альтернативно как курсивный литерал *ОТКАЗАТЬ*. Например, рисунок 7.3 эквивалентен рис. 7.4:

Таблица коэффициента кредитного риска

U	Категория риска	Коэффициент кредитного риска
1	*ВЫСОКИЙ, ОТКАЗАТЬ*	0.6
2	*СРЕДНИЙ*	0.7
3	*НИЗКИЙ, ОЧЕНЬ НИЗКИЙ*	0.8

Рисунок 7.3: Таблица решения с курсивным литералом

Таблица коэффициента кредитного риска

U	Категория риска	Коэффициент кредитного риска
1	“ВЫСОКИЙ”, “ОТКАЗАТЬ”	0.6
2	“СРЕДНИЙ”	0.7
3	“НИЗКИЙ”, “ОЧЕНЬ НИЗКИЙ”	0.8

Рисунок 7.4: Таблица решения со строчным литералом

Чтобы избежать двусмысленности (например, *ВЫСОКИЙ, ОТКАЗАТЬ* может означать "ВЫСОКИЙ", "ОТКАЗАТЬ" или "ВЫСОКИЙ, ОТКАЗАТЬ") НЕ СЛЕДУЕТ использовать запятые в типографических строковых литералах (символ «,»). Типографские строковые литералы FEEL ДОЛЖНЫ соответствовать 22 правилу грамматики (имя).

7.2.2.2 Типографские литералы даты и времени

Выражение даты, времени, даты и времени или продолжительности, например, "2013-08-09", в качестве альтернативы может быть представлено жирным курсивным литералом ***2013-08-09***. Литерал ДОЛЖЕН подчиняться синтаксису, указанному в разделе 10.3.2.3.4, 10.3.2.3.5 и 10.3.2.3.7.

7.2.3 Табличные вызовы

Вызов представляет собой контейнер для привязок параметров, содержащих контекст для оценки тела модели бизнес-знаний.

Представление вызова – это имя модели бизнес-знаний с явными перечислениями параметров.

В виде табличного выражения вызов представляет собой ячейку, содержащую имя модели бизнес-знаний, которую нужно вызвать, и ячейки для списка привязок, где каждая привязка представлена двумя табличными выражениями в ряду: ячейка слева содержит имя параметра, а ячейка справа содержит выражение привязки, то есть выражение, значение которого присваивается параметру с целью оценки модели бизнес-знаний (см. рис. 7.5).

Имя	
Вызванная модель бизнес-знания	
параметр 1	Выражение привязки 1
...	
параметр 2	Выражение привязки 2
параметр *n*	Выражение привязки *n*

Рисунок 7.5: Табличный вызов

Вызванная модель бизнес-знаний представлена названием модели бизнес-знаний. Любые другие визуальные привязки предусматриваются самостоятельно разработчиками ПО для моделирования.

7.3 Метамодель

Важной характеристикой решений и моделей бизнес-знаний является возможность содержать выражение, описывающее логику принятия смоделированного решения или части этой логики.

Класс `Expression` является абстрактным суперклассом для всех выражений, которые используются для описания логики принятия решений или ее части в моделях **DMN**, и которые возвращают одиночное значение при интерпретации (см. 7.3.1). В данном контексте «одиночное значение» может включать структурированные данные, такие как таблица принятия решений с несколькими выходными разделами.

DMN определяет три конкретных типа выражения: `LiteralExpression`, `DecisionTable` (см. Раздел 8) и `Invocation`.

Выражение может ссылаться на переменные, таким образом значение выражения при интерпретации зависит от значений, присвоенных адресуемым переменным. Класс `InformationItem` используется для моделирования переменных в выражениях.

Значение выражения, как и значение, присвоенное переменной, может иметь структуру и допустимый диапазон. Класс `ItemDefinition` используется для моделирования структур данных и диапазонов.

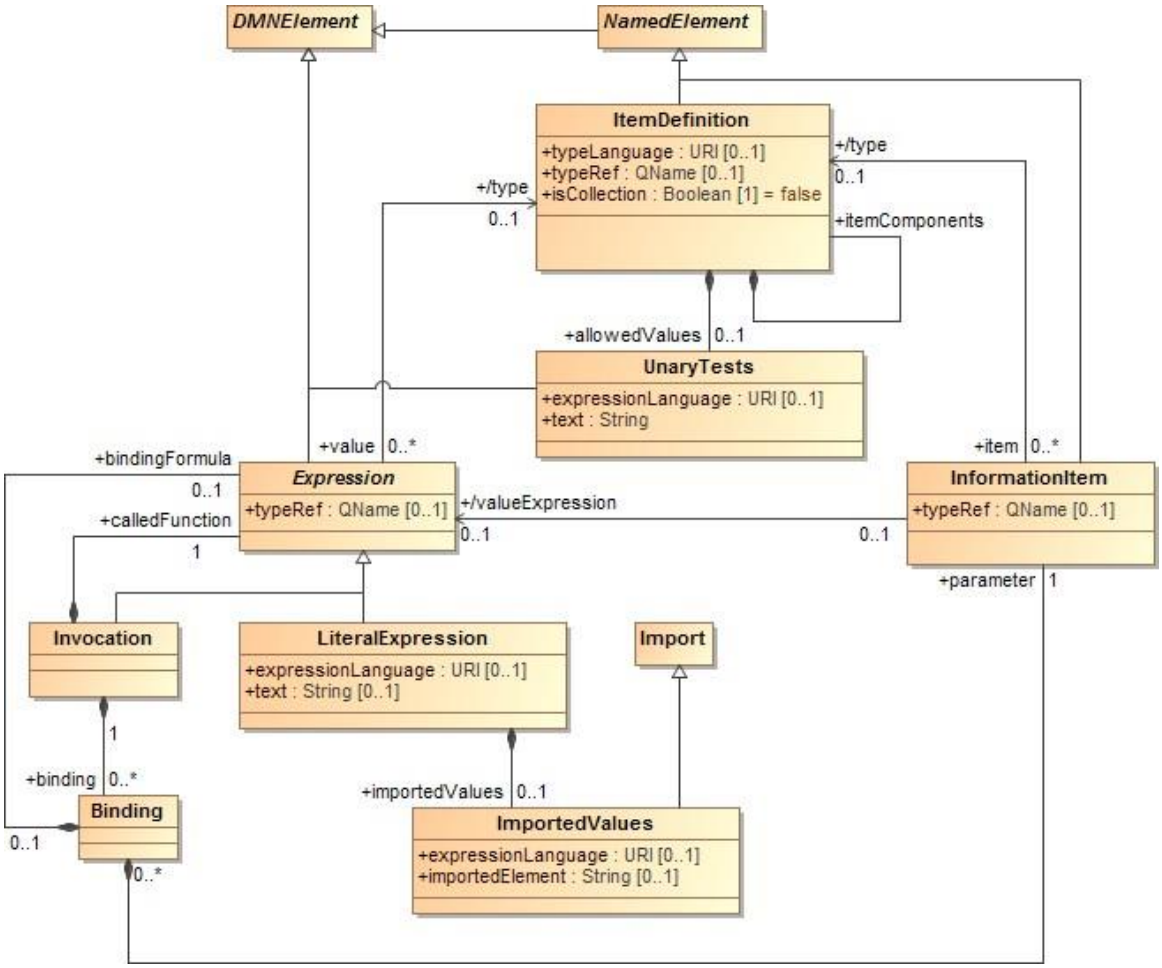


Рисунок 7.6: Диаграмма класса `Expression`

7.3.1 Метамодель Expression

Важной характеристикой решений и моделей бизнес-знаний является возможность содержать выражение, описывающее логику принятия смоделированного решения или части этой логики.

`Expression` представляет собой абстрактную специализацию элемента `DMNElement`, от которого оно наследует дополнительные атрибуты `id, description и label`.

Экземпляр `Expression` является компонентом элементов `Decision`, `BusinessKnowledgeModel` или `ItemDefinition`. Он также прямо или косвенно может быть компонентом другого экземпляра `Expression`.

`Expression` неявно ссылается на ноль или более переменных, используя их имена в тексте выражения. Эти переменные, являющиеся экземплярами `InformationItem`, лексически ограничены в зависимости от типа `Expression`. Если `Expression` является логикой `Decision`, в сферу охвата входят требования `Decision`. Если `Expression` – это тело инкапсулированной логики `BusinessKnowledgeModel`, то в сферу охвата входят параметры `FunctionDefinition` и требования `BusinessKnowledgeModel`. Если `Expression` представляет собой значение `ContextEntry`, сфера охвата включает предыдущие записи в контексте `Context`. Экземпляр `Expression` ссылается на необязательный `typeRef`, который указывает либо на базовый тип в языке по умолчанию `typeLanguage`, либо на настраиваемый тип, заданный `ItemDefinition`, либо на импортированный тип. Адресуемый тип задает диапазон возможных значений

Expression. Если экземпляр Expression, определяющий вывод элемента Decision, включает typeRef, то адресуемый тип ДОЛЖЕН совпадать с типом содержащего его элемента Decision.

Экземпляр Expression может интерпретироваться для получения единственного значения из значений, присвоенных его переменным. Конкретный вид Expression определяет, каким образом значение элемента Expression выводится из значений, присвоенных его переменным.

Expression наследуется от атрибутов и ассоциаций моделей DMNElement.

7.3.2 Метамодель UnaryTests

Класс UnaryTests используется для того, чтобы смоделировать логический тест boolean test, где проверяемый аргумент либо неявно определен, либо обозначен вопросительным знаком ?. Его значение при этом приведено в виде текста на одном из указанных языков выражений.

UnaryTests является конкретным подклассом Expression.

Экземпляр UnaryTests наследует опциональный typeRef от Expression, который НЕ ДОЛЖЕН быть использован. Экземпляр UnaryTests так же имеет опциональный текст text, который является строкой, и опциональный язык выражений expressionLanguage, который является строкой, определяющей язык выражений текста. Если не указан ни один из языков expressionLanguage, то языком выражений текста будет язык expressionLanguage, связанный с содержащим его экземпляром Definitions. Язык выражений должен быть указан в формате URI. Языком выражений по умолчанию является FEEL. Если языком выражений является FEEL, текст должен соответствовать правилу грамматики 15 раздела 10.3.1.2. В таблице 25 приведены дополнительные атрибуты и ассоциации элемента UnaryTests.

Таблица 25: Атрибуты и ассоциации модели UnaryTests

Атрибут	**Описание**
text: String [0..1]	Текст данного UnaryTests. Это ДОЛЖНО быть валидное выражение на языке expressionLanguage
expressionLanguage: anyURI[0..1]	Данный атрибут идентифицирует язык, используемый в этом UnaryTests. Это значение переопределяет выражение, указанное для содержащего экземпляра DecisionRequirementDiagram. Язык ДОЛЖЕН быть указан в формате URI.

7.3.3 Метамодель ItemDefinition

Входные данные и результаты решений, моделей бизнес-знаний и служб решений, а также входных и выходных данных (всех элементов `DRGElements`) – это элементы данных, значение которых на уровне логики решения присваивается переменным или представлено выражениями `Expressions`.

Важной характеристикой элементов данных в моделях решений является их структура. В **DMN** для структур данных не предусмотрен один обязательный формат, однако подмножество FEEL определяется как формат по умолчанию.

Класс `ItemDefinition` используется для моделирования структуры и диапазона значений входных данных и результатов решений.

В качестве конкретной специализации `NamedElement` экземпляр `ItemDefinition` имеет имя `name`, необязательный идентификатор `id` и описание `description`. Имя `name` элемента `ItemDefinition` ДОЛЖНО быть отличным от других `ItemDefinitions` и Импортов в рамках одной и той же модели.

Язык типа по умолчанию для всех элементов может быть указан в элементе `Definitions` при помощи атрибута `typeLanguage`. Например, значение `typeLanguage` http://www.w3.org/2001/XMLSchema указывает, что структуры данных, используемые элементами в пределах этих `Definitions`, представлены в виде типов XML-схем. Если не указано иное, языком по умолчанию является FEEL.

Следует отметить, что типы данных, которые входят в атрибут `typeLanguage`, связанный с экземпляром `Definitions`, не должны переопределяться элементами `ItemDefinition`, содержащимися в этом `Definitions`: такие типы данных считаются импортированными и на них можно ссылаться в элементах **DMN** в рамках `Definitions`.

Язык типа можно переопределить локально, используя атрибут `typeLanguage` в элементе `ItemDefinition`.

Также обратите внимание, что типы и структуры данных, которые определены на верхнем уровне модели данных, импортируемой с помощью элемента `Import`, связанного с экземпляром `Definitions`, не должны переопределяться элементами `ItemDefinition`, содержащимися в данном элементе `Definitions`: такие типы и структуры данных считаются импортированными и на них можно ссылаться в элементах **DMN** в рамках `Definitions`.

У элемента `ItemDefinition` МОЖЕТ быть атрибут `typeRef`, являющийся строкой, которая, в качестве полного имени, ссылается либо на `ItemDefinition` в текущем экземпляре `Definitions`, либо на встроенный тип в указанном `typeLanguage`, или тип, определенный в импортированном DMN, XSD или другом документе. В последнем случае внешний документ ДОЛЖЕН быть импортирован в элемент `Definitions`, который содержит экземпляр `ItemDefinition`, при помощи элемента `Import`, где будут определены как значение пространства имен, так и имя, используемое в качестве квалификатора. Например, в случае структур данных, предоставленных XML-схемой, `Import` будет использоваться для указания расположения файла этой схемы, а атрибут `typeRef` будет ссылаться на определение типа или элемента в импортированной схеме. Если языком типа является FEEL, встроенными типами являются встроенные типы данных FEEL: *number, string, boolean, days and time duration, years and months duration, date, time, date and time* и *Any (число, строка, булевское значение, продолжительность в днях и продолжительность времени, продолжительность в годах и месяцах, дата, время, дата и время* и *любое)*. Атрибут `typeRef`, ссылающийся на встроенный тип, не ДОЛЖЕН содержать префикс.

Элемент `ItemDefinition` может ограничивать допустимые для `typeRef` значения, используя атрибут `allowedValues`. `allowedValues` – это экземпляр `unaryTests`, который указывает допустимые значения или диапазоны допустимых значений в домене `typeRef`. Тип допустимых значений ДОЛЖЕН соответствовать содержащему элементу `ItemDefinition`. Если элемент `ItemDefinition` содержит одно или несколько допустимых значений, `allowedValues` указывает полный диапазон значений, которые представляет это `ItemDefinition`. Если элемент `ItemDefinition` не содержит

allowedValues, его диапазон допустимых значений соответствует полному диапазону адресуемого typeRef. В тех случаях, когда значения, представляемые элементом ItemDefinition, являются совокупностями значений в разрешенном диапазоне, кратность может быть спроецирована в атрибут isCollection. По умолчанию данный атрибут имеет значение – *false*.

Альтернативным способом определения экземпляра является сложение элементов ItemDefinition. Экземпляр ItemDefinition может содержать ноль или более itemComponent, которые сами являются ItemDefinitions. Каждый элемент itemComponent, в свою очередь, может быть определен как typeRef, так и allowedValues или вложенным элементом itemComponent. Таким образом можно определить сложные типы в **DMN**. Имя элемента itemComponent (вложенный ItemDefinition) должно быть уникальным в пределах содержащего ItemDefinition или itemComponent.

Альтернативным способом определения экземпляра ItemDefinition является указание элемента FunctionItem, который определяет сигнатуру функции: ее параметры и выходные данные. Экземпляр ItemDefinition может содержать максимум один элемент FunctionItem. FunctionItem может содержать ноль или больше параметров, определяемых как InformationItems а также один выходной тип, определяемый как typeRef. Имена параметров элемента FunctionItem должны быть уникальными.

Элемент ItemDefinition ДОЛЖЕН быть определен с использованием только одного из альтернативных способов:

- при помощи ссылки на вложенный или импортированный typeRef, возможно, ограниченный при помощи allowedValues;
- при помощи сложения элементов ItemDefinition.
- при помощи элемента сигнатуры функции.

Элемент ItemDefinition уточняет NamedElement и наследует его атрибуты и ассоциации моделей.

Таблица 26: Атрибуты и ассоциации модели ItemDefinition

Атрибут	**Описание**
typeRef: String [1]	Данный атрибут определяет базовый тип ItemDefinition при помощи префикса пространства имён.
typeLanguage: String [0..1]	Данный атрибут идентифицирует язык типа, используемый для указания базового типа ItemDefinition. Это значение переопределяет язык типа, указанный в элементе Definitions. Язык ДОЛЖЕН быть указан в формате URI.
allowedValues: UnaryTests [0..1]	Данный атрибут перечисляет возможные значения или диапазоны значений в базовом типе, которые разрешены в ItemDefinition.
itemComponent: ItemDefinition[*]	Данный атрибут определяет ноль или более вложенных ItemDefinitions, которые входят в один ItemDefinition.
IsCollection: Boolean	Если значение *true* отмечено флагом, фактические значения, определенные ItemDefinition, являются наборами допустимых значений. По умолчанию используется *false*.
functionItem: FunctionItem [0..1]	Этот атрибут описывает опциональный FunctionItem, который является частью этого ItemDefinition.

Таблица 27: Атрибуты и ассоциации модели `FunctionItem`

Атрибут	Описание
outputtypeRef: String [0..1]	Ссылка на выходной тип функции
parameters: InformationItem [0..*]	Параметры функции как элементы `InformationItems`

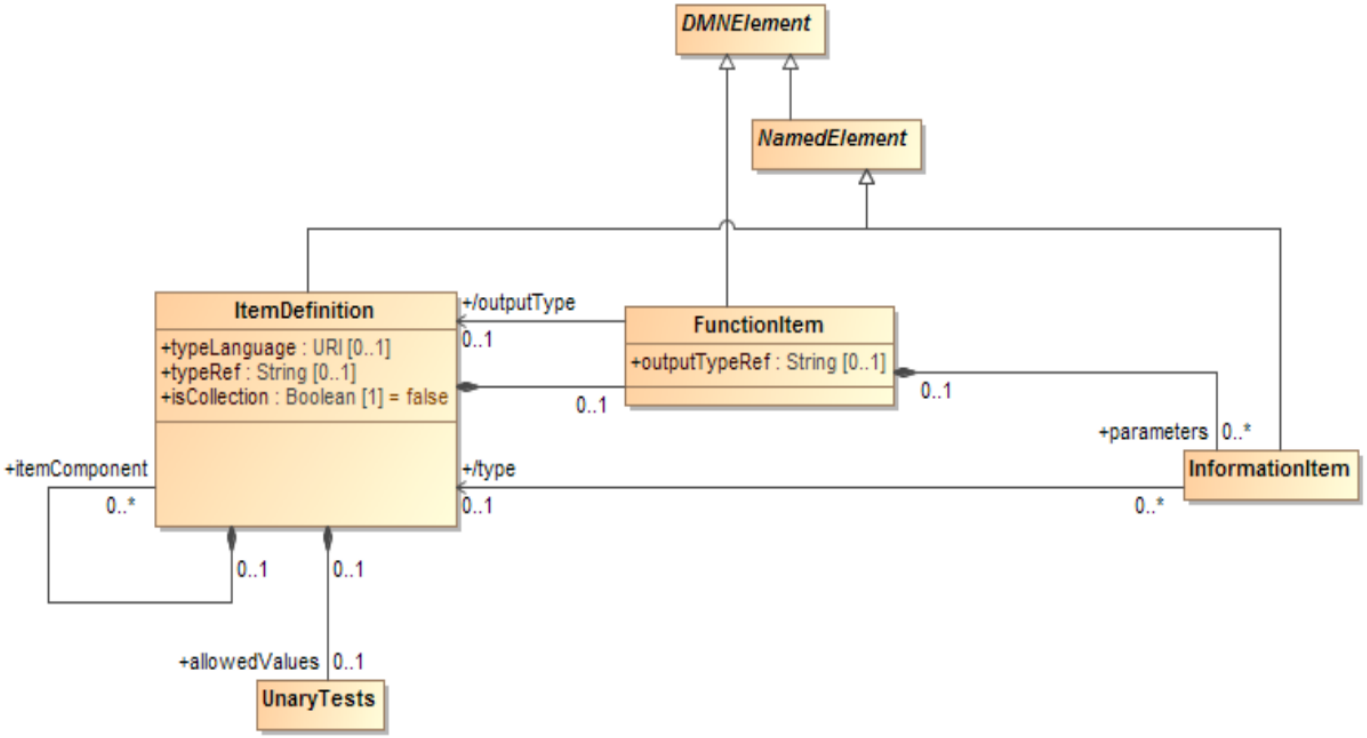


Рисунок 7.7: Диаграмма класса ItemDefinition

7.3.4 Метамодель InformationItem

Класс `InformationItem` используется для моделирования переменных на уровне логики решения в моделях решений.

`InformationItem` является конкретным подклассом элемента `NamedElement`, от которого он наследует `id` и необязательные атрибуты `name`, `description`, и `label`. Однако у элемента `InformationItem` ДОЛЖЕН быть атрибут `name`, который является именем, используемым для представления элемента `InformationItem` в других элементах `Expression`. Имя элемента `InformationItem` ДОЛЖНО быть уникальным в рамках его области действия.

Переменные представляют значения:

- которые являются результатом принятия решения;
- присваиваются входным данным внешним источником данных;
- или передаются модулю логики решения, который определяется как функция (представлен элементом модели бизнес-знаний).

В первом или во втором случае другие зависимые решения могут ссылаться на переменную через требования к информации. В третьем случае переменная является одним из параметров функции, являющейся реализацией элемента модели бизнес-знаний на уровне логики решения.

Выражение значения логики решения в элементе `Decision`, который содержит элемент `InformationRequirement`, ДОЛЖНО ссылаться на переменную, представляющую экземпляр `Decision` или `InputData`, на который ссылается `InformationRequirement`. Параметр в экземпляре `BusinessKnowledgeModel` ДОЛЖЕН быть переменной в выражении значения данного элемента `BusinessKnowledgeModel`.

Элементу `InformationItem`, содержащемуся в `Decision`, присваивается значение, заданное выражением значения `Decision`:

- элемент `Binding` присваивает значение элементу `InformationItem`, являющемуся параметром в `FunctionDefinition` как части экземпляра `Invocation`;
- внешний источник знаний, который привязывается на время выполнения, присваивает значение элементу `InformationItem`, содержащемуся в `InputData`;
- выражение значения `ContextEntry` присваивает значение элементу `InformationItem`, содержащемуся в `ContextEntry`.

В любом случае обозначенный атрибутом `typeRef` тип данных, связанный с экземпляром `InformationItem`, ДОЛЖЕН быть совместим с типом данных, связанным с элементом модели **DMN**, от которого он принимает значения.

`InformationItem` наследует все атрибуты и ассоциации модели от `NamedElement`. В таблице 24 представлены дополнительные атрибуты и ассоциации модели элемента `InformationItem`.

Таблица 24: Атрибуты и ассоциации модели `InformationItem`

Атрибут	**Описание**
valueExpression: Expression [0..1]	`Expression`, значение которого присваивается данному `InformationItem`. Это производный атрибут.

typeRef: : String [1]	Полностью определенное имя типа данного экземпляра `InformationItem`.

7.3.5 Метамодель Literal expression

Класс `LiteralExpression` используется для моделирования выражения значения, значение которого задается текстом на определенном языке выражения.

`LiteralExpression` – это конкретный подкласс `Expression`, от которого он наследует атрибуты `id` и `typeRef`.

У экземпляра `LiteralExpression` есть необязательный строковый атрибут `text` и необязательный строковый атрибут `expressionLanguage`, который определяет язык выражения атрибута `text`. Если `expressionLanguage` не задан, языком выражения `text` является `expressionLanguage`, связанный с содержащим его экземпляром `Definitions`. `expressionLanguage` ДОЛЖЕН быть указан в формате URI. Язык выражения по умолчанию – FEEL.

Будучи подклассом `Expression`, каждый экземпляр `LiteralExpression` имеет значение. `text` в экземпляре `LiteralExpression` определяет его значение в соответствии с семантикой атрибута `expressionLanguage`, заданного `LiteralExpression`. Семантика моделей решений **DMN 1.3**, как описано в этой спецификации, применяется только в том случае, если атрибут `text` всех экземпляров `LiteralExpression` в модели является допустимым выражением в соответствующем языке выражений.

Экземпляр `LiteralExpression` может включать атрибут `importedValues`, который является экземпляром подкласса `Import`, идентифицирующим расположение текста `LiteralExpression`. `importedValues` – это выражение, которое выбирает текст из импортированного документа. Экземпляр `LiteralExpression` НЕ МОЖЕТ иметь сразу два атрибута `text` и `importedValues`. `ImportType`, указанный в `importValues` идентифицирует тип документа, содержащий импортированный текст, и ДОЛЖЕН быть совместим с `expressionLanguage` элемента `LiteralExpression`. Атрибут `expressionLanguage` элемента `importedValues` определяет, каким образом осуществляется выборка текста из импортированного документа. Например, если `importType` указывает XML-документ, `expressionLanguage` заданный в `importedValues` может быть XPATH 2.0.

`LiteralExpression` наследует все атрибуты и ассоциации модели `Expression`. В таблице 29 представлены дополнительные атрибуты и ассоциации модели элемента `LiteralExpression`.

Таблица 29: Атрибуты и ассоциации модели `LiteralExpression`

Атрибут	Описание
text: string [0..1]	Текст `LiteralExpression`. Он ДОЛЖЕН быть действительным выражением в `expressionLanguage`.
expressionLanguage: anyURI [0..1]	Данный атрибут идентифицирует язык выражения, используемый `LiteralExpression`. Это значение переопределяет язык выражения, указанный для содержащего его экземпляра `DecisionRequirementDiagram`. Язык ДОЛЖЕН быть указан в формате URI.
importedValues: ImportedValues [0..1]	Экземпляр `ImportedValues`, который указывает, где находится текст `LiteralExpression`.

7.3.6 Метамодель Invocation

Вызов – это механизм, который позволяет оценивать одно выражение значения – вызванное выражение – внутри другого выражения значения – вызывающего выражения – путем локального связывания входных переменных вызываемого выражения со значениями внутри вызывающего выражения. В вызове входные переменные вызываемого выражения обычно называются *параметрами*. Вызов позволяет повторно использовать одно и то же выражение значения в нескольких выражениях, при этом не дублируя его в качестве суб-выражения во всех используемых выражениях.

Класс `Invocation` используется для моделирования вызовов как своего рода `Expression`. `Invocation` – это конкретная специализация `Expression`.

Экземпляр `Invocation` состоит из нуля или более `binding`, которые являются экземплярами `Binding`, и моделируют, как `bindingFormulas` привязаны к `formalParameters` вызываемой функции. `formalParameters`, указанные в `FunctionDefinition`, являются `InformationItems`. Параметрами `Bindings` являются `InformationItems`. Связывание осуществляется путем сопоставления имён `InformationItem`.

`Invocation` содержит `calledFunction`, `Expression`, которые должны оценивать функцию. Чаще всего это `LiteralExpression`, называющий `BusinessKnowledgeModel`.

Значение экземпляра `Invocation` – это значение соответствующего тела `calledFunction` со значениями `formalParameters`, присвоенными во время выполнения на каждую привязку в `Invocation`.

`Invocation` МОЖЕТ использоваться для моделирования вызовов в моделях решений, когда у элемента `Decision` есть только один элемент `knowledgeRequirement` и когда `decisionLogic` в элементе `Decision` состоит только из вызова элемента `BusinessKnowledgeModel`, на который ссылается `requiredKnowledge`, а более сложное выражение значения не требуется.

Использование экземпляров `Invocation` в качестве `decisionLogic` в элементах `Decision` позволяет повторно использовать `encapsulatedLogic` элемента `BusinessKnowledgeModel` в качестве логики для любого экземпляра `Decision`, требующего `BusinessKnowledgeModel`. При этом каждый требующий элемент `Decision` указывает свои собственные привязки для параметров `encapsulatedLogic`.

Атрибут `calledFunction`, связанный с элементом `Invocation`, ДОЛЖЕН БЫТЬ `encapsulatedLogic` элемента `BusinessKnowledgeModel`, обязательного для элемента `Decision`, который содержит `Invocation`. Элемент `Invocation` ИМЕЕТ только одну привязку для каждого параметра, указанного в `encapsulatedLogic` элемента `BusinessKnowledgeModel`.

`Invocation` наследует все атрибуты и ассоциации модели от `Expression`. В таблице 30 представлены дополнительные атрибуты и ассоциации модели элемента `Invocation`.

Таблица 30: Атрибуты и ассоциации модели `Invocation`

Атрибут	**Описание**
calledFunction: Expression [1]	Выражение, значение которого является функцией
binding: Binding [*]	В данном атрибуте перечислены экземпляры `Binding`, используемые для привязки `formalParameters` указанных в `calledFunction` в рамках данного экземпляра `Invocation`.

7.3.7 Метамодель Binding

Класс Binding используется в элементе Invocation для моделирования привязки formalParameters, определённых для calledFunction, к значениям.

Binding состоит из одной bindingFormula, которая является Expression, и одного параметра parameter, который является InformationItem.

Имена параметров в элементах Binding ДОЛЖНЫ быть подмножеством formalParameters, определенных для calledFunction.

Когда элемент Invocation выполняется, то каждому элементу InformationItem, на который, как на parameter, ссылается binding в элементе Invocation, присваивает во время выполнения значение bindingFormula.

В таблице 31 представлены атрибуты и ассоциации модели элемента Binding.

Таблица 31: Атрибуты и ассоциации модели Binding

Атрибут	**Описание**
parameter: InformationItem	InformationItem с привязкой Binding, от которого зависит calledFunction содержащего экземпляра Invocation
bindingFormula: Expression [0..1]	Экземпляр Expression, к которому привязан parameter в Binding, при оценке экземпляра Invocation.

Эта страница намеренно оставлена пустой

8 Таблица принятия решения

8.1 Введение

Таблица принятия решений является одним из способов выражения логики решения, соответствующей артефакту решения на ДТР. Таблица принятия решений представляет собой табличное представление набора связанных входных и выходных выражений, организованных в правила, указывающие, какая выходная запись применяется к определенному набору входных записей. Таблица принятия решений содержит все (и только) входы, необходимые для определения выхода. Более того, полная таблица содержит все возможные комбинации входных значений (все правила).

Таблицы принятия решений и иерархии таблиц решений успешно зарекомендовали себя в качестве инструмента представления логики решения. Одной из целей **DMN** является стандартизация различных форм и типов таблиц решений.

Таблица принятия решений включает:

- имя информационного элемента: имя информационного объекта, если таковое имеется, для которого Таблица принятия решений является его выражением значения. Это, как правило, название модели решения или бизнес-знаний, для которой Таблица принятия решений обеспечивает логику принятия решения;

- набор входных условий (ноль или более). Каждое входное условие состоит из входного выражения и нескольких опциональных значений для входных записей, которые соответствуют этому условию. Входные записи содержатся в правилах, и входная запись под номером *i* соответствует входному условию под номером *i*;

- набор выходных условий (один или более). Каждое выходное условие состоит из имени и опциональных допустимых значений выходных записей, соответствующих условию. Выходные записи содержатся в правилах, и выходная запись под номером *i* соответствует выходному условию под номером *i*. У выходного условия с одним выходом нет имени. Два и более выходных условия описывают таблицу принятия решений, которая на каждый вызов возвращает контекст, где каждому выходном условию соответствует запись. Если выходных условий несколько, у каждого из них ДОЛЖНО быть имя.

- набор выходов (один или более). В том случае, если у решения есть только один выход, то у него нет имени, а только значение. Два или более выхода называются выходными компонентами. У каждого выходного компонента ДОЛЖНО быть имя. Каждый выход (компонент) ДОЛЖЕН указывать выходную запись для каждого правила. Спецификация имени выходного компонента (при наличии нескольких выходов) и все выходные записи называются выходным условием;

- список условий аннотации (ноль или более). Каждое условие аннотации состоит из имени. Имя каждой аннотации ДОЛЖНО являться частью условия аннотации правила. Записи аннотации содержатся в правилах, запись аннотации под номером *i* соответствует условию аннотации под номером *i*;

- список правил (одно или более) в строках или столбцах таблицы (в зависимости от ориентации), где каждое правило состоит из конкретных входных и выходных записей строки таблицы (или столбца). Если правила выражаются в виде строк, столбцы являются условиями и наоборот.

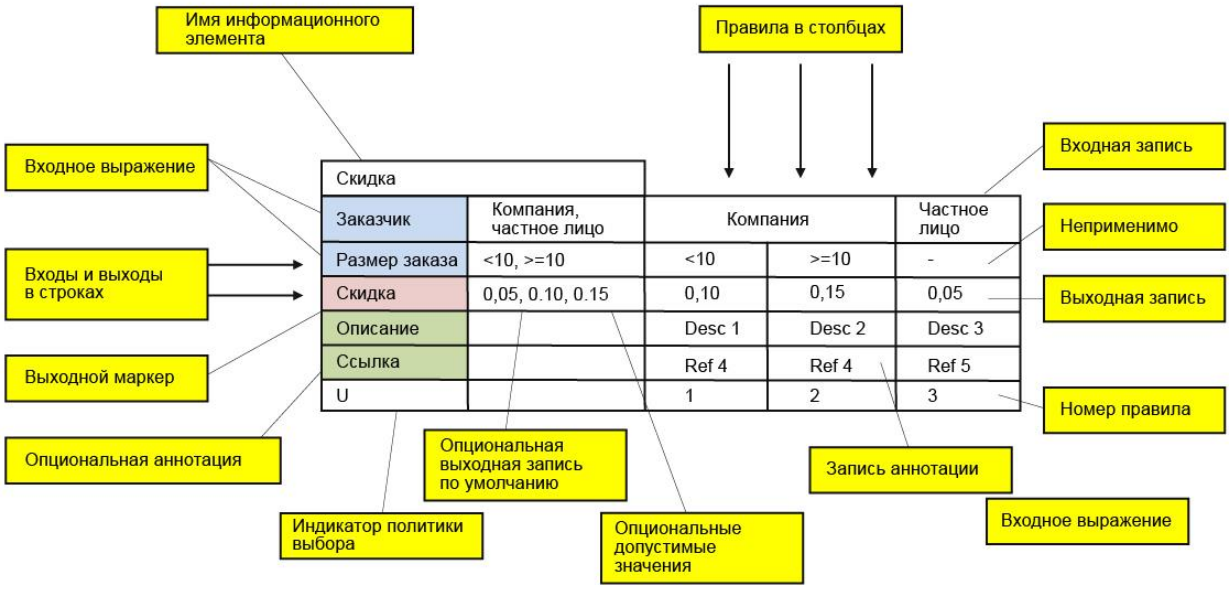


Рисунок 8.1: Пример таблицы принятия решений (вертикальная ориентация: правила располагаются в столбцах)

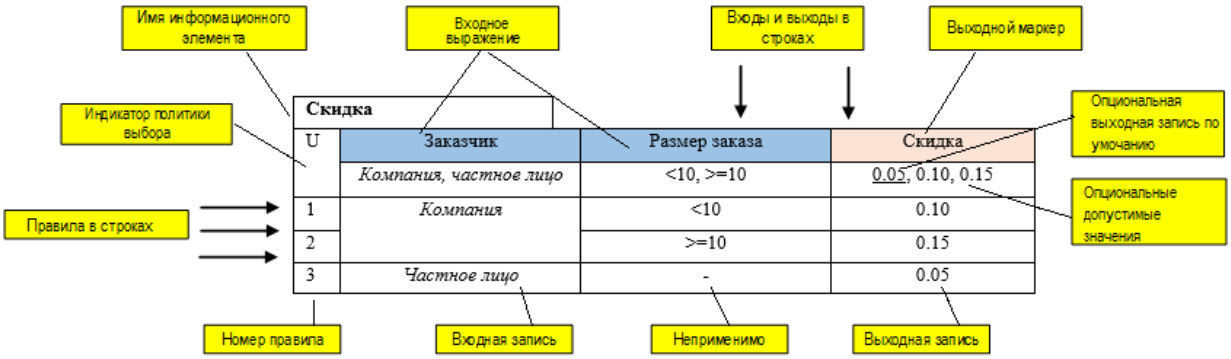


Рисунок 8.2: Пример таблицы принятия решений (горизонтальная ориентация: правила располагаются в строках)

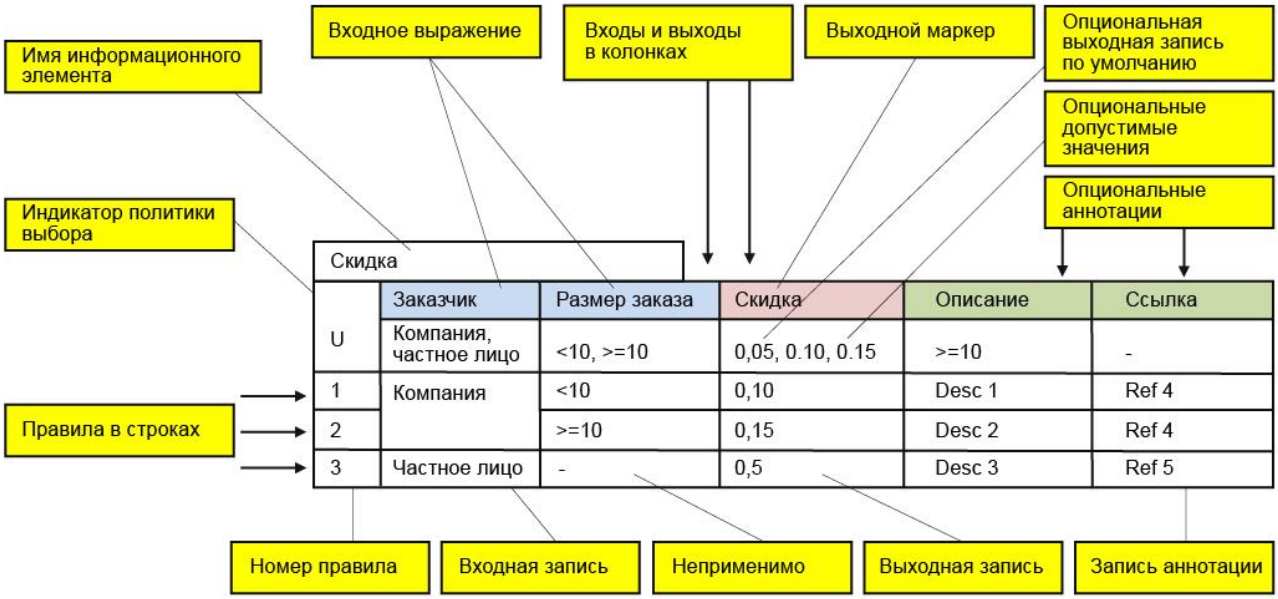


Рисунок 8.3: Пример таблицы принятия решений (вертикальная ориентация: несколько выходных компонентов)

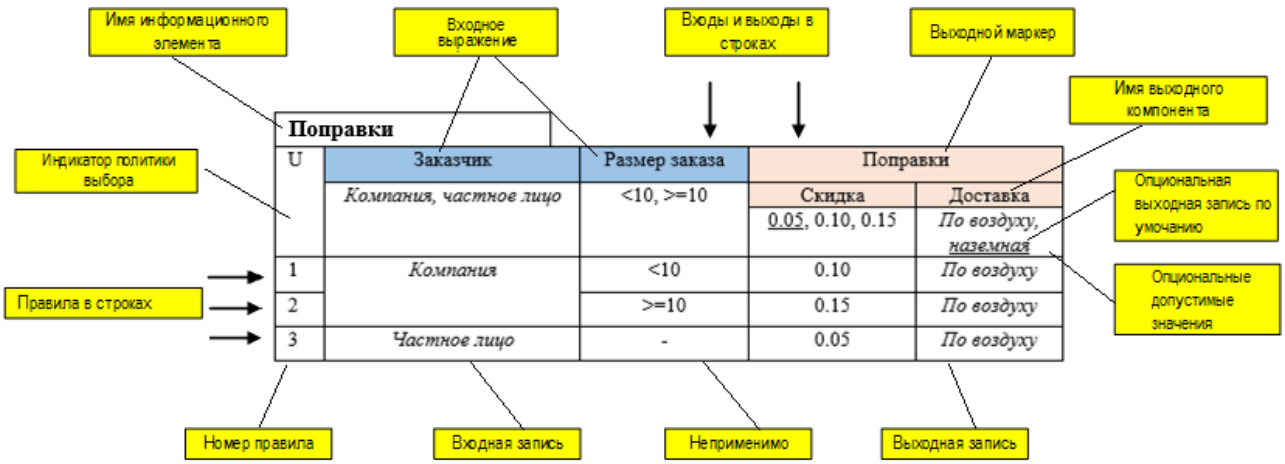


Рисунок 8.4: Пример таблицы принятия решений (горизонтальная ориентация: несколько выходных компонентов)

Таблица принятия решений позволяет отобразить правила в сокращенной форме, путем упорядочивания записей в ячейках. Это позволяет отображать все входные значения в одном и том же порядке для всех правил и дает такие преимущества, как лёгкость чтения и проверки.

Например,

Заказчик	Размер заказа	Скидка
Компания	<10	0.10

где:

If Заказчик = “Компания” **and** Размер заказа < 10 **then** Скидка = 0.10
Обычно это выражается как:

Входное выражение 1	Входное выражение 2	Выходной маркер
Входная запись a	Входная запись b	Выходная запись c

Три выделенные ячейки в вышеприведенном фрагменте таблицы решений представляют следующее правило:

> **Если** значение входного выражения 1 удовлетворяет входной записи a
>
> **и** значение входного выражения 2 удовлетворяет входной записи b,
>
> **то** правило *применимо* и результатом таблицы решений является выходная запись c.

Значение входного выражения *удовлетворяет* входной записи, если значение равно входной записи или принадлежит списку значений, заданных входной записью (например, перечнем или диапазоном) или одно из выражений входной записи имеет значение true. Полное описание удовлетворяющих условий для входной записи находится в разделе 8.3.3. Если входная запись имеет значение «–» (т.е. *неприменимо*), все значения входного выражения удовлетворяют входной записи, при этом такая запись не имеет значения в указанном правиле.

Правило *применимо*, если значение каждого входного выражения удовлетворяет соответствующей входной записи. При отсутствии входных записей любое правило применимо.

Перечень правил выражает логику решения. Для заданного набора входных значений применимое правило (правила) указывает(ют) результирующее значение для выходного имени. Если правила частично *перекрываются*, несколько правил могут стать применимыми. В этом случае *политика выбора* указывает, каким образом обрабатываются все применимые в данном случае правила.

Если две входные записи одного и того же входного выражения не имеют одинаковых значений, записи (ячейки) называются *непересекающимися*. Если есть пересечение, записи называются *перекрывающимися* (или даже равными). «Неприменимо» ('–') может перекрываться любой входной записью входного выражения.

Два правила считаются *перекрывающимися*, если все соответствующие входные записи перекрывают друг друга. Конкретная конфигурация входных данных может соответствовать двум правилам.

Два правила *не пересекаются* (не перекрываются), если хотя бы одна пара соответствующих входных выражений не пересекается. Ни одна конкретная конфигурация входных данных не будет соответствовать двум правилам.

Если в таблицах содержатся перекрывающиеся правила, политика выбора указывает, как обрабатывать такие правила и каковы результирующие значения для выходного имени, что позволяет избежать несогласованности.

8.2 Нотация

Этот подпункт основывается на общей нотации для логики принятия решения и табличных выражений, которая описывается в пункте 7.2. «Нотация».

Графическое представление таблицы решений стандартизирует следующие аспекты:

- ориентация (правила в виде строк, столбцов или кросс-таблицы), как показано в таблице;
- размещение входов, выходов и (опционально) допустимых значений в стандартных местах на сетке ячеек. Каждое входное выражение опционально может быть связано с унарными тестами,

ограничивающими допустимые входные значения. В этом тексте необязательные ячейки с допустимыми значениями указаны белым шрифтом на темном поле. Каждый выход (компонент) опционально может быть связан с допустимыми значениями. В этом тексте необязательные допустимые выходные значения указываются белым шрифтом на темном поле;

- стиль линий и опциональное использование цвета;
- содержимое определенных входных и выходных ячеек ввода правил;
- политика выбора результатов, определяющая, как интерпретировать перекрывающиеся комбинации ввода;
- размещение имен информационных элементов, политики выбора правил (П) и номера правил, как показано на рисунках 8.5, 8.7 и 8.9. Номера правил являются последовательными натуральными числами, начиная с 1. Нумерация правил требуется для таблиц с индикатором F (“first” – в первую очередь) или R (“rule order”– порядок применения правил), поскольку значение зависит от последовательности правил. В кросс-таблицах номера правил не используются. Нумерация правил необязательна для других типов таблиц.

Входные выражения, входные и выходные значения, входные и выходные записи могут быть любым текстом (например, текстом, составленным на естественном языке, формальном языке, псевдокоде). ПО для моделирования, претендующие на 2 или 3 уровень соответствия, ДОЛЖНЫ поддерживать синтаксис (S-)FEEL. ПО для моделирования, претендующие на уровень соответствия 1, могут не интерпретировать выражения. Во избежание неправильного толкования, СЛЕДУЕТ указывать, какое входное выражение не предназначено для интерпретации с использованием URI: http://www.omg.org/spec/DMN/uninterpreted/20140801. Например, в тех случаях, когда выражение изначально не является выражением (S-) FEEL, но может противоречить внешнему виду синтаксиса (S-) FEEL.

8.2.1 Стиль линий и цвет

Стиль линии является нормативным. Входные и выходные условия разделяются двойной линией, которая затем проходит между входными и выходными записями. Двойная линия также разделяет выходные условия и условия аннотаций, а затем проходит между выходными записями и записями аннотаций. Эти две двойные линии параллельны друг другу. Существует третья двойная линия: она пересекает предыдущие две под прямым углом и проходит между входными условиями и входными записями, а затем между условиями аннотаций и записями аннотаций. Другие ячейки разделяются одной линией.

Рекомендуется использовать цветовое обозначение, однако использование цвета не дает дополнительной смысловой нагрузки для содержимого таблицы. Считается хорошей практикой использовать разные цвета для входных условий, выходных условий и условий аннотаций, а также отдельным цветом обозначать входные/выходные записи и записи аннотаций (или оставить их без заливки).

8.2.2 Ориентация таблицы

В зависимости от размера, таблица принятия решений может быть представлена горизонтально (правила расположены в строках), вертикально (правила расположены в столбцах) или как кросс-таблица (правила, располагаются в двух плоскостях). Для кросс-таблиц используется только политика выбора правил по умолчанию.

Входы и выходы таблицы решений не должны совмещаться. В горизонтальной таблице все входные столбцы ДОЛЖНЫ быть представлены слева от всех выходных столбцов. В вертикальной таблице все входные строки ДОЛЖНЫ быть представлены над всеми выходными строками. В кросс-таблице все выходные ячейки ДОЛЖНЫ находиться в нижней правой части таблицы.

Таблица ДОЛЖНА быть составлена в соответствии с одним из представленных ниже образцов (см. рис. 8.5, рис. 8.7, рис. 8.9). Ячейки, содержащие белый текст на темном поле, являются необязательными.

Запись '–' во входной ячейке означает «неприменимость». Индикатор политики выбора обозначается заглавной буквой (например, U, A, F, ...).

имя информационного элемента			
H	входное выражение 1	входное выражение 2	выходной маркер
	значение 1а, значение 1б	значение 2а, значение 2б	значение 1а, значение 1б
1	входная запись 1.1	входная запись 2.1	выходная запись 1.1
2		входная запись 2.2	выходная запись 1.2
3	входная запись 1.2	–	выходная запись 1.3

Рисунок 8.5: Правила располагаются в строках – схематическое изображение

Скидка				
U	Заказчик	Размер заказа (OrderSize)	Доставка	Скидка
	Компания, частное лицо, правительственное учреждение	<10, >=10	*В тот же день, обычная*	0, 0.05, 0.10, 0.15
1	*Компания*	<10	–	0.05
2		>=10	–	0.10
3	*Частное лицо*	–	*В тот же день*	0
4			*обычная*	0.05
5	*Правительственное учреждение*	–	–	0.15

Рисунок 8.6: Правила располагаются в строках – пример

имя информационного элемента				
входное выражение 1	значение 1а, значение 1 б	входная запись 1.1		входная запись 1.2
входное выражение 2	значение 2 а, значение 2 б	входная запись 2.1	входная запись 2.2	–
Выходной маркер	значение 1а, значение 1 б	выходная запись 1.1	выходная запись 1.2	входная запись 1.3
H		1	2	3

Рисунок 8.7. Правила располагаются в столбцах – схематическое изображение

Скидка						
Заказчик	*Компания, частное лицо, правительственное учреждение*	*Компания*		*Частное лицо*		*Правительственное учреждение*
Размер заказа	<10, >=10	<10	>=10	–		–
Доставка	*В тот же день, обычная*	–	–	*В тот же день*	*Обычная*	–
Скидка	0, 0.05, 0.10, 0.15	0.05	0.10	0	0.05	0.15
U		1	2	3	4	5

Рисунок 8.8. Правила располагаются в столбцах – пример

Имя информационного элемента			
Выходной маркер		Входное выражение 1	
		Входная запись 1.1	Входная запись 1.2
Входное выражение 2	Входная запись 2.1	Выходная запись 1.1	Выходная запись 1.3
	Входная запись 2.2	Выходная запись 1.2	Выходная запись 1.4

Рисунок 8.9. Правила в виде кросс-таблицы – схематическое изображение (опциональные входные и выходные значения не показаны)

Скидка				
Скидка		Заказчик		
		Компания	*Частное лицо*	*Правительственное учреждение*
Размер заказа	<10	0.05	0	0.15
	>=10	0.10	0	0.15

Рисунок 8.10. Правила в виде кросс-таблицы – упрощенный пример с двумя входами

Скидка					
Скидка		Заказчик, доставка			
		Компания	*Частное лицо*		*Правительственное учреждение*
		–	*В тот же день*	*обычная*	–
Размер заказа	<10	0.05	0	0.05	0.15
	>=10	0.10	0	0.05	0.15

Рисунок 8.11: Правила в виде кросс-таблиц – пример с тремя входами

Возможно использование кросс-таблиц с более, чем двумя входами (как показано на рисунке 8.11).

8.2.3 Входные выражения

Как правило, используются простые входные выражения, например, имя (например, CustomerStatus) или тест (например, Возраст <25).

Порядок входных выражений не связан с порядком исполнения модели в ПО для моделирования.

8.2.4 Входные значения

Результатом входных выражений может быть ограниченное число значений или ограниченный диапазон значений. Важно моделировать эти ожидаемые входные значения, поскольку таблица принятия решений будет считаться полной, если содержащиеся в ней правила охватывают все комбинации ожидаемых входных значений для всех входных выражений.

Независимо от того, как моделируются ожидаемые входные значения, они ДОЛЖНЫ быть исключающими и полными. Под исключающими понимаются непересекающиеся значения. Значение считается полным, если используются все соответствующие входные значения из домена.

Например, следующие два диапазона входных значений перекрываются: <5, <10. Следующие два диапазона являются неполными: <5,> 5.

Список входных значений не является обязательным. Если это предусмотрено, он может представлять собой список унарных тестов, которые должны удовлетворяться соответствующим входом.

8.2.5 Имена информационных элементов, выходные маркеры и имена выходных компонентов

Таблица принятия решений с несколькими выходными компонентами ДОЛЖНА задавать имя для каждого выходного компонента.

Таблица принятия решений, являющаяся выражением значения элемента InformationItem (например, логика принятия решения или формула привязки табличного вызова), ЗАДАЕТ имя InformationItem как имя информационного элемента (Information Item name). Таблица принятия решений, которая не содержится в другом табличном выражении, должна помещать имя информационного элемента в ячейку имени, расположенную выше и примыкающую к таблице.

Таблица решений, содержащаяся в другом табличном выражении, может использовать содержащее выражение для указания имени информационного элемента. Например, имя информационного элемента для таблицы решений, привязанное к параметру функции, является именем параметра функции. Кроме того, в

целях экономии места ячейка «Название информационного элемента» может быть опущена, а вместо нее может использоваться выходной лейбл.

8.2.6 Множественные выходы

В таблице принятия решений можно отобразить множественные выходы (см. рис. 8.12, рис. 8.13, рис. 8.14)

имя информационного элемента				
H	Входное выражение 1	Входное выражение 2	Выходной маркер	
			Выходной компонент 1	Входное выражение 2
	Входное значение 1a, Входное значение 1b	Входное значение 2a, Входное значение 2b	Выходное значение 1a, Выходное значение 1b	Выходное значение 2a, Выходное значение 2b
1	Входная запись 1a	Входная запись 2a	Выходная запись 1.1	Выходная запись 2.1
2		Входная запись 2b	Выходная запись 1.2	Выходная запись 2.2
3	Выходная запись 1b	–	Выходная запись 1.3	Выходная запись 2.3

Рисунок 8.12. Горизонтальная таблица с несколькими выходными компонентами

имя информационного элемента					
входное выражение 1		входное значение 1a, входное значение 1b	входная запись 1a		входная запись 1b
входное выражение 2		входное значение 2a, входное значение 2b	входная запись 2a	входная запись 2b	–
выходной маркер	Выходной компонент 1	выходное значение 1a, выходное значение 1b	выходная запись 1.1	выходная запись 1.2	выходная запись 1.3
	Выходной компонент 2	выходное значение 2a, выходное значение 2b	выходная запись 2.1	выходная запись 2.2	выходная запись 2.3
H			1	2	3

Рисунок 8.13. Вертикальная таблица с несколькими выходными компонентами

имя информационного элемента			
выходной маркер		входное выражение 1	
Выходной компонент 1, Выходной компонент 2		входная запись 1a	входная запись 1b
входное выражение 2	входная запись 2a	выходная запись 1.1, выходная запись 2.1	выходная запись 1.3 выходная запись 2.3
	входная запись 2b	выходная запись 1.2, выходная запись 2.2	выходная запись 1.4, выходная запись 2.4

Рисунок 8.14. Кросс-таблица с несколькими выходными компонентами

8.2.7 Входные записи

Входные записи правил являются унарными тестами (правило грамматики 15).

Символ тире ('–') используется для обозначения любого входного значения, т. е. вход не имеет отношения к содержащему его правилу.

Входные записи в унарном тесте ЖЕЛАТЕЛЬНО указывать в виде тире '–' либо подмножества входных значений. Например, если входные значения для входа «Возраст» указаны как *[0..120]*, то входную запись *<0* ЖЕЛАТЕЛЬНО определять как недействительную.

Таблицы, содержащие хотя бы одну входную запись в виде тире '–', называются *редуцированными таблицами*. Остальные таблицы называются *расширенными*.

Таблицы, где каждая входная запись имеет значение *true*, *false* или *'–'*, принято называть *таблицами с ограниченными записями*, но нет необходимости придерживаться этой терминологии.

Оценка входных выражений в таблице принятия решений не оказывает побочного влияния на оценку других входных выражений. Это означает, что оценка выражения или выполнение правила не должны изменять оценку других выражений или правил той же таблицы. Это особенно важно при использовании политики выбора «First», где правила оцениваются в предопределенной последовательности: оценка или выполнение правила не должны влиять на другие правила.

8.2.8 Объединенные ячейки ввода правил

Смежные ячейки ввода из разных правил с одним и тем же содержимым и одинаковыми предшествующими ячейками (или без таковых) могут быть объединены, как показано на рисунке 8.15 и 8.16. Ячейки вывода правила не могут быть объединены (кроме кросс-таблиц).

имя информационного элемента			
H	входное выражение 1	входное выражение 2	Выходной маркер
	входное значение 1a, входное значение 1b	входное значение 2a, входное значение 2b	выходное значение 1a, выходное значение 1b
1	входная запись 1a	входная запись 2a	выходная запись 1.1
2		входная запись 2b	выходная запись 1.2
3	входная запись 1b	–	выходная запись 1.3

Рисунок 8.15. Правильное использование смежных ячеек ввода правил

имя информационного элемента			
H	входное выражение 1	входное выражение 2	Выходной маркер
	входное значение 1a, входное значение 1b	входное значение 2a, входное значение 2b	выходное значение 1a, выходное значение 1b
1	входная запись 1a	входная запись 2a	выходная запись 1.1
2		входная запись 2b	выходная запись 1.2
3	входная запись 1b	входная запись 2b	выходная запись 1.3
4		входная запись 2a	выходная запись 1.4

Рисунок 8.16. Недопустимое использование смежных ячеек ввода правил

8.2.9 Выходная запись

Выходная запись правила является выражением.

Ячейки вывода правила не могут быть объединены (кроме кросс-таблиц, где смежные выходные ячейки с одним и тем же контентом могут быть объединены).

Сокращенная нотация

В вертикальных таблицах (правила в виде столбцов) с единственным выходным именем (аналогичным имени информационного элемента) может использоваться сокращенная нотация: выходное значение применяется (‘X’) или не применяется (‘–’), согласно общей практике использования таблиц принятия решений.

Поскольку для выходного имени существует только одна выходная запись, каждое правило должно указывать не более одного символа ‘X’. Другие выходные записи должны содержать символ ‘–’.

Таблица на рис. 8.17 представляет собой сокращенный вариант таблицы, изображенной на рисунке 8.18. Нотация, использованная для заполнения таблицы, считается сокращенной, поскольку выходные записи не записываются (перезаписываются) в каждом столбце, а обозначаются при помощи одного символа ('X' или ' – '). Тем самым экономится пространство в вертикальных таблицах, которые имеют тенденцию расширяться по мере увеличения количества правил. Выходные значения записываются только один раз, перед правилами в части выражения выхода.

Если указано имя информационного элемента и есть только одно выходное имя (которое должно совпадать с именем информационного элемента), выходное имя использовать необязательно.

Категория риска заявителя					
Возраст заявителя	< 25		[25..60]	> 60	
Анамнез	*неотягощенный*	*отягощенный*	–	*неотягощенный*	*отягощенный*
Низкий	X	–	–	–	–
Средний	–	X	X	X	–
Высокий	–	–	–	–	X
U	1	2	3	4	5

Рисунок 8.17: Сокращенная нотация для вертикальных таблиц (правила в виде столбцов)

Категория риска заявителя					
Возраст заявителя	< 25		[25..60]	> 60	
Анамнез	*неотягощенный*	*отягощенный*	–	*неотягощенный*	*отягощенный*
Категория риска заявителя	*Низкий*	*Средний*	*Средний*	*Средний*	*Высокий*
U	1	2	3	4	5

Рисунок 8.18: Полная нотация для вертикальных таблиц (правила в виде столбцов)

8.2.10 Политика выбора результатов

Обычно таблица решений имеет несколько правил. По умолчанию правила не перекрываются. Если правила перекрываются, т.е. к заданному набору входных значений могут применяться несколько правил, используется индикатор политики выбора для распознавания типа таблицы и однозначного понимания логики решения. Политика выбора может использоваться для проверки правильности во время разработки.

Политика выбора определяет результат таблицы решений в случаях перекрывающихся правил, т. е. когда к входным данным может быть применено несколько правил. Для ясности политика выбора обозначается одним символом в определенной ячейке таблицы принятия решений. В горизонтальных таблицах это верхняя левая ячейка (рис. 33), а в вертикальных таблицах это нижняя левая ячейка (рис. 32). В качестве символа используется начальная буква названия соответствующей политики выбора (**U**nique, **A**ny, **P**riority, **F**irst, **C**ollect, Output order или Rule order). Кросс-таблицы всегда уникальны и не нуждаются в индикаторе.

Политика выбора по умолчанию ВСЕГДА – «Unique» (Уникальный). В этом случае использование индикатора выбора является опциональным. Таблицы принятия решений с политикой «Unique» НЕ ДОЛЖНЫ содержать перекрывающиеся правила.

Инструменты для моделирования могут поддерживать только непустое подмножество вариантов политики выбора. Однако, тип таблицы ДОЛЖЕН быть четко определен, и поэтому использование индикаторов политики выбора является обязательным для всех таблиц, за исключением таблиц с политикой «Unique» по умолчанию. Инструменты для моделирования ДОЛЖНЫ обеспечивать возможность использования таблиц принятия решений с политикой «Unique».

Таблицы с одним или несколькими результатами

Таблица с одним результатом должна возвращать выход только одного правила. Таблица с несколькими результатами может возвращать выходы нескольких правил (или функцию выходов, например, сумму значений). Если допускается перекрытие правил, политика выбора указывает, как интерпретировать перекрывающиеся правила.

Первая буква названия политики выбора также определяет, сколько результатов будет в таблице: один или множество.

Не зависимо от того, применяется ли политика с одним или несколькими результатами, некоторые столбцы в таблице решений могут быть обозначены как *аннотации правил*. Аннотации правил содержат текст, который не возвращается как часть результатов выражения, поэтому они игнорируются при проверках выбора результата, описанных ниже. Несмотря на отсутствие стандартного механизма для того, чтобы во время исполнения получить доступ к аннотациям применимых правил в таблице, во внедрениях могут быть использованы аннотации в различных целях, в том числе для аудита, поиска ошибок, логирования, документирования, аналитики и использования в системах, идущих следом в технологической цепочке.

Политика выбора одного результата для таблиц решений с одним выходом**:**

1. **U**nique (уникальный): перекрытие правил невозможно, правила не пересекаются. Только одно правило может применяться. **U**nique является политикой выбора по умолчанию.

2. **A**ny (любой): перекрытие правил допустимо, но все применимые правила показывают одинаковые выходные записи для каждого выхода (при игнорировании аннотаций правил), поэтому можно использовать любую запись. Если выходные записи неодинаковые (при игнорировании аннотаций правил), политика выбора неверна и результат не определен.

3. **P**riority (приоритет): применимы несколько правил, при этом они возвращают различные выходные записи. При использовании политики выбора **P**riority возвращается результат одного применимого правила с самым высоким приоритетом выхода. Приоритеты выхода указываются в упорядоченном списке выходных значений в порядке убывания приоритета. Обратите внимание, что приоритеты не зависят от последовательности правил.

4. **F**irst (первый): допускается, что параметрам применимости соответствуют несколько (перекрывающихся) правил, при этом они возвращают различные выходные записи. При использовании политики выбора **F**irst возвращается первый результат, согласно порядку применения правил (после этого оценка может быть остановлена). Эта политика выбора имеет широкое применение, поскольку она позволяет устранить несоответствия, при помощи принудительного выбора первого результата. Однако использование таблиц с такой политикой выбора не считается рациональным, поскольку оно не дает четкого представления о логике принятия решения. Важно отличать этот тип таблиц от других, потому что результат зависит от порядка применения правил. Последнее правило часто является обработчиком оставшихся значений. Поскольку используется установленный порядок применения правил, таблицу трудно проверить вручную, и, следовательно, нужно использовать с осторожностью. Таблица с несколькими результатами может возвращать выходные записи из нескольких правил. Результатом будет являться список выходов правил или простая функция выходов.

Политика выбора нескольких результатов для таблиц решений с одним выходом:

5. **O**utput order **(**Порядок вывода результатов): возвращает все результаты в порядке убывания приоритета. Приоритеты выхода указаны в упорядоченном списке выходных значений в порядке убывания приоритета.

6. **R**ule order (Порядок применения правил): возвращает все соответствующие результаты в порядке применения правил. Примечание: значение может зависеть от последовательности правил.

7. **C**ollect (Сбор): возвращает все результаты в произвольном порядке. Чтобы применить простую функцию к выходам, можно добавить операторы ('+', '<', '>', '#'). Если оператор не применяется, результатом будет список всех выходных записей.

   Операторы для политики выбора Collect:

   a) + (sum): результат таблицы решений представляет собой сумму всех выходов;
   b) < (min): результат таблицы решений – выход с наименьшим значением;
   c) > (max): результатом таблицы решений является выход с наибольшим значением;
   d) # (count): результатом таблицы решений является количество выходов.
   Другие политики, подразумевающие более сложные манипуляции с выходами, могут реализовываться путем последующей обработки выходного списка (вне таблицы решений).

Таблицы принятия решений со сложным выходом поддерживают только следующие политики выбора: Unique, Any, Priority, First, Output order, Rule order и Collect без оператора, поскольку невозможно определить оператор для нескольких выходов. Это ограничение не относится к аннотациям правил, которых может быть несколько независимо от принятой политики выбора.

При использовании политик выбора Priority и Output order в таблицах со сложным выходом приоритет определяется по всем выходам, для которых указаны выходные значения. Приоритет для каждого выхода указывается в упорядоченном списке выходных значений в порядке убывания приоритета, а общий приоритет устанавливается путем рассмотрения упорядоченных выходов слева направо в горизонтальных таблицах (т.е. столбцы слева имеют приоритет над столбцами справа) или сверху вниз в вертикальных таблицах. Выходы, для которых не заданы выходные значения, не учитываются, хотя их выходные записи включены в упорядоченный сложный выход.

Так, например, если выполнить запрос со следующими данными: «Возраст = 17, Категория риска = «ВЫСОКИЙ» и «Просмотр долга» = «истина», Таблица правил маршрутизации (рис. 8.19) вернёт выходы всех четырех правил в порядке 2, 4, 3, 1.

Правила маршрутизации						
O	Возраст	Категория риска	Просмотр долга	Маршрут	Уровень проверки	Причина
		НИЗКИ, СРЕДНИЙ, ВЫСОКИЙ		*ОТКЛОНИТЬ, ПЕРЕДАТЬ НА РАССМОТРЕНИЕ, ПРИНЯТЬ*	*УРОВЕНЬ 2, УРОВЕНЬ 1, НЕ ТРЕБУЕТСЯ*	
1	–	–	–	*ПРИНЯТЬ*	*НЕ ТРЕБУЕТСЯ*	*Приемлемо*
2	< 18	–	–	*ОТКЛОНИТЬ*	*НЕ ТРЕБУЕТСЯ*	*Заявитель слишком молод*
3	–	*ВЫСОКИЙ*	–	*ПЕРЕДАТЬ НА РАССМОТРЕНИЕ*	*УРОВЕНЬ 1*	*Высоко рисковое заявление*
4	–	–	true	*ПЕРЕДАТЬ НА РАССМОТРЕНИЕ*	*УРОВЕНЬ 2*	*Задолженность заявителя проверятся*

Рисунок 8.19: Политика выбора Output order с сложным выходом

Примечание 1

Кросс-таблицы являются уникальными и полными по определению и поэтому политика выбора к ним не применяется.

Примечание 2

Последовательность правил в таблице принятия решений не влияет на значение, за исключением таблиц с политикой выбора **F**irst (один результат) и **R**ule order (несколько результатов). Такие таблицы следует использовать с осторожностью.

8.2.11 Выходные значения по умолчанию

Для таблиц можно задать выходы по умолчанию. Значение по умолчанию подчеркивается в списке выходных значений.

8.3 Метамодель

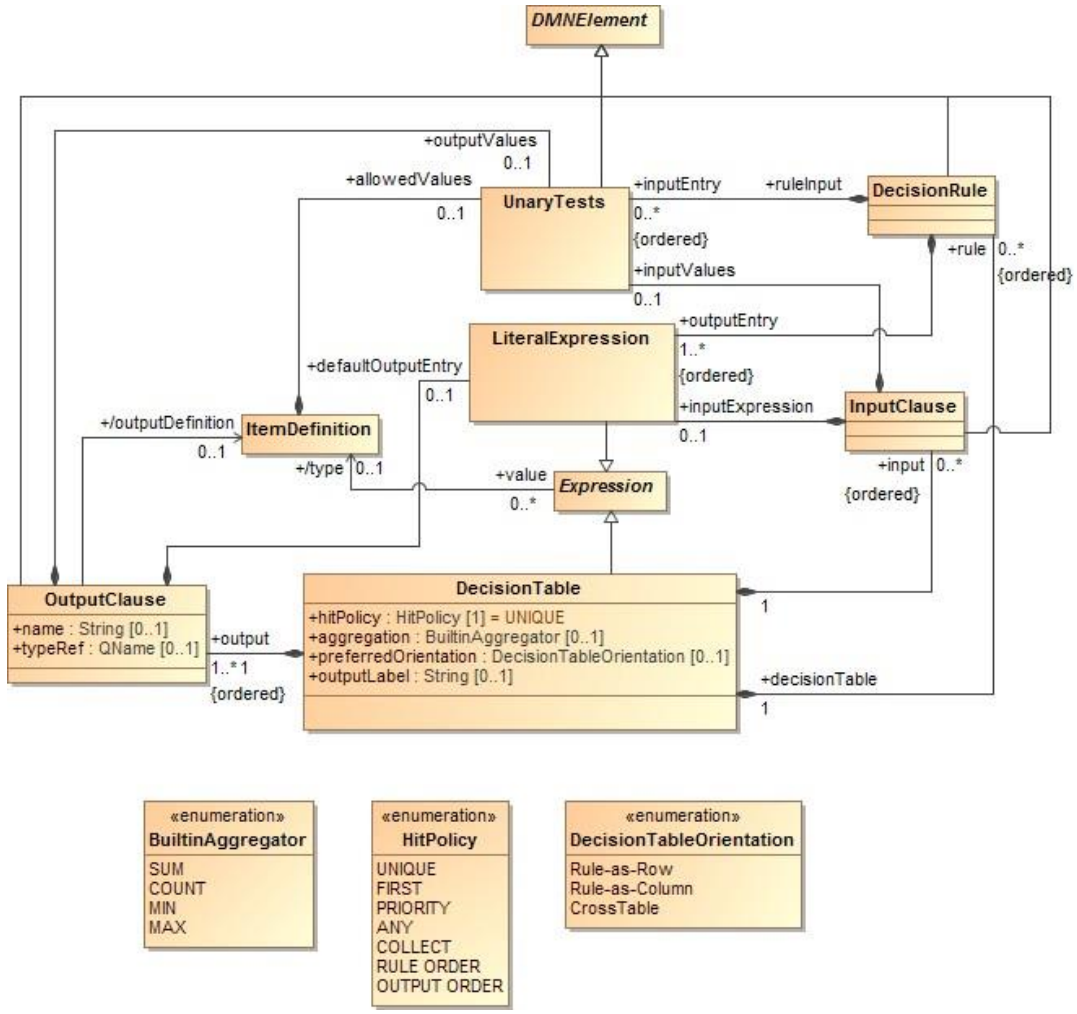


Рисунок 8.20: Диаграмма класса DecisionTable

8.3.1 Метамодель Decision Table

Класс `DecisionTable` используется для моделирования таблиц принятия решений. `DecisionTable` является конкретной специализацией `Expression`.

Экземпляр `DecisionTable` содержит набор правил, которые являются экземплярами `DecisionRule`, список входов, являющихся экземплярами `InputClause` и список выходов, являющихся экземплярами `OutputClause`, а также список аннотаций, являющихся экземплярами `RuleAnnotationClause`.

У `DecisionTable` есть атрибут `preferredOrientation`, который ДОЛЖЕН иметь одно из перечисленных далее значений `DecisionTableOrientation`: `Rule-as-Row`, `Rule-as-Column` или `CrossTable`. ЖЕЛАТЕЛЬНО отображать экземпляр `DecisionTable` в соответствии с его атрибутом `preferredOrientation`, согласно п. 8.2.2.

У экземпляра DecisionTable есть ассоциированный атрибут hitPolicy, который ДОЛЖЕН иметь одно из перечисленных значений HitPolicy: UNIQUE, FIRST, PRIORITY, ANY, COLLECT, RULE ORDER, OUTPUT ORDER. Значение по умолчанию для атрибута hitPolicy: UNIQUE. В диаграммном представлении экземпляра DecisionTable hitPolicy представляется в соответствии с разделом 8.2.10.

Значение, связанное с экземпляром DecisionTable, зависит от связанной с ним hitPolicy, как указано ниже и в разделе 8.2.10. Атрибут hitPolicy экземпляра DecisionTable отображается так, как указано в разделе 8.2.10.

Если атрибут hitPolicy, связанный с экземпляром DecisionTable, имеет значение FIRST или RULE ORDER, то правила, связанные с DecisionTable, ДОЛЖНЫ быть упорядочены. Упорядочивание выполняется путем явной нумерации правил в диаграммном представлении DecisionTable.

Если атрибут hitPolicy, связанный с экземпляром DecisionTable, имеет значение PRIORITY или OUTPUT ORDER, то outputValues определяют результат согласно разделу разделом 8.2.10.

Если атрибут hitPolicy, связанный с экземпляром DecisionTable, имеет значение COLLECT, у него МОЖЕТ быть связанный атрибут aggregation, с одним из перечисленных значений одним BuiltinAggregator (см. 8.2.10).

Являясь разновидностью Expression, экземпляр DecisionTable имеет значение, которое зависит от результатов связанных правил, а также ассоциированных атрибутов hitPolicy и aggregration, если таковые имеются. Значение экземпляра DecisionTable определяется в соответствии со следующей спецификацией, приведенной в разделе 10.3.2.10.

DecisionTable наследует все атрибуты и ассоциации модели от Expression. В таблице 28 представлены дополнительные атрибуты и ассоциации модели элемента DecisionTable.

Таблица 32. Атрибуты и ассоциации модели элемента DecisionTable

Атрибут	Описание
input: InputClause [*]	В данном атрибуте перечислены экземпляры InputClause, которые входят в DecisionTable.
output: OutputClause [*]	В данном атрибуте перечислены экземпляры OutputClause, которые входят в DecisionTable.
annotation: RuleAnnotationClause [*]	В данном атрибуте перечислены экземпляры RuleAnnotationClause, которые входят в DecisionTable.
rule: DecisionRule [*]	В данном атрибуте перечислены экземпляры DecisionRule, которые входят в DecisionTable.
hitPolicy: HitPolicy	Политика выбора правил, определяющая семантику DecisionTable. Значение по умолчанию: UNIQUE.
aggregation: BuiltinAggregator	Данный атрибут определяет функцию агрегации, которая применяется к неупорядоченному набору значений соответствующих правил rules и позволяет определить значение DecisionTable, если значение атрибута hitPolicy равно COLLECT.

preferredOrientation: DecisionTableOrientation [0..1]	Предпочтительная ориентация для схематического представления `DecisionTable`. `DecisionTable` ЖЕЛАТЕЛЬНО отображать в соответствии со значением данного атрибута.
outputLabel: string[0..1]	Данный атрибут дает описание выхода таблицы решений и часто совпадает с именем `InformationItem`, для которого таблица решений является выражением значения.

8.3.2 Метамодель Decision Table Input и Decision Table Output

В `DecisionTable` вход `input` определяет `inputExpression` (субъект) и количество `inputEntries`. Выход `output` определяет имя, область определения выходного значения и количество `outputEntries`.

Класс `InputClause` используется для моделирования входа таблицы принятия решений, класс `OutputClause` используется для моделирования выхода таблицы решений, а класс `RuleAnnotationClause` используется для моделирования аннотации таблицы решений.

Экземпляр `InputClause` состоит из необязательного `inputExpression` и упорядоченного списка `inputEntry`, которые являются экземплярами `UnaryTests`. Экземпляр `OutputClause` опционально может ссылаться на `typeRef` для указания типа данных. Кроме того, экземпляр `OutputClause` состоит из упорядоченного списка атрибутов `outputEntry`, которые являются экземплярами `LiteralExpression`, и опционального атрибута `defaultOutputEntry`, который также является экземпляром `LiteralExpression`. Экземпляр `RuleAnnotationClause` содержит имя типа Строка.

Если `DecisionTable` содержит несколько `OutputClause`, у каждого `OutputClause` ДОЛЖНО быть имя. Если `DecisionTable` содержит один `OutputClause`, у `OutputClause` НЕ ДОЛЖНО быть имени. У `RuleAnnotationClause` имя должно быть.

В таблицах 33, 34 и 35 представлены атрибуты и ассоциации моделей `InputClause`, `OutputClause` и `RuleAnnotationClause` соответственно.

Таблица 33. Атрибуты и ассоциации модели InputClause

Атрибут	Описание
inputExpression: Expression [0.1]	Субъект `InputClause`
inputValues: UnaryTests [0.1]	Данный атрибут содержит `UnaryTests`, которые ограничивают результаты `inputExpression` в данном `InputClause`.

Таблица 34. Атрибуты и ассоциации модели OutputClause

Атрибут	Описание
typeRef: QNameString [0..1]	Атрибут `OutputClause` таблицы принятия решений с одним выходом НЕ ДОЛЖЕН задавать `typeRef`. Атрибут `OutputClause` таблицы решений с несколькими выходами МОЖЕТ задавать `typeRef`. `typeRef` – это имя типа данных выхода, либо `ItemDefinition`, либо базовый тип в указанном expressionLanguage, либо импортированный тип.

name: string [0.1]	`OutputClause` таблицы решения с одним результатом НЕ ДОЛЖЕН указывать имя. `OutputClauses` таблицы решений с несколькими результатами ДОЛЖЕН указывать имя.
outputValues: UnaryTests [0..1]	Данный атрибут содержит `UnaryTests`, которые ограничивают результаты `outputEntries` правил `DecisionRules`, соответствующих данному `OutputClause`.
defaultOutputEntry: Expression [0..1]	В неполной таблице этот атрибут отображает экземпляр `Expression`, который будет возвращаться в случае, если ни одно из правил не применимо к таблице решений.

Таблица 35. Атрибуты и ассоциации модели RuleAnnotationClause

Атрибут	Описание
name: string [1]	`RuleAnnotationClause` ДОЛЖЕН указывать имя, используемое в качестве имени колонки правила аннотации соответствующей таблицы решений.

8.3.3 Метамодель Decision Rule

Класс `DecisionRule` используется для моделирования правил в таблице решений (см. 8.2).

Экземпляр `DecisionRule` содержит упорядоченный список экземпляров `inputEntry`, которые являются экземплярами `UnaryTests`, упорядоченный список экземпляров `outputEntry`, которые являются экземплярами `LiteralExpression`, а также упорядоченный список `RuleAnnotations`.

В табличном представлении содержащего экземпляра `DecisionTable`, представление экземпляра `DecisionRule` зависит от ориентации таблицы решений. Например, если таблица решений представлена горизонтально (правила как строка, см. раздел 8.2.2), экземпляры `DecisionRule` представлены в виде строк, причем все `inputEntrys` представлены слева от всех `outputEntries`, а также `RuleAnnotations`, представленные справа от них.

По определению, элемент `DecisionRule` без `inputEntries` всегда применим. В противном случае считается, что экземпляр `DecisionRule` *применим* только в тех случаях, когда все значения `inputExpression` таблицы `DecisionTable` сопоставимы с соответствующей `inputEntry`.

`inputExpression` удовлетворяет соответствующей `inputEntry` только в тех случаях, когда один из следующих вариантов является true:

a) Одно из выражений `inputEntry` представлено значением, которому равно значение `inputExpression`;

b) Одно из выражений `inputEntry` представлено списком значений и значение `inputExpression` равно хотя бы одному из значений из этого списка;

c) Одно из выражений `inputEntry` является унарным тестом, а унарный тест является true, когда к нему применяется значение `inputExpression`;

d) Одно из выражений `inputEntry` является булевым выражением со специальной переменной ‘?’ и это выражение является true, когда значение `inputExpression` приписывается ‘?’.

Записи `inputEntries` сопоставляются в произвольном порядке.

Элементы `inputEntries` должны располагаться в том же порядке, что и соответствующие им выходы `DecisionTable`.
n-ый `inputExpression` ДОЛЖЕН соответствовать *n*-му элементу `inputEntry` для всех `inputEntries`, чтобы обеспечить *применимость* `DecisionRule`, согласно разделу 8.1.

Элементы `outputEntries` должны располагаться в том же порядке, что и соответствующие им выходы `DecisionTable`.
n-ый `outputEntry` ДОЛЖЕН соответствовать `typeRef` *n*-го `OutputClause`.

Элементы `ruleAnnotation` должны располагаться в том же порядке, что и соответствующие им аннотации `DecisionTable`.
n-ый `ruleAnnotation` ссылается на *n*-ый `ruleAnnotationClause`.

Таблица 36 представляет все атрибуты и ассоциации модели элемента `DecisionRule`. Таблица 37 представляет все атрибуты и ассоциации модели элемента `RuleAnnotation`.

Таблица 36: Атрибуты и ассоциации модели `DecisionRule`

Атрибут	Описание
inputEntry: UnaryTests[0..*]	Экземпляры `UnaryTests`, определяющие условия входа, которые затем `DecisionRule` сопоставляет с соответствующим (по индексу) `inputExpression`.
outputEntry: LiteralExpression [1..*]	Список экземпляров `LiteralExpression`, которые составляют выходные компоненты `DecisionRule`.

Таблица 37: Атрибуты и ассоциации модели `RuleAnnotation`

Атрибут	Описание
text: string [0..1]	Текст `RuleAnnotation`.

8.4 Примеры

В таблице 38 приведены примеры различных типов таблиц принятия решений, описанных в этом подразделе. Дополнительные примеры можно найти в разделе 11.1.4 в контексте исчерпывающего примера использования модели принятия решения **DMN**.

Таблица 38. Примеры таблиц принятия решений

Политика выбора одного применимого правила

Unique

Категория риска заявителя

U	Возраст заявителя	Анамнез	Категория риска заявителя
1	>60	*неотягощенный*	*Средний*
2		*отягощенный*	*Высокий*
3	[25…60]	–	*Средний*
4	<25	*неотягощенный*	*Низкий*
5		*неотягощенный*	*Средний*

Категория риска заявителя

Возраст заявителя	< 25		[25..60]	> 60	
Анамнез	*неотягощенный анамнез*	*отягощенный анамнез*	–	*неотягощенный анамнез*	*отягощенный анамнез*
Категория риска	*Низкий*	*Средний*	*Средний*	*Средний*	*Высокий*
U	1	2	3	4	5

Категория риска заявителя

Возраст заявителя	< 25		[25..60]	> 60	
Анамнез	*неотягощенный анамнез*	*отягощенный анамнез*	–	*неотягощенный анамнез*	*отягощенный анамнез*
Низкий	X	–	–	-–	–
Средний		X	X	X	
Высокий					X
U	1	2	3	4	5

Политика выбора одного применимого правила

Any

Соблюдение кредитных обязательств заемщика

A	Оценка кредитоспособности заемщика (БКИ)	Задолженность по кредитной карте	Остаток задолженности по кредиту на образовательные нужды	Соблюдение кредитных обязательств
1	A	< 10000	< 50000	*Удовлетвори-тельно*
2	Отличная от A	–	–	*Неудовлетвори-тельно*
3	–	>=10000	–	*Неудовлетвори-тельно*
4	–	–	>=50000	*Неудовлетвори-тельно*

Пример: отличная от A, >10000$, >=50000$ -> Неудовлетворительно (правила 2,3,4)

Политика выбора одного применимого правила

Priority

Категория риска заявителя

P	Возраст заявителя	Анамнез	Категория риска заявителя
			Высокий, Средний, Низкий
1	>=25	*неотягощенный*	*Средний*
2	> 60	*отягощенный*	*Высокий*
3	–	*отягощенный*	*Средний*
4	<25	*неотягощенный*	*Низкий*

Политика выбора одного применимого правила

First

Особая скидка

F	Способ заказа	Месторасположение заказчика	Типа заказчика	Особая скидка,%
1	Сайт	США	Предприятие оптовой торговли	10
2	Телефон	–	–	0
3	–	За переделами США	–	0
4	–	–	Предприятие розничной торговли	5

Особая скидка

Способ заказа	Сайт		–
Месторасположение заказчика	США		–
Типа заказчика	Предприятие оптовой торговли	Предприятие розничной торговли	–
Особая скидка,%	10	5	0
F	1	2	3

Пример: Сайт, за пределами США, Предприятие розничной торговли -> 0 (правило 3)

Политика выбора нескольких применимых правил

No order

Длительность отпуска

Возраст	–	<18	>=60	–	[18..60]	>=60	–
Стаж работы	–	–		>=30	[15..30)	–	>=30
Длительность отпуска	22	5	5	5	2	3	3
C+	1	2	3	4	5	6	7

Пример: Возраст =58, Стаж работы =31 -> Результат=сумма (22, 5, 3) =30

Политика выбора нескольких применимых правил

Output order

Длительность отпуска

O	Возраст	Стаж работы	Длительность отпуска
			22,5,3,2
1	–	–	22
2	>=60	–	3
3	–	>=30	3
4	<18	–	5
5	>=60	–	5
6	–	>=30	5
7	[18..60]	[15..30]	2
8	[45..60)	<30	2

Пример: Возраст =58, Стаж работы =31 -> Результат= (22, 5, 3)

Политика выбора нескольких применимых правил

Rule order

Право на получение студенческого финансового пакета

R	Средний академический балл студента	Внепрограммная деятельность студента	Членство в Национальном обществе почета	Право на получение студенческого финансового пакета
1	> 3.5	>= 4	Да	Оплачивается 20% от стоимости обучения
2	> 3.0	–	Да	Займ в размере 30% от стоимости обучения
3	> 3.0	>= 2	Нет	Оплата 20% стоимости обучения в счет заработной платы за выполнение работ при университете
4	<= 3.0	–	–	Оплата 5% стоимости обучения в счет заработной платы за выполнение работ при университете

Пример: Средний академический балл =3.6, Внепрограммная деятельность =4, Членство в Национальном обществе почета -> Результат= (Оплачивается 20% от стоимости обучения, Оплачивается 20% от стоимости обучения)

9 Простой язык выражений (S-FEEL)

9.1 Введение

DMN 1.3 определяет FEEL (friendly enough expression language – достаточно удобный язык выражений), с помощью которого описывается стандартная исполняемая семантика различных типов выражений на модели принятия решений (см. Раздел 10).

В этом разделе определяется простое подмножество FEEL, так называемый язык S-FEEL. С помощью S-FEEL описывается стандартная исполняемая семантика моделей принятия решений, в которых используются только простые выражения. В частности, это модели принятия решений, где логика решения моделируется в основном (или исключительно) с использованием таблиц решений.

Опыт использования S-FEEL с момента его выхода показал, что практически ни одну модель принятия решений нельзя полностью описать, используя S-FEEL. Можно описать отдельные таблицы решений, используя только S-FEEL, но в рамках модели принятия решений как минимум одно решение обычно требует FEEL для описания. Разработчикам и пользователям предлагается использовать и применять полную спецификацию FEEL, а не только подмножество S-FEEL.

9.2 Синтаксис S-FEEL

Синтаксис выражений S-FEEL, используемый в этом разделе, определяется через РБНФ, т.е. в виде подмножества синтаксиса FEEL, см. раздел 10.3.1.2.

Правила грамматики:

1.	выражение – простое выражение;
2.	арифметическое выражение =;
2.a	сложение | вычитание |;
2.b	умножение | деление |;
2.c	возведение в степень |;
2.d	арифметическое отрицание;
3	простое выражение = арифметическое выражение | простое значение | сравнение;
4	простые выражения = простое выражение, {",", простое выражение};
5	простой положительный унарный тест =;
5.a	["<" | "<=" | ">" | "> ="], конечная точка |;
5.b	интервал;
6	интервал = (начало открытого интервала | начало закрытого интервала), конечная точка, "..", конечная точка (конец открытого интервала | конец закрытого интервала);
7	начало открытого интервала = "(" | "]";
8	начало закрытого интервала = "[";
9	конец открытого интервала = ")" | "[";
10	конец закрытого интервала = "]";
11	простые положительные унарные тесты = простой положительный унарный тест, {",", простой положительный унарный тест};
12	простые унарные тесты =;

12.a простые положительные унарные тесты |;

12.b "не", "(", простые положительные унарные тесты ")";

12.c "-";

13 конечная точка = простое значение;

14 простое значение = квалифицированное имя | простой литерал;

15 квалифицированное имя = имя, {"." , имя } ;

16 дополнение = выражение, "+", выражение;

17 вычитание = выражение, "-", выражение;

18 умножение = выражение, "*", выражение;

19 деление = выражение, "/", выражение;

20 возведение в степень = выражение, "**", выражение;

21 арифметическое отрицание = "-", выражение;

22 имя = начало имени, {часть имени | дополнительные символы имен};

23 начало имени = начальный символ имени, {символ части имени};

24 часть имени = символ части имени, { символ части имени };

35 начальный символ имени = "?" | [A-Z] | "_" | [a-z] | [\uC0-\uD6] | [\uD8-\uF6] | [\uF8-\u2FF] | [\u370-\u37D] |[\u37F-\u1FFF] | [\u200C-\u200D] | [\u2070-\u218F] | [\u2C00-\u2FEF] | [\u3001-\uD7FF] | [\uF900-\uFDCF] | [\uFDF0-\uFFFD] | [\u10000-\uEFFFF];

26 символ части имени = символ имени | цифры | \ uB7 | [\ u0300- \ u036F] | [\ u203F- \ u2040];

27 дополнительные символы имени = "." | "/" | "-" | "" "| "+" | «*»;

28 простой литерал = числовой литерал | строковый литерал | булевский литерал | литерал даты и времени;

29 строковый литерал = """, {символ - (""" | интервал по вертикали) | экранированная последовательность для строк}, """;

30 булевский литерал = "истина" | "ложь" ;

31 числовой литерал = ["-"], (цифры, [".", цифры] | ".", цифры);

32 цифра = [0-9];

33 цифры = цифра, {цифра};

34 литерал даты и времени = ("дата" | "время" | "длительность"), "(", строковый литерал, ")";

35 сравнение = выражение, ("=" | "! =" | "<" | "<=" | ">" | "> ="), выражение.

36 пробельный символ = интервал по вертикали | \u0009 | \u0020 | \u0085 | \u00A0 | \u1680 | \u180E | [\u2000-\u200B] | \u2028 | \u2029| \u202F | \u205F | \u3000 | \uFEFF ;

37 интервал по вертикали = [\u000A-\u000D];

38 экранированная последовательность для строк = "\'" | "\"" | "\\" | "\n" | "\r" | "\t" | "\u", hex-цифра, hex-цифра, hex-цифра, hex-цифра;

9.3 Типы данных S-FEEL

S-FEEL поддерживает все типы данных FEEL: *number, string, boolean, days and time duration, years and months duration, time and date (число, строка, булевское значение, продолжительность в днях и*

продолжительность времени, продолжительность в годах и месяцах, дата и время). Однако для некоторых из этих типов данных S-FEEL дает упрощенные определения.

Number (числовой тип данных) в S-FEEL имеет те же литералы и пространства значений, что и тип данных *decimal (десятичное число)* в XML Schema. В ПО для моделирования, допускается ограничить разрядность до 34 десятичных знаков и округлить число до ближайшего значения с четным (нулевым) наименее значимым битом. Стоить отметить, что «*разрядность не отражается в данном пространстве значений: Number 2.0 и Number 2.00 не отличаются*» [XML Schema]. Заметим также, что это пространство значений полностью упорядочено. Определение *Number* в S-FEEL – это упрощённое определение *Number в* FEEL.

Типы данных *string (строка)* и (*Boolean),* используемые в FEEL, поддерживаются в S-FEEL. Тип данных *string* в FEEL имеет те же литералы и пространства значений, что и тип данных *string* в Java и XML Schema. Тип данных *boolean* в FEEL имеет те же литералы и пространства значений, что и типы данных *boolean* в Java и XML Schema.

Тип данных *time (время)*, используемый в FEEL, поддерживается в S-FEEL. Тип данных *time* в FEEL имеет те же лексические пространства и пространства значений, что и тип данных *time* в XML Schema. Обратите внимание, что «*поскольку лексическое представление допускает необязательный индикатор часового пояса, значения time частично упорядочиваются, так как порой бывает невозможно определить порядок двух значений, одно из которых имеет часовой пояс, а другое – нет. Пары значений time с индикаторами часового пояса или без них полностью упорядочены*» [XSD].

Тип данных *date and time* (дата и время), используемый в FEEL, не поддерживается в S-FEEL. Тем не менее, S-FEEL поддерживает тип данных *date,* аналогичный *date and time* в FEEL, но без часов, минут и секунд. Тип данных *date* в FEEL имеет те же лексические пространства и пространства значений, что и тип данных *date* в XML Schema.

Типы данных *days and time duration (продолжительность в днях и продолжительность времени)* и *years and months duration (продолжительность в годах и месяцах),* используемые в FEEL, поддерживаются в S-FEEL. Типы данных *days and time duration* и *years and months duration* имеет те же литералы и пространства значений, что и типы данных dayTimeDuration и yearMonthDuration в Xpath Data Model соответственно. Таким образом, *days and time duration* в FEEL являются производными типа данных duration в XML Schema, с ограничением лексического представления до компонентов день, час, минута и секунда. В то время как *years and months duration* в FEEL определяется из типа данных duration в XML Schema с ограничением лексического представления до компонентов год и месяц.

Типы данных FEEL подробно описаны в разделе 10.3.2.2 «Равенство, тождество и эквивалентность».

9.4 Семантика S-FEEL

S-FEEL содержит только ограниченный набор основных функций, которые являются общими для большинства языков выражения и программирования; семантика функций не противоречит большинству языков выражения и программирования.

Семантика выражений S-FEEL определена в этом разделе в терминах семантики типов данных XML Schema, XQuery 1.0 и XPath 2.0 Data Model, а также с точки зрения соответствующих функций и операторов, определенных XQuery 1.0 и XPath 2.0 Функции и операторы (с префиксом «op:» ниже). Полную отдельную спецификацию семантики можно найти в разделе 10.3.2. «Семантика», где она представлена как часть определения FEEL. В рамках S-FEEL оба приведенных определения эквивалентны и в равной степени нормативны.

Арифметическое сложение и вычитание (правило 2a грамматики) имеют одинаковую семантику:

- op: numeric-add и op: numeric-subtract, когда оба операнда являются числами;
- op: add-yearMonthDurations и op: subtract-yearMonthDurations, когда оба операнда – это продолжительность в годах и месяцах;

- op: add-dayTimeDuration и subtract:dayTimeDurations, когда оба операнда являются продолжительность в днях и продолжительностью времени;

- op: add-yearMonthDuration-to-date и op: subtract-yearMonthDuration-from-date, когда первый операнд – это продолжительность в годах и месяцах, а второй операнд – дата;

- op: add-dayTimeDuration-to-date и op: subtract-dayTimeDuration-from-date, когда первый операнд – это продолжительность в днях и продолжительностью времени, а второй операнд – дата;

- op: add-dayTimeDuration-to-time и op: subtract-dayTimeDuration-from-time, когда первый операнд – это продолжительность дня и времени, а второй операнд – это время.

Кроме того, арифметическое вычитание имеет семантику op: subtract-date или op: subtract-times, когда оба операнда являются датами или временем соответственно.

Арифметическое сложение и вычитание в других случаях не определены.

Арифметическое умножение и деление (правило грамматики 2b) имеют одинаковую семантику, определенную для op: numeric-multiply и op: numeric-divide, соответственно, когда оба операнда являются числами. В других случаях они не определяются. Арифметическое возведение в степень (правило грамматики 2c) определяется как результат повышения первого операнда до степени второго операнда, когда оба операнда являются числами. В других случаях он не определяется.

Арифметическое отрицание (правило 2d грамматики) определяется только тогда, когда операндом является число: в этом случае его семантика соответствует спецификации op: numeric-unary-minus.

Операторы сравнения (правило 35 грамматики):

- для чисел определяются в соответствии со спецификацией op: numeric-equal, op-numeric-less-than и op: numeric-greater-than;

- для дат определяется в соответствии со спецификацией op: date-equal, op: date-less-than и op: date-greater-than;

- операторы сравнения между интервалами времени определяются в соответствии со спецификацией op: time-equal, op: time-less-than и op: time-greater-than;

- продолжительности в годах и месяцах определяются в соответствии со спецификацией op: duration-equal, op: yearMonthDuration-less-than и op: op:year-MonthDuration-greater-than;

- продолжительности в днях и продолжительности времени определяются в соответствии со спецификацией op: duration-equal, op: dayTimeDuration-less-than и op: dayTimeDuration-greater-than.

В случае String и Booleans, можно проверить только их равенство: семантика strings и Booleans определяется в соответствии со спецификацией fn: codepoint-equal и op: Boolean-equal соответственно.

Операторы сравнения определяются только тогда, когда два операнда имеют один и тот же тип, за исключением years and months duration, которые можно проверять на равенство. Следует, однако, отметить, что «*за исключением zero-length duration, ни один экземпляр xs: dayTimeDuration не может быть равен экземпляру xs: yearMonthDuration*». [XFO].

При условии, что o – это выражение, которое нужно проверить, а e1 и e2, конечные точки:

- находятся в интервале (e1..e2), также обозначенном]e1..e2[, тогда и только тогда o> e1 и o < e1;
- находятся в интервале (e1..e2], также обозначенном]e1..e2], тогда и только тогда o> e1 и o ≤ e2;
- находятся в интервале [e1..e2], тогда и только тогда o ≥ e1 и o ≤ e2;

- находятся в интервале [e1..e2), также обозначается [e1..e2[, тогда и только тогда o ≥ e1 и o <e2.

Выражение, которое нужно проверить, удовлетворяет экземпляру простых унарных тестов (правило 14 грамматики) тогда и только тогда, когда выражение является списком и удовлетворяет хотя бы одному простому унарному тесту в списке, либо, когда простые унитарные тесты имеют значение «-».

9.5 Использование выражений S-FEEL

В данном разделе резюмируется, какие типы выражений S-FEEL разрешены для той или иной функции при использовании языка S-FEEL.

9.5.1 Определения элемента

Выражение, которое определяет **допустимое значение**, ДОЛЖНО быть экземпляром *простых унарных тестов* (правило 12 грамматики), в которых допустимыми являются только те значения в определенном или адресуемом типе, которые удовлетворяют тесту.

9.5.2 Вызовы

В привязках вызовов, **формула привязки** ДОЛЖНА быть *простым выражением* (правило 3 грамматики).

9.5.3 Таблицы решений

Каждое **входное выражение** ДОЛЖНО быть *простым выражением* (правило 3 грамматики).

Каждый список **входных значений** ДОЛЖЕН быть экземпляром *простых унарных тестов* (правило грамматики 12).

Каждый список выходных значений ДОЛЖЕН быть экземпляром *простых унарных тестов* (правило 12 грамматики).

Каждая **входная запись** ДОЛЖНА быть экземпляром *простых унарных тестов* (правило грамматики 12).

Каждая **выходная запись** ДОЛЖНА быть *простым выражением* (правило 3 грамматики).

10 Язык выражений (FEEL)

10.1 Введение

В **DMN** вся логика принятия решений представлена в виде *табличных выражений*. В разделе 7.2 была введена концепция табличного выражения и определены два простых вида выражений: литеральны выражения и табличные вызовы. В разделе 8 определены таблицы решений, очень важный вид табличного выражения. Данный раздел посвящен другим видам табличных выражений и тем самым дополняет представленное выше описание логики принятия решения.

Выражения в «полях таблицы» представляют собой выражения FEEL. Аббревиатура FEEL расшифровывается как «Friendly Enough Expression Language». Данному языку присущи следующие характеристики:

- отсутствие побочного действия;
- простая модель данных с числами, датами, строками, списками и контекстами;
- простой синтаксис, разработанный для широкой аудитории;
- трехзначная логика (**true**, **false**, **null**)

Этот раздел также полностью определяет синтаксис и семантику FEEL. Синтаксис задается как грамматика (10.3.1). Подмножество синтаксиса, визуализируемое в виде табличных выражений, задается как метамодель (10.5).

У языка FEEL есть 2 функции в **DMN**:

1. Он используется как текстовая нотация в полях таких табличных выражений, как таблицы решений.

2. FEEL используется как несколько более полный язык описания логики выражений и ГТР, который, главным образом, позволяет описать семантику более простым и унифицированным способом.

10.2 Нотация

10.2.1 Табличные выражения

При написании данного раздела мы опирались на общую нотацию для логики принятия решений и табличных выражений, которая приводилась ранее в разделе 7.2. «Нотация».

Мы определяем графическую нотацию для логики решения, представленной в виде **табличного выражения**. Графическая нотация позволяет разбить модель логики решения на небольшие составляющие, которые могут быть связаны с артефактами ГТР. ГТР вкупе с табличными выражениями образует исчерпывающий, преимущественно графический язык, который полностью определяет модели принятия решений.

Виды **табличных выражений**:

- **таблица принятия решений,**
- **табличные выражения FEEL,**
- **табличный вызов,**
- **табличный контекст,**
- **табличный список,**

- **Отношение,** или,
- **табличная функция.**

Табличные выражения определяются рекурсивно, т.е. табличные выражения могут содержать другие табличные выражения. Верхне-уровневые табличные выражения соответствуют логике решения одного артефакта ГТР. У такого табличного выражения ДОЛЖНА быть ячейка имени, которая содержит имя артефакта ГТР. Ячейка имени может располагаться сверху, как показано на рисунке 10.1:

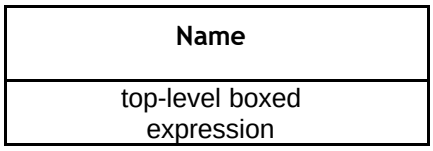


Рисунок 10.1: Табличное выражение

Кроме того, ячейка имени и ячейка выражения могут быть отделены друг от друга пробелом и соединены с левой стороны с помощью линии, как показано на рисунке 10.2:

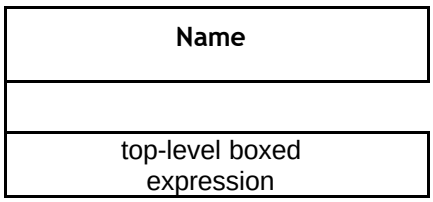


Рисунок 10.2. Табличное выражение с раздельными полями имени и выражения

Ожидается, что графические инструменты будут поддерживать соответствующие графические ссылки, например, при нажатии на фигуру решения откроется таблица принятия решений.

10.2.1.1 Таблицы принятия решений

В приведенных здесь исполняемых таблицах принятия решений используются те же обозначения, что и в таблицах принятия решений, определенные в разделе 8. Их семантика выполнения описывается в разделе 10.3.2.8.

10.2.1.2 Табличные выражения FEEL

Табличное выражение FEEL – это любое выражение FEEL *"e"*, заключенное в ячейку таблицы (см. рис. 10.3), согласно грамматике FEEL (10.3.1):

Рисунок 10.3: Табличное выражение FEEL

Значение табличного выражения, содержащего *e*, равно **FEEL** (*e*, **s**), где **s** – область действия. Область действия включает в себя контекст, полученный из ДТР согласно разделу 10.4, а также любые табличные контексты, содержащие *e*.

Рекомендуется использовать относительно простые *e* и составлять большие табличные выражения из нескольких маленьких.

10.2.1.3 Табличные вызовы

Синтаксис табличного вызова описывается в разделе 7.2.3. Этот синтаксис может использоваться для вызова любой функции, например, модели бизнес-знаний, встроенной функции FEEL, определения табличной функции.

Ячейка с лейблом «вызываемая модель бизнес-знаний» может быть любым табличным выражением, значение которого является функцией, как показано на рисунке 10.4:

Name	
function-valued expression	
parameter 1	binding expression 1
parameter 2	binding expression 2
...	
parameter *n*	binding expression *n*

Рисунок 10.4: Табличный вызов

Табличный синтаксис сопоставляется с текстовым синтаксисом, определяемым правилами грамматики 38, 39, 40, 41. Табличный вызов использует именованные параметры. Позиционный вызов может быть реализован с помощью табличного выражения, содержащего текстовый позиционный вызов.

Для табличного синтаксиса требуется хотя бы один параметр. Функция без параметров должна вызываться с использованием текстового синтаксиса, например, как показано на рисунке 10.5.

function-valued expression()

Рисунок 10.5: Параметрическая функция

Формально значение табличного вызова задается семантикой эквивалентного текстового вызова, например, **функционально-значного выражения** (параметр1: **binding expression1**, parameter2: **binding expression2**, ...).

10.2.1.4 Табличный контекст

Табличный контекст представляет собой набор пар *n* («имя – значение») с опциональным результирующим значением. Каждая пара называется контекстной записью. Контекстные записи могут быть разделены пробелами и соединены слева (сверху) при помощи линии. Таким образом записи контекста могут быть легко идентифицированы, поскольку располагаются по вертикали слева или по горизонтали сверху. Ячейки ДОЛЖНЫ быть организованы одним из следующих способов (см. Рис. 10.6, рис. 10.7):

Name 1	Value 1
Name 2	Value 2
Name n	Value n
Result	

Рисунок 10.6. Вертикальный контекст

Name 1	Name 2	Name *n*		Result
Value 1	Value 2	Value n		

Рисунок 10.7. Горизонтальный контекст

Записи контекста часто используются для декомпозиции сложного выражения на более простые выражения, каждое из которых имеет имя. Такие контекстные записи можно рассматривать как промежуточные результаты. Например, контекст без поля Результат полезен для представления данных кейза (см. рис. 10.8).

Данные заявителя		
Возраст Age	51	
Семейное положение MaritalStatus	“ЗАМУЖЕМ”	
Занятость EmploymentStatus	“ТРУДОУСТРОЕН”	
Существующий клиент ExistingCustomer	false	
Ежемесячно Monthly	Доход	10000,00
	Погашение	2500.00
	Затраты	3000.00

Рисунок 10.8: Использование записей контекста

Контекст с ячейкой окончательного результата удобно использовать для представления вычислений (см. рис. 10.9).

Соответствие критериям	
Возраст	Заявитель. Возраст
Ежемесячный доход	Заявитель. Ежемесячно. Доход
Категория риска до получения оценки БКИ	Возможность. Категория риска до получения оценки БКИ
Возможность частичного платежа	Возможность. Возможность частичного платежа
if Категория риска до получения оценки БКИ = "ОТКАЗАТЬ" или Возможность частичного платежа = false или Возраст <18 или Ежемесячный доход <100 then "НЕ СООТВЕТСТВУЕТ" else "СООТВЕТСТВУЕТ"	

Рисунок 10.9. Использование поля конечного результата

Когда таблицы решений (без ячеек результата) являются контекстными записями, ячейка выхода может использоваться для наименования записи, что позволяет сэкономить место.

В вертикальном контексте можно использовать таблицы принятия решений любого формата. Допускается ступенчатый правый край. Целесообразно также оставлять свободное пространство между элементами контекста см. рис. 10.10.

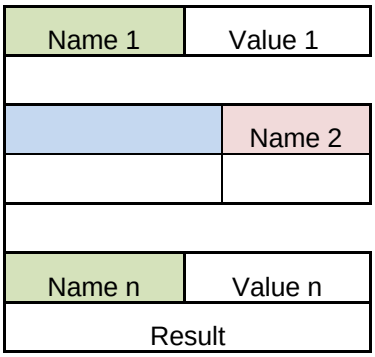


Рисунок 10.10. Вертикальный контекст

Рекомендовано использования цвета.

Допускается использовать ТОЛЬКО утвержденные имена FEEL. Значения и опциональный результат являются табличными выражениями.

В табличном контексте таблица решений может использоваться для представления результата, а именованные элементы контекста – для вычисления входов и определения их имен. Например, (см. рис. 10.11):

Категория риска после оценки БКИ	
Существующий клиент	Заявитель. СуществующийКлиент
Оценка кредитоспособности	Отчет. ОценкаКредитоспособности
Степень риска заявления	Модель доступности (заявитель, продукт). Степень риска заявления

U	Существующий клиент	Степень риска заявления	Оценка кредитоспособности	Категория риска после оценки БКИ
1	true	<=120	<590	“ВЫСОКИЙ”
2			[590..610]	“СРЕДНИЙ”
3			>610	“НИЗКИЙ”
4		>120	<600	“ВЫСОКИЙ”
5			[600..625]	“СРЕДНИЙ”
6			>625	“НИЗКИЙ”
7	false	<=100	<580	“ВЫСОКИЙ”
8			[580..600]	“СРЕДНИЙ”
9			>600	“НИЗКИЙ”
10		>100	<590	“ВЫСОКИЙ”
11			[590..615]	“СРЕДНИЙ”
12			>615	“НИЗКИЙ”

Рисунок 10.11: Использование табличных выражений с таблицами принятия решений

Формально значение табличного контекста представляет собой следующее: **{ Name 1”: Value 1, “Name 2”: Value 2, …, “Name n”: Value n }**, если не указан Result. В противном случае значение будет **{ Name 1”: Value 1, “Name 2”: Value 2, …, “Name n”: Value n, “result”: Result } .result**. Обратите внимание, что жирным шрифтом обозначены элементы семантической области FEEL. Область включает контекст, полученный из ГТР, согласно разделу 10.4.

Записи табличного контекста для контекстов, у которых нет ячейки окончательного результата, могут быть доступны и за пределами контекста (как КИ), при этом должны выполнятся правила, определенные в разделе 10.3.2.11. Записи табличного контекста для контекстов, у которых есть ячейка окончательного результата, не доступны за пределами контекста.

10.2.1.5 Табличный список

Табличный список – это список из *n* элементов. Ячейки ДОЛЖНЫ быть организованы одним из следующих способов (см. рис. 10.12, рис. 10.13):

Item 1
Item 2
Item *n*

Рисунок 10.12: Вертикальный список

Item 1, Item 2, Item *n*

Рисунок 10.13: Горизонтальный список

Стили линий являются нормативными. Элементы представляют собой табличные выражения. Формально значение табличного списка – это простое значение списка, т. е. **[Item 1, Item 2, …, Item *n*]**. Область включает контекст, полученный из ГТР, согласно разделу 10.4.

10.2.1.6 Отношение

Вертикальный список однородных горизонтальных контекстов (без ячеек результатов) может отображаться с именами, которые появляются только один раз в верхней части списка, например, реляционная таблица, как показано на рисунке 10.14:

Name 1	Name 2	Name *n*
Value 1a	Value 2a	Value *n*a
Value 1b	Value 2b	Value *n*b
Value 1*m*	Value 2*m*	Value *nm*

Рисунок 10.14: Отношение

10.2.1.7 Табличная функция

Определение табличной функции является нотацией для параметризованных табличных выражений.

Табличное выражение, связанное с моделью бизнес-знаний, ДОЛЖНО быть определением табличной функции или таблицей решений, входные выражения которой считаются именами параметров.

У табличной функции есть 3 ячейки:

1. Ячейка «Вид». Содержит начальную букву одного из следующих слов;

- **F**EEL;
- **P**MML;
- **J**ava.

Ячейка «Вид» может быть опущена, при использовании функций Feel, включая таблицы принятия решений.

2. Ячейка «Параметры». Содержит 0 или более разделенных запятыми имен, в круглых скобках

3. Ячейка «Тело»: Табличное выражение

Вид	(Параметр1, Параметр2, …)
Тело	

Рисунок 10.15: Определение табличной функции

У функций FEEL со значением FEEL в ячейке «Вид» или с пустой ячейкой тело ДОЛЖНО быть выражением FEEL, ссылающимся на параметры.

У функций, определённых извне, со значением Java в ячейке «Вид» тело ДОЛЖНО быть контекстом, согласно разделу 10.3.2.11.2, а форма сопоставления информации ДОЛЖНА быть формой java.

У функций, определенных извне, со значением PMML в ячейке «Вид», тело ДОЛЖНО быть контекстом, согласно разделу 10.3.2.11.2, а форма сопоставления информации ДОЛЖНА быть формой pmml. Формально значение табличных функций – это простое значение функции, например, FEEL *(function(Parameter1, Parameter2, ...) Body)*, если это функция вида FEEL. В противном случае – FEEL*(function(Parameter1, Parameter2, ...) external Body)*

Область действия включает контекст, полученный из ГТР, согласно разделу 10.4.

10.2.2 FEEL

Подмножество FEEL, определенное в следующем разделе, используется как нотация для табличных выражений. Объектом FEEL может быть число, строка, дата, время, продолжительность, функция, контекст или список объектов FEEL (включая вложенные списки).

Примечание: Объектом JSON является число, строка, контекст (в JSON они называются карты) или список объектов JSON. Таким образом, в этом отношении FEEL является расширением JSON. Кроме того, FEEL предоставляет более дружественный синтаксис для буквенных значений и не требует указания кавычек для контекстных ключей.

Приводимые ниже примеры, помогут вам получить понимание языка FEEL.

10.2.2.1 Сравнение диапазонов

Диапазоны и списки диапазонов отображаются в таблице принятия решений:

- в ячейках входных записей;
- в ячейках входных значений;
- в ячейках выходных значений.

В примерах в таблице 32 эта часть синтаксиса показана при помощи подчеркивания. Строки, даты, время и продолжительность также могут сравниваться с использованием типографских литералов, описанных в разделе 7.2.2.1.

Таблица 39. Сравнение диапазонов FEEL

Выражение FEEL	**Значение**
5 в (<=5)	true
5 в ((5..10])	false
5 в ([5..10])	true
5 в (4, 5, 6)	true
5 в (<5, >5)	false
2012-12-31 в ((***2012-12-25..2013-02-14***))	true

10.2.2.2 Числа

Примеры использования чисел и расчётов приводятся в Таблице 40.

Таблица 40. Числа и расчеты FEEL

Выражение FEEL	Значение
Двоичное число(1, 2)	1.00
.25 + .2	0.45
.10 * 30.00	3.0000
1 + 3/2*2 - 2**3	-4.0
1/3	0.33333333333333333333333333333333
Двоичное число (1/3, 2)	0.33
1 = 1.000	true
1.01/2	0.505
Двоичное число (0.505, 2)	0.50
Двоичное число (0.515, 2)	0.52
1.0*10**3	1000.0

10.3 Полное описание синтаксиса и семантики FEEL

В разделе 9 читатель познакомился с подмножество FEEL в объеме, достаточном для обеспечения соответствия 2 уровня для таблиц принятия решений (см. раздел 2). В данном разделе приводится полное описание используемого в **DMN** языка выражения FEEL в объеме, обеспечивающем соответствие 3 уровня. FEEL – это простой язык, образцом для которого послужили такие языки, как Java, JavaScript, XPath, SQL, PMML, Lisp и др.

Синтаксис определяется с помощью правил грамматики, которые показывают, как сложные выражения составляются из более простых. Аналогично, семантические правила показывают, как смысл сложного выражения состоит из значения составляющих их простых выражений.

DMN полностью определяет значение выражений **FEEL**, которые не вызывают определяемые извне функции. Семантика никогда не определяется ПО для моделирования. Выражения FEEL (которые не вызывают определяемые извне функции) не имеют побочных эффектов и имеют одинаковую интерпретацию во всех ПО для моделирования. ЖЕЛАТЕЛЬНО, чтобы функции, определяемые извне, были детерминированными и не имели побочных эффектов.

10.3.1 Синтаксис

Синтаксис FEEL определяется в данной нотации как грамматика, а также как диаграмма класса UML на метамодели (10.5).

10.3.1.1 Грамматическая нотация

Правила грамматики используют нотацию ISO РБНФ. Каждое правило определяет нетерминальный символ *S* в терминах других символов *S1, S2,* ... В следующей таблице представлена нотация РБНФ.

Таблица 41. Нотация РБНФ

Выражение FEEL	**Значение**
S = S1 ;	Символ *S* определяется в терминах символа *S1*
S1 | S2	Либо *S1*, либо *S2*
S1, S2	*S1*, за которым следует *S2*
[S1]	*S1* происходит *0* или *1* раз
{S1}	*S1* повторяется *0* или более раз
*k * S1*	*S1* повторяется *k* раз
"and"	буквальный терминальный символ

Мы расширили нотацию ISO, дополнив ее для краткости следующими диапазонами символов:

Диапазон символов имеет следующий синтаксис РБНФ:

```
диапазон символов = "[", символ, обозначающий начало диапазона,"–", символ, обозначающий конец диапазона,"]";
подстрочный символ = символ unicode;
надстрочный символ = символ unicode;
символ unicode = простой символ | кодовая точка;
кодовая точка = "\ u", 4 *  шестнадцатеричная цифра, | "\U", 6 * шестнадцатеричная цифра;
шестнадцатеричная цифра ="0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"|
                          "a"|"A"|"b"|"B"|"c"|"C"|"d"|"D"|"e"|"E"|"f"|"F"
```

Простым символом является один символ unicode, например, a, 1, $ и т. д. Кроме того, символ может быть задан шестнадцатеричным значением кодовой точки с префиксом *\u*.

Каждый символ unicode имеет значение числовой кодовой точки. Числовое значение символа, обозначающего начало диапазона, должно быть меньше числового значения символа, обозначающего конец диапазона.

Например, шестнадцатеричная цифра может быть описана более кратко с использованием диапазонов символов следующим образом:

```
шестнадцатеричная цифра = [0-9]|[a-f]|[A-F];
```

Обратите внимание, что диапазон символов, который включает все символы unicode, это – *[\ u0- \ u10FFFF]*.

10.3.1.2 Правила грамматики

Ниже приводится полное описание грамматических правил FEEL. Правила грамматики пронумерованы, а в некоторых случаях, альтернативные правила обозначены буквами, чтобы в дальнейшем на них можно было ссылаться. Синтаксис табличных выражений (правило 55) используется для определения семантики исполнения табличных выражений.

1. выражение =;

1.a табличное выражение |;

1.b текстовое выражение;

2. текстовое выражение =;

2.a выражение for | условное выражение | количественное выражение |;

2.b дизъюнкция |;

2.c конъюнкция |;

2.d сравнение |;

2.e арифметическое выражение |;

2.f экземпляр |;

2.g выражение маршрута |выражение фильтра | вызов функции |;

2.h буквальный | простой положительный унарный тест | имя | "(", выражение, ")";

3. текстовые выражения = текстовое выражение, {",", текстовое выражение};

4. арифметическое выражение =;

4.a сложение | вычитание |;

4.b умножение | деление |;

4.c возведение в степень |;

4.d арифметическое отрицание;

5. простое выражение = арифметическое выражение | простое значение;

6. простые выражения = простое выражение, {",", простое выражение};

7. простой положительный унарный тест =;

7.a ["<" | "<=" | ">" | "> ="], конечная точка |;

7.b интервал;

8. интервал = (начало открытого интервала | начало закрытого интервала), конечная точка, "..", конечная точка (конец открытого интервала | конец закрытого интервала);

9. начало открытого интервала = "(" | "]";

10. начало закрытого интервала = "[";

11. конец открытого интервала = ")" | "[";

12. конец закрытого интервала = "]";

13. положительный унарный тест = выражение;

14. положительный унарный тест = положительный унарный тест, {",", положительный унарный тест};

15. унарный тест =

15.a положительный унарный тест |

15.b "не", "(", положительные унарные тесты ")" |

15.c "-"

16. конечная точка = простое значение;

17. простое значение = квалифицированное имя | простой литерал;

18. квалифицированное имя = имя, {"." , имя };

19. дополнение = выражение, "+", выражение;

20. вычитание = выражение, "-", выражение;

21. умножение = выражение, "*", выражение;

22. деление = выражение, "/", выражение;

23. возведение в степень = выражение, "**", выражение;

24. арифметическое отрицание = "-", выражение;

25. имя = начало имени, {часть имени | дополнительные символы имени};
26. начало имени = начальный символ имени, {символ части имени};
27. часть имени = символ части имени, {символ части имени};
28. начальный символ имени = "?" | [A-Z] | "_" | [a-z] | [\uC0-\uD6] | [\uD8-\uF6] | [\uF8-\u2FF] | [\u370-\u37D] |[\u37F-\u1FFF] | [\u200C-\u200D] | [\u2070-\u218F] | [\u2C00-\u2FEF] | [\u3001-\uD7FF] | [\uF900-\uFDCF] | [\uFDF0-\uFFFD] | [\u10000-\uEFFFF];
29. символ части имени = символ имени | цифры | \ uB7 | [\ u0300- \ u036F] | [\ u203F- \ u2040];
30. дополнительные символы имени = "." | "/" | "-" | "" "| "+" | «*»;
31. литерал = простой литерал | "ноль";
32. простой литерал = числовой литерал | строковый литерал | булевский литерал | литерал даты и времени;
33. строковый литерал = "", {символ - ("" | интервал по вертикали)} | последовательность экранирования для строк}, """ ;
34. булевский литерал = "истина" | "ложь";
35. числовой литерал = ["-"], (цифры, [".", цифры] | ".", цифры);
36. цифра = [0-9];
37. цифры = цифра, {цифра};
38. вызов функции = выражение, параметры;
39. параметры = "(", (именованные параметры | позиционные параметры), ")";
40. именованные параметры = имя параметра, ":", выражение, {",", имя параметра, ":", выражение};
41. имя параметр = имя;
42. позиционные параметры = [выражение, {",", выражение}];
43. выражение маршрута = выражение, "." , имя ;
44. для выражения = "for", имя, "in", контекст итерации {",", имя, "in", контекст итерации}, "return", выражение;
45. условное выражение = "if", выражение, "then", выражение, "else" выражение;
46. квантифицированное выражение = ("some" | "every"), имя, "in", выражение, { имя, "in", выражение }, "удовлетворяет", выражение ;
47. дизъюнкция = выражение, "или", выражение;
48. конъюнкция = выражение, "and", выражение;
49. сравнение =

49.a выражение, ("=" | "! =" | "<" | "<=" | ">" | "> ="), выражение |

49.b выражение, "between", выражение, "and", выражение |

49.c выражение, "in", положительный унарный тест |

49.d выражение, "in", "(", положительные унарные тесты ")";

50. выражение фильтра = выражение, "[", выражение, "]";
51. экземпляр = выражение, "экземпляр", "of", тип;
52. тип =

квалифицированное имя |

'список' '<' type '>' |
'контекст' '<' name ':' type { ',' name ':' type } '>' |
'функция' '<' [type { ', ' type }] '>' '->' type
;

53. табличное выражение = список | определение функции | контекст;
54. список = "[" [выражение, {",", выражение}], "]";
55. определение функции = "функция", "(", [формальный параметр {",", формальный параметр}], ")", ["извне"], выражение;
56. формальный параметр = имя параметра [":" тип] ;
57. контекст = "{", [контекстная запись, {",", контекстная запись}], "}";
58. контекстная запись = ключ, ":", выражение;
59. ключ = имя | строковый литерал;
60. литерал даты и времени = литерал at | вызов функции ;
61. пробельный символ = интервал по вертикали | \u0009 | \u0020 | \u0085 | \u00A0 | \u1680 | \u180E | [\u2000-\u200B] | \u2028 | \u2029 | \u202F | \u205F | \u3000 | \uFEFF ;
62. интервал по вертикали = [\u000A-\u000D]
63. контекст итерации = выражение [“..”, выражение];
64. последовательность экранирования для строк = "\"" | "\"" | "\\" | "\n" | "\r" | "\t" | кодовая точка;
65. литерал at = “@”, строковый литерал

Дополнительные правила синтаксиса:

- приоритет операторов определяется порядком альтернатив грамматических правил 1, 2 и 4 от наименьшего к самому высокому. Например, вызов (табличный) имеет более высокий приоритет, чем умножение, умножение имеет более высокий приоритет, чем сложение, и сложение имеет более высокий приоритет, чем сравнение. Сложение и вычитание имеют одинаковый приоритет и, как и все бинарные операторы FEEL, остаются ассоциативными;
- можно использовать комментарии в стиле Java, т. е. "//" до конца строки и / * ... * /.
- В правиле 26 единственные допустимые функции – это встроенные функции *date, time, date and time, and duration.*
- Строка в правиле 65 должна соответствовать синтаксису строки date, строки time, строки date and time и строки duration, как указано в разделе 10.3.4.1.

10.3.1.3 Литералы, типы данных, встроенные функции

FEEL поддерживает литеральный синтаксис чисел, строк, булевых значений, даты, времени, даты и времени, продолжительности и нулевых значений. (См. Правила грамматики, раздел 10.3.1.2. «Правила грамматики»). Литералы могут сопоставляться непосредственно со значениями в семантическом домене FEEL (раздел 10.3.2.1 «Семантический домен»).

FEEL поддерживает следующие типы данных:

- number;
- string;
- boolean;
- days and time duration;
- years and months duration;
- date;
- time;
- date and time
- list;
- range
- context
- function.

10.3.1.4 Токены, имена и пробелы

Выражение FEEL состоит из нескольких токенов, которые могут быть отделены друг от друга пробелами (правило грамматики № 63). Токен представляет собой одну из следующих последовательностей символов Unicode:

- Терминальный символ литерала в любом грамматическом правиле за исключением правила № 30.

  Такие символы в грамматических правилах заключаются в двойные кавычки, например, “и”, “+”, “=”, или
- Последовательность, соответствующая правилу грамматики № 28, 29, 35 или 37

Пробелы (за исключением пробелов внутри строк) являются разделителями для токенов. Большинство грамматических правил применяются к токенам, но не распространяются на пробелы, поскольку последние не являются токенами.

Имя (правило грамматики № 27) – это последовательность токенов. Например, имя `Сумма подоходного налога` определяется, как список токенов **[Сумма, подоходного, налога]**. Имя `Доход + Расход` определяется, как список токенов **[Доход, +, Расход]**. Как следствие, такие имена, как `Номер телефона` с одним пробелом между токенами, считаются идентичными именам с несколькими пробелами между токенами, например, `Номер  телефона`.

Имя НЕ ДОЛЖНО начинаться с терминального символа литерала (Правило грамматики № 28).

Часть имени МОЖЕТ быть терминальным символом литерала (Правило грамматики № 29).

10.3.1.5 Контекст, списки, квалифицированные имена и контекстные списки

Контекст – это карта пар «ключ-значение», которые называются записями контекста. Контекст записывается внутри фигурных скобок. Для разделения записей контекста используются запятые, для разделения ключа и

значения – двоеточие (правило грамматики № 59). Ключ может быть строкой или именем. Значение представляет собой выражение.

Список записывается внутри квадратных скобок. Для разделения элементов списка используются запятые (правило грамматики № 56).

Контексты и списки могут ссылаться на другие контексты и списки, что приводит к формированию ориентированного ациклического графа. Именование происходит на основе маршрута.

Квалифицированное имя (КИ) контекстной записи имеет вид $N_1.N_2 \dots N_n$, где N_1 – это имя контекста в области видимости.

Вложенные списки, встречающиеся в интерпретации $N_1.N_2 \dots N_n$, сохраняются. Например,

[{a: {b: [1]}}, {a: {b: [2.1, 2.2]}}, {a: {b: [3]}}, {a: {b: [4, 5]}}].a.b =
[{b: [1]}, {b: [2.1,2.2]}, {b: [3]}, {b: [4, 5]}].b =
[[1], [2.1, 2.2], [3], [4, 5]]

Вложенные списки могут быть свернуты с помощью встроенной функции *flatten()* (10.3.4 «Встроенные функции»).

10.3.1.6 Неоднозначность

Выражения FEEL ссылаются на InformationItems по их квалифицированному имени (КИ). Части квалифицированного имени разделяются точкой. Например, переменные с несколькими компонентами называются [varName]. [ComponentName]. На импортируемые элементы такие, как InformationItems и ItemDefinitions ссылается квалифицированное имя из пространства имен, в котором первая часть – это имя, определенное через элемент Import. Например, импортированная переменная с несколькими компонентами называется [import name].[varName].[componentName].

Поскольку имена представляют собой последовательность токенов, а некоторые из этих токенов могут быть операторами и ключевыми словами FEEL, контекст необходим для устранения неоднозначности. Примеры, представленные ниже, могут считаться именами или другими выражениями:

- a-b
- a - b
- что если?
- Доходы и расходы

Неоднозначность разрешается с использованием области действия. Имена сопоставляются слева направо с именами в области, и самое длинное совпадение является предпочтительным. В том случае, когда самое длинное совпадение нежелательно, скобки или другие знаки пунктуации (которые не разрешены в имени) могут использоваться для устранения неоднозначности выражения FEEL. Например, чтобы вычесть b из a, если a-b – это имя записи контекста в области, следуют поступить следующим образом*:(a) - (b).* Обратите внимание, не имеет смысла записывать выражение, как *a - b*, используя пробел для разделения токенов. Пробел не является частью последовательности токенов и, следовательно, не является частью имени.

10.3.2 Семантика

Семантика FEEL определяется путем сопоставления фрагментов синтаксиса со значениями в семантической области FEEL. Литералы (статья 10.3.1.3 «Литералы, типы данных, встроенные функции») могут сопоставляться напрямую. Выражения, составленные из литералов, сопоставляются со значениями в семантическом домене с использованием простых логических и арифметических операций над отображаемыми значениями литералов. В общем, семантика любого выражения FEEL состоит из семантики его подвыражений.

10.3.2.1 Семантический домен

Семантический домен (**D**) FEEL состоит из бесконечного числа типизированных значений. Все типы представляют собой упорядоченное множество – решетку L.

Различают:

- простые типы данных, такие как число, логическое значение, строка, дата, время и продолжительность
- сложные типы данных, такие как функции, списки и контексты
- тип Null, который включает только нулевое значение
- особый тип Any, который включает все значения в D

Функция представляет собой лямбда-выражение с лексическим замыканием или внешне определяется Java или PMML. Список представляет собой упорядоченный набор элементов области, а контекст представляет собой частично упорядоченный набор пар («строка – значение»), называемых записями контекста.

Мы используем *курсив*, чтобы обозначать синтаксические элементы и **полужирный шрифт** для обозначения семантических элементов. Например,

FEEL *([1 + 1, 2 + 2])* **- [2, 4]**

Обратите внимание, что мы используем квадратные скобки **[],** выделенные полужирным шрифтом, для обозначения списка в семантической области FEEL, а также числа **2, 4**, выделенные полужирным шрифтом для обозначения десятичных значений в семантической области FEEL.

10.3.2.2 Равенство, тождество и эквивалентность

Семантика равенства описывается в разделе 10.3.2.15. В целом, чтобы результат не был нулевым (non-null), сравниваемые значения должны быть одного и того же вида, например, два числа.

При определении тождества, устанавливается, являются ли два объекта в семантическом домене одним и тем же объектом. Обозначим тест на тождество с помощью инфикса **is** и его отрицание с помощью инфикса **is not**. Например, FEEL ("1" = 1) **is null**. Обратите внимание, что **null** не может быть результатом **is**.

Каждое выражение FEEL e в области s может быть сопоставлено с элементом e в семантическом домене FEEL. Это сопоставление определяет значение *e* в *s*. Сопоставление может быть записано в виде e **is** FEEL (e, s). Два выражения FEEL e 1 и e 2 эквивалентны в области s тогда и только тогда, когда FEEL (e_1, s) **is** FEEL (e_2, s). Если значение s понятно из контекста (или значение s не важно), запись можно сокращенно записать, как e_1 **is** e_2.

10.3.2.3 Семантика литералов и типов данных

Типы данных FEEL описаны в следующих подразделах. Значение типов данных включает:

1. Сопоставление литеральной формы (которая в некоторых случаях является строкой) со значением в семантической области.

2. Точное определение типа данных, которому принадлежит набор значений семантической области, и операций с ними.

Каждый тип данных описывает набор значений (возможно, бесконечный). Наборы для типов данных, определенных ниже, не пересекаются.

Мы используем *курсив* для обозначения литералов и жирный шрифт, чтобы указать значение в семантической области

10.3.2.3.1 number

Использование number в FEEL основано на стандарте IEEE 754-2008 Decimal128. Number ограничивается до 34 десятичных знаков и округляется до ближайшего значения с четным (нулевым) наименее значимым битом. Number – это ограниченный тип precisionDecimal в XML Schema, а также эквивалент BigDecimal в Java и DECIMAL128 в MathContext.

Правило грамматики 37 определяет числовые константы. Литералы состоят из базовых 10 цифр и необязательной десятичной точки. -INF, + INF и NaN не поддерживаются. Нет различия между -0 и 0. Встроенная функция *number (from, grouping separator, decimal separator)* поддерживает более полный формат литерала. Например, FEEL (*number("1.000.000,01", ".", ",")*) = **1000000.01**.

FEEL не поддерживает экспоненциальное представление литерала. Например, 1.2e3 недействителен синтаксис FEEL. Вместо этого необходимо использовать *1.2 * 10 ** 3*.

Типа данных number в FEEL представлен в семантической области как пара целых чисел (**p, s**), при этом **p** представляет собой целое 34-значное число со знаком, несущее информацию о точности, а **s** - масштаб в диапазоне [-6111..6176]. Каждая такая пара представляет число $\mathbf{p}/10_s$. Чтобы указать числовое значение, запишем **value(p,s)**. Например, **value(100,2) = 1**. Если точность для нас не важна, мы можем записать это значение просто **1**. Обратите внимание, что многие разные пары имеют одинаковое значение. Например, **value(1,0) = value(10,1) = value(100,2)**.

Для notNumber, positiveInfinity или negativeInfinity значений не существует. Вместо следует использовать **null**.

10.3.2.3.2 string

Правило 33 грамматики определяет литеральные строки как последовательность символов Unicode в двойных кавычках (см. https://unicode.org/glossary/#character), например, "abc". Поддерживаемый диапазон символов Unicode: [\u0-\u10FFFF]. Строковые литералы описываются правилом 33. Соответствующие кодовые точки Unicode используются для кодирования строкового литерала. Литеральная строка "abc" сопоставляется с семантической областью как последовательность из трех символов Unicode **a, b** и **c**, записанных как "**abc**". Литерал *"\U01F40E"* сопоставляется с последовательностью из одного символа Unicode, записанного как "**ȍ**", который соответствует кодовой точке U+1F40E.

10.3.2.3.3 boolean

Булевы литералы задаются правилом 36 грамматики. Значения в семантической области могут быть **истинными (true)** и **ложными** (**false**).

10.3.2.3.4 time

Тип данных time в FEEL можно выразить при помощи литерала времени (см. правило грамматики № 65) или встроенной функции *time ()* (см. 10.3.4.1). Для представления значений в семантическом домене мы используем литералы времени, выделенные полужирным шрифтом.

В семантической области time является значением типа данных time в XML Schema. Тип данных time может быть представлен последовательностью чисел, отображающей часы, минуты, секунды и опционально смещение по времени относительно всемирного координированного времени (UTC). Если указано смещение по времени, включая смещение = 00:00, значение time указывается в формате UTC и сопоставимо со всеми значениями time в формате UTC. Если смещение времени не указано, time интерпретируется как местное время, отношение которого к времени UTC зависит от правил часового пояса для этого местоположения и может меняться со дня на день. Значение местного времени суток иногда сравнимо с значениями времени UTC, согласно разделу Part 2 Datatypes XML Schema.

Время **t** также может быть представлено как количество секунд с полуночи. Запишем это как $\mathbf{value_t(t)}$. Например, $\mathbf{value_t}$ **(01:01:01) = 3661**.

Функция $\mathbf{value_t}$ является взаимно однозначной, но ее диапазон ограничен [0..86400]. Таким образом, она имеет обратную функцию $\mathbf{value_t}^{-1}$ (x), которая возвращает соответствующее значение time для x, если x

находится в [0..86400]; и **value**$_{t\ -1}$ (y), где y = x – нижнее округление(x/86400) * 86400, если x не находится в [0..86400].

Примечание: таким образом, **value**$_{t\ -1}$ (x) всегда применяется к x modulo 86400. Например, **value**$_{t\ -1}$ (x) (3600) вернет время суток, которое равно «01: 00: 00», **value**$_{t\ -1}$ (90000) также вернет «T01: 00: 00», а **value**$_{t\ -1}$ (-3600) вернет время суток «23: 00: 00», обрабатывая -3600 секунд как один час *до* полуночи.

10.3.2.3.5 date

Date в FEEL можно выразить при помощи литерала данных (см. правило грамматики № 65) или встроенной функции date() (см. 10.3.4.1.). Тип данных date в семантическом домене – это последовательность чисел, представляющая год, месяц, день месяца. Год должен находиться в диапазоне [- 999,999,999..999,999,999]. Для отображения значений в семантическом домене используются литералы даты, выделенные полужирным шрифтом.

При необходимости, включая для функции value***dt*** (см. 10.3.2.3.6), значение date считается эквивалентным значению date time, в котором time – это полночь по UTC (00:00:00).

10.3.2.3.6 date-time

Date и *time* в FEEL может быть выражен с использованием *литерала даты-времени* (см. правило грамматики № 65) или встроенной функции *date and time()* (см. 10.3.4.1). Для представления значений в семантическом домене используются *литералы даты-времени*, выделенные полужирным шрифтом.

Тип данных date and time в семантической области – это последовательность чисел, представляющая год, месяц, день, час, минуты, секунды и опционально смещение по времени относительно всемирного координированного времени (UTC). Год должен находиться в диапазоне [- 999,999,999..999,999,999]. Если указано смещение по времени, включая смещение = 00:00, значение date-time указывается в формате UTC и сопоставимо со всеми другими значениями date-time в формате UTC. Если смещение времени не указано, time считается местным временем в соответствии с правилами часового пояса для этого местоположения. Когда часовой пояс указан, например, в формате IANA tz (см. 10.3.4.1 «Функции преобразования»), значение date-time может быть преобразовано в формат UTC с использованием правил часового пояса для этого местоположения, если это применимо.

Примечание: использование правил преобразования часового пояса целесообразно только для значений date-time, находящихся в ближайшем временном интервале.

Date-time (**d**) в формате UTC, могут быть представлены в виде нескольких секунд с начала отсчета даты и времени (момент времени). Количество секунд между **d** и моментом времени записывается как **value**$_{dt}$**(d)**. Функция **value**$_{dt}$ является взаимно однозначной и поэтому имеет обратную функцию **value**$_{dt\ -1}$. Например, **value**$_{dt\ -1}$ **(value**$_{dt}$ **(d)) = d. value**$_{dt\ -1}$ возвращает **null**, а не дату с годом, выходящим за пределы допустимого диапазона.

10.3.2.3.7 days and time duration

Days и time duration в FEEL может быть выражен с использованием литерала продолжительности (см. правило грамматики № 65) или встроенной функции duration() (см. 10.3.4.1). Для представления значений в семантическом домене используются литералы продолжительности в днях и часах, выделенные полужирным шрифтом. Формат символов в кавычках описан лексическим представлением спецификации Xpath Data Model для типа данных dayTimeDuration. days and time duration в семантической области – это последовательность чисел для выражения продолжительности в днях, часах, минутах и секундах, нормализованная таким образом, чтобы сумма этих чисел была минимизирована. Например, FEEL*(duration("P0DT25H"))* = **P1DT1H**.

Значение days and time duration может быть выражено в виде нескольких секунд. Например, **value**$_{dtd}$ **(P1DT1H) = 90000**. Функция **value**$_{dtd}$ является взаимно однозначной и поэтому имеет обратную функцию **value**$_{dtd\ -1}$**.**

Например, **value**$_{dtd\ -1}$**(90000) = P1DT1H.**

10.3.2.3.8 years and months duration

Years and months duration в FEEL может быть выражен с использованием литерала продолжительности (см. правило грамматики № 65) или встроенной функции duration() (см. 10.3.4.1). Для представления значений в семантическом домене используются литералы продолжительности в годах и месяцах, выделенные полужирным шрифтом. Формат символов в кавычках описан лексическим представлением спецификации Xpath Data Model для типа данных dayTimeDuration. years and months duration в семантическом домене – это последовательность чисел для выражения продолжительности в годах и месяцах, нормализованная таким образом, чтобы сумма этих чисел была минимизирована. Например,
FEEL*(duration("P0Y13M"))* = **P1Y1M**.

Значение years and months duration может быть выражено в виде нескольких месяцев.
Например, **value$_{ymd}$ (P1Y1M) = 13.**

Функция **value$_{ymd}$** является взаимно однозначной и поэтому имеет обратную функцию **value$_{ymd}$** $^{-1}$.

Например, **value$_{ymd}$** $^{-1}$ **(13) = P1Y1M**.

10.3.2.4 Трехуровневая логика

FEEL, подобно SQL и PMML, использует трехуровневую логику для значений истинности. Поэтому **and** и **or** выполняют функции из ***D*** *x* ***D*** → ***D***. Трехуровневая логика используется в языке разметки интеллектуального моделирования (Predictive Modeling Markup Language) для моделирования отсутствующих значений данных.

10.3.2.5 Списки и фильтры

Списки являются неизменными и могут быть вложенными. Доступ к *первому* элементу списка *L* можно получить с помощью *L [1]*, а к последнему элементу – с помощью *L [-1]*. Доступ к *n-ому* элементу с начала списка можно получить, используя *L [n]*, а к *n-ому* элементу с конца – используя *L [-n]*.

Если FEEL (*L*) = **L** – это список в семантическом домене FEEL, первым элементом является FEEL *(L [1])* = **L [1]**. Если **L** не содержит *n* элементов, то **L[n]** имеет значение **null**.

L можно отфильтровать с помощью булева выражения в квадратных скобках. Выражение в квадратных скобках может ссылаться на элемент списка, используя имя *item*, если только элемент списка не является контекстом, содержащим ключ "**item**". Если элемент списка является контекстом, то на записи контекста можно ссылаться внутри выражения фильтров без префикса *'item.'*. Например:

[1, 2, 3, 4] [item > 2] = [3, 4]

[{x:1, y:2}, {x:2, y:3}][x=1] = [{x:1, y:2}]

Выражение фильтра оценивается для каждого элемента в списке, и возвращается список, содержащий только те элементы, где выражение фильтра **истинно**. Например:

[{x:1, y:2}, {x:null, y:3}][x < 2] = [{x:1, y:2}]

Фильтруемое выражение подлежит неявным преобразованиям (10.3.2.9.4) до того, как будет вычислено все выражение.

Для удобства, выборка, сделанная с помощью оператора "." со списком контекстов слева, возвращает список выбранных значений. То есть FEEL(*e.f*, **c**) = **[** FEEL(*f*, **c'**), FEEL(*f*, **c''**)**, …]**, где FEEL(*e*) = **[e', e'', …]**, а **c'** является **c** с записями контекста **e''**, и т. д. Например,

[{x:1, y:2}, {x:2, y:3}].y = [2,3]

10.3.2.6 Контекст

Контекст FEEL представляет собой частично упорядоченную совокупность пар («ключ – выражение»), называемых записями контекста. В синтаксисе ключи могут быть либо именами, либо строками. Ключи

сопоставляются со строками в семантическом домене. Эти строки различаются в контексте. Контекст в домене обозначается полужирным шрифтом со строковыми ключами, например, **{ "key$_1$" : expr$_1$, "key$_2$" : expr$_2$, … }**.

Синтаксис для выбора значения записи с именем *key$_1$* из выражения *m*, оцениваемого в контексте, будет *m.key$_1$*.

Если *key$_1$* не является установленным именем, или, по какой-либо причине, ключ не должен быть строкой, допускается следующий синтаксис: *get value (m, "key$_1$").* Выбор значения по ключу из контекста m в семантической области обозначается как **m.key$_1$** или **get value (m, "value1")**

Чтобы получить список пар «ключ-значение» из контекста *m*, можно использовать следующую встроенную функцию: *get entries(m).*

Например, следующее выражение является верным:

get entries ({key$_1$: "value$_1$"}) [key = "key1"]. value = "value$_1$"

Допускается, что выражение в записи контекста не будет ссылаться на ключ той же записи. Однако при этом оно может иметь ссылку на ключи (квалифицированные имена) из других записей того же контекста, а также на другие значения (квалифицированные имена) в области. Эти ссылки ДОЛЖНЫ быть ациклическими и ДОЛЖНЫ частично упорядочивать записи контекста. Выражения в контексте ДОЛЖНЫ оцениваться в соответствии с этим частичным порядком.

10.3.2.7 Диапазоны

FEEL поддерживает компактный синтаксис для диапазона значений, что особенно полезно для тестовых ячеек таблицы решений. Синтаксически диапазоны могут быть представлены в виде:

a. оператора сравнения и единственной конечной точки (правило грамматики 7.a.)

b. пары конечных точек и флагов включенности, которые указывают, входит ли одна или обе конечные точки в данный диапазон (правило грамматики 7.b.); в этом случае типы конечных точек должны быть тождественны (определение тождественности типов см. в разделе 10.3.2.9.1), а сами конечные точки должны быть упорядочены таким образом, чтобы начало диапазона было меньше или равно (<=) конец диапазона.

Конечные точки могут быть литералами или квалифицированными именами следующих типов: number, string, date, time, date and time, or duration. Ниже приведены примеры допустимых диапазонов:

- < 10
- >= date(“2019-03-31”)
- >= @”2019-03-31”
- <= duration(“PT01H”)
- <= @”PT01H”
- [5 .. 10]
- (birthday .. @”2019-01-01”)

Диапазоны представлены в семантическом домене как типизированный экземпляр типа *range*. Если используется синтаксис с одной конечной точкой и оператором, то другая конечная точка не определяется (представлена значением null), а флаг включенности устанавливается в значение false. Например:

Таблица 42: Примеры значений свойств диапазона

Диапазон	Начальное значение включено?	Начальное значение	Конечное значение	Конечное значение включено?
[1..10]	true	1	10	true
(1..10]	false	1	10	true
<=10	false	null	10	true
>1	false	1	null	false

10.3.2.8 Функции

Литерал функции FEEL задается правилом грамматики № 57. Функции в **DMN** также могут определяться через Function Definition (см. 6.3.9). Сложный тип (*T1, ..., Tn*) → *U* содержит значения функций, которые принимают аргументы типов *T1,. , , , Tn* и возвращают результаты типа *U*, независимо от синтаксиса функции (литерал FEEL или определение функции в **DMN**). В случае с одним аргументом, тип *T* → *U* используется как сокращение для *(T)* → *U*.

10.3.2.9 Отношения между типами

У любого выражения FEEL, выполняемого в определенном контексте, есть значение в домене **D**. Каждое значение можно отнести к какому-то типу. Типы FEEL структурированы (см. рис. 10.16) в виде решетки (упорядоченного множества). При этом можно выделить верхнеуровневый тип *Any* и нижнеуровневый тип *Null*. Структурой определяется соответствие типов друг другу. Например, сравнивать можно только те значения, у которых типы соответствуют друг другу. Нельзя сравнить число с логическим значением или строкой.

type(*e*) – это тип элемента домена **FEEL (e,** *c***)**, где *e* является выражением, определяемым правилом грамматики № 1. Литералы чисел, строк, логических значений, нуля, даты, времени и литералы продолжительности даты и времени соотносятся с соответствующими узлами решётки L. Такие сложные выражения, как список, контексты и функции соотносятся с соответствующими параметризованными узлами решетки L. Например, см. Таблицу 43.

Таблица 43: Примеры типов элементов домена

e	**type(*e*)**
123	number
true	boolean
"abc"	string
date("2017-01-01")	date
["a", "b", "c"]	date
["a", true, 123]	list<Any>
(1..10]	range<number>
>= @”2019-01-01” range	range<date>
{"name": "Peter", "age": 30}	context<”age”:number, “name”:string>

function f(x: number, y: number) x + y	(number, number) → number

DecisionA
где *typeRef* для *DecisionA* будет:
`<itemDefinition name="Employee">`
`<itemComponent name="id">`
`<typeRef>number</typeRef>`
`</itemComponent>`
`<itemComponentname="name">`
`<typeRef>string</typeRef>`
`</itemComponent>`
`</itemDefinition>`
	context<"id":number, "name":string>

BkmA
Где инкапсулированная логика будет:
`<encapsulatedLogic>`
`<formalParameter name="x" typeRef="number"/>`
`<formalParameter name="y" typeRef="number"/>`
`<literalExpression typeRef="number">`
`<text>x + y</text>`
`</literalExpression>`
`</encapsulatedLogic>`
	(number, number) → number

Выражение типа *e*, определенное правилом грамматики № 54, соотносится с узлами решетки **L** по функции **type**(*e*) следующим образом:

- имена примитивных типов данных сопоставляются с одноименными узлами (например, string сопоставляется с узлом **string**)
- *Any* сопоставляется с узлом **Any**
- *Null* сопоставляется с узлом **Null**
- *list*<*T*> сопоставляется с узлом **list** с параметром **type**(*T*)
- *context*(k_1:T_1, …, k_n:T_n> *where* $n \geq 1$ сопоставляется с узлом **context** с параметром k_1: **type**(T_1), …, k_n: **type**(T_n)
- *function*<T_1, …T_n> -> *T* сопоставляется с узлом **function** с сигнатурой **type**(T_1), …, **type**(T_n) -> *type*(*T*)
- Имена типов, определенные в разделе itemDefinitions, сопоставляется аналогично типам контекста (см. Правило выше).
- Если ни одно из вышеперечисленных правил не может быть применено (например, имя типа не существует в модели решения), выражение типа считается семантически неверным.

Существует два вида отношений:

- Тождество (T ≡ S) – типы T и S являются взаимозаменяемыми во всех контекстах.
- Соответствие (T <: S) – экземпляр типа T можно заменить везде, где предполагается использование экземпляра типа S.

10.3.2.9.1 Тождество типов

Тождество (≡) между типами определяется следующим образом:

- Примитивные типы данных тождественны, например, строка ≡ строка.
- Два типа списков *list*<*T*> и *list*<*S*> тождественны, если *T* равно *S*. Например, типы ["a", "b"] и ["c"] тождественны.
- Два типа контекста *context*<k_1 :T_1, …, k_n :T_n> и *context*<l_1 :S_1, …, l_m :S_m> тождественны, если n = m, для каждого *ki* :*Ti* существуют уникальные *lj* :*Sj*, при этом *kj* = *lj* и *Ti* ≡ *S j* для i = 1, n. Типы контекста –

это типы, определенные посредством ItemDefinitions или типы, связанные с литералами контекста FEEL, например, { “name”: “John”, “age”: 25}.

- Два типа функций (T_1 , …, T_n) →*U* и (S_1 , …, S_m) →*V* тождественны, если n = m, T*i* ≡ S*j* для i = 1, n и *U* ≡ *V*

- Два типа диапазона *range* <*T*> и *range* <*S*> тождественны, если *T* тождественно *S*. Например, типы диапазонов [1..10] и [30..40] тождественны.

Тождественность типов транзитивна, т.е. если *type1* тождественен *type2*, и *type2* тождественен *type3*, тогда *type1* тождественен *type3*.

10.3.2.9.2 Соответствие типов

Соответствие типов (<:) определяется следующим образом:

- Соответствие включает тождественность. Если *T* ≡ *S*, то *T* <: *S*

- Для каждого типа *T*, *Null* <: *T* <: *Any*, где *Null* – нижнеуровневый тип, а *Any* – верхнеуровневый.

- Список типа *list*<*T*> соответствует *list*<*S*>, если *T* соответствует *S*.

- Контекст типа *context*<k_1 :T_1, …, k_n :T_n> соответствует *context*<l_1 :S_1, …, l_m :S_m>, если n ≥ m и для каждого l_i :S_i существуют уникальные k_j :T_j, при этом l_i = k_j и T_j <: S_i для i = 1, m.

- Функция типа *(*T_1 , …, T_n *)* →*U* соответствует типу *(*S_1 , …, S_m *)* →*V*, если n = m, S_i <: T_i для i = 1, n и *U* <: *V*.
  Для обеспечения безопасности типов, в функциях FEEL реализованы принципы "контрвариантности типа аргумента" и "ковариантности возвращаемого типа".

- Тип диапазона *range* <*T*> соответствует *range* <*S*>, если T соответствует S.

Соответствие типов транзитивно, т.е. если *type1* соответствует *type2*, а *type2* соответствует *type3*, то *type1* соответствует *type3*.

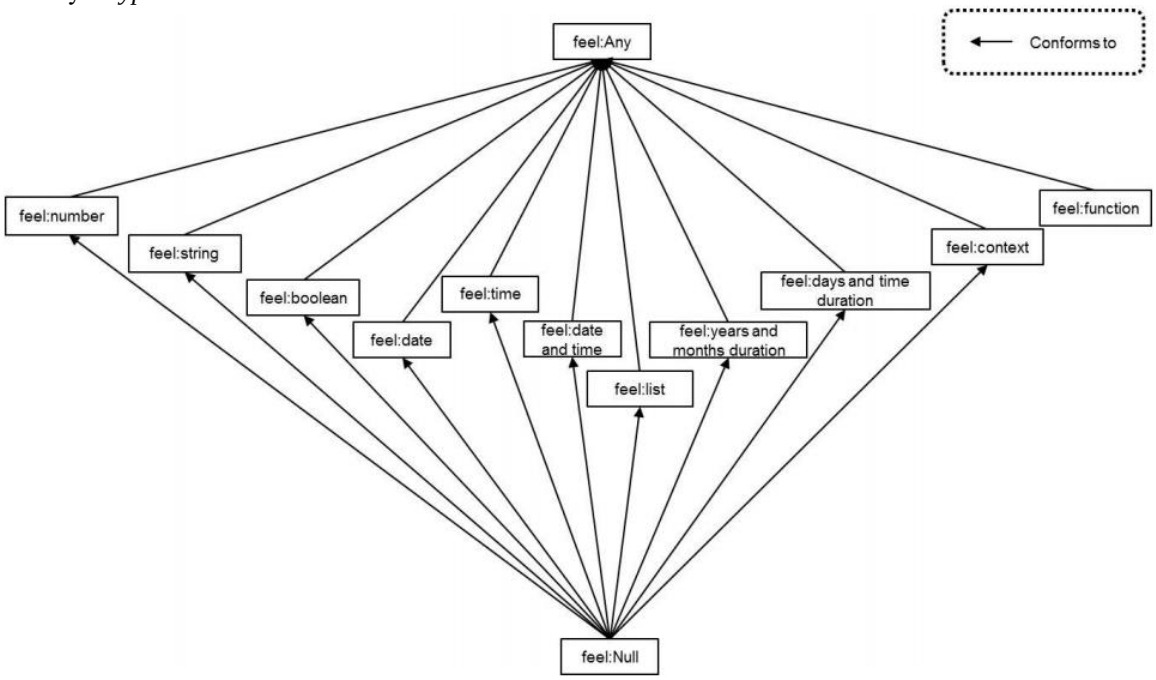


Рисунок 10.16: Структура типов FEEL

10.3.2.9.3 Примеры

Рассмотрим для примера определение ItemDefinitions

```
<itemDefinition name="Employee1">
	<itemComponent name="id">
		<typeRef>number</typeRef>
	</itemComponent>
	<itemComponent name="name">
		<typeRef>string</typeRef>
	</itemComponent>
</itemDefinition>

<itemDefinition name="Employee2">
	<itemComponent name="name">
		<typeRef>string</typeRef>
	</itemComponent>
	<itemComponent name="id">
		<typeRef>number</typeRef>
	</itemComponent>
</itemDefinition>

<itemDefinition name="Employee3">
	<itemComponent name="id">
		<typeRef>number</typeRef>
	</itemComponent>
	<itemComponent name="name">
		<typeRef>string</typeRef>
	</itemComponent>
	<itemComponent name="age">
		<typeRef>number</typeRef>
	</itemComponent>
</itemDefinition>

<itemDefinition isCollection=”true” name="Employee3List">
	<itemComponent name="id">
		<typeRef>number</typeRef>
	</itemComponent>
	<itemComponent name="name">
		<typeRef>string</typeRef>
	</itemComponent>
```

```
        <itemComponent name="age">
              <typeRef>number</typeRef>
        </itemComponent>
</itemDefinition>
```

и решения *Decision1*, *Decision2*, *Decision3* и *Decision4* с соответствующими *typeRefs Employee1*, *Employee2*, *Employee3* и *Employee3List*.

В таблице 44 приведены примеры тождества и соответствия.

Таблица 44: Примеры отношений между типами: тождество и соответствие

type1	**type2**	**тождество**	**соответствие**
number	number	True	True
string	string	True	True
string	date	False	False
date	date and time	False	True
type(Decision1)	**type**(Decision2)	True	True
type(Decision1)	**type**(Decision3)	False	False
type(Decision3)	**type**(Decision1)	False	True
type(Decision1)	**type**({"id": 1,"name":"Peter"})	True	True
type({"id": 1,"name":"Peter"})	**type**(Decision3)	False	False
type({"id": ,"name":"Peter", "age": 45})	**type**(Decision1)	False	True
type({"id": ,"name":"Peter", "age": 45})	**type**(Decision3)	True	True
type([1, 2, 3])	**type**(["1", "2", "3"])	False	False
type([1, 2, 3])	**type**(Decision3)	False	False
type({"id": ,"name":"Peter", "age": 45})	**type**(Decision4)	True	True
type(Decision4)	**type**(Decision3)	False	False
type(function(x:Employee1) →Employee1)	**type**(function(x:Employee1) →Employee1)	True	True

type(function(x:Employee1) →Employee1)	**type**(function(x:Employee1) →Employee2)	True	True
type(function(x:Employee1) →Employee3)	**type**(function(x:Employee1) →Employee1)	False	True
type(function(x:Employee1) →Employee1)	**type**(function(x:Employee1) →Employee1)	False	False
type([1..10])	**type**((20..100))	True	True
type([1..10])	**type**([“a”..”x”])	False	False

10.3.2.9.4 Преобразование типов

Тип выражения FEEL *e* определяется на основании значения **e** = FEEL(e, ***s***) в семантическом домене, где s - набор привязок переменных (см. 10.3.2.11 и 10.3.2.12). Когда выражение появляется в определенном контексте, оно должно быть совместимо с типом, который предполагается использовать в данном контексте, или иными словами, с целевым типом. В некоторых случаях, после того, как тип выражения будет установлен, можно выполнить неявное преобразование типа выражения в целевой тип. Если неявное преобразование является обязательным, но не может быть выполнено, результат будет равен null.

Типы можно преобразовать следующим образом:
- *в одноэлементный список:*
  Когда тип выражения – T, а целевой тип – List <T>, выражение преобразуется в одноэлементный список.

- *из одноэлементного списка:*
  Когда типом выражения является List <T>, значением выражения – одноэлементный список, а целевым типом – T, выражение конвертируется путем разворачивания первого элемента.

- *соответствие:*
  Когда типом выражения является T1, целевым типом является T2, и T1 соответствует T2, значение выражения остается неизменным. В противном случае результат будет **null**.

Различают несколько видов контекстов, в которых могут происходить неявные преобразования:

- Контекст фильтра (10.3.2.5), в котором присутствует выражение фильтра. Фильтруемое выражение подлежит неявному преобразованию *в одноэлементный список*.

- Контекст вызова (Таблица 63), в котором аргумент связан с формальным параметром функции. Аргументы подлежат неявному преобразованию *из одноэлементного списка*.

- Контексты привязок, в которых значение выражения связано с переменной с соответствующей информацией о типе (например, привязка фактических параметров к формальным параметрам в вызове или привязка результата логики решения к выходной переменной решения). Выражение подлежит преобразованию типа *соответствие*.

10.3.2.9.4.1 Примеры

В приведенной ниже таблице представлено несколько примеров преобразования одноэлементных списков.

Таблица 45: Примеры преобразования одноэлементных списков

Выражение	Преобразование	Результат
3[item > 2]	3 преобразуется в *[3]*, поскольку это контекст фильтра, и применяется преобразование *"в одноэлементный список"*	**[3]**
contains(["foobar"], "of")	*["foobar"]* преобразуется в *"foobar"*, так как это контекст вызова, и применяется преобразование *"из одноэлементного списка"*	**false**

В примере, приведенном ниже, перед привязкой переменной *decision_003* к значению "123" преобразование в целевой тип (число) не выполняется, следовательно, переменная связывается с *null*.

```
<decision name="decision_003" id="_decision_003">
        <variable name="descision_003" typeref= "number"/>
        <literalExpression>
              <text> "123"</text>
        </literalExpression>
</decision>
```

10.3.2.10 Таблицы принятия решений

Нормативная нотация таблиц принятия решений приводится в разделе 8. Каждое входное выражение ДОЛЖНО быть текстовым выражением (правило грамматики № 2). Каждый список входных значений ДОЛЖЕН быть экземпляром унарного теста (правило грамматики №17). Проверяемым значением является входное выражение, содержащееся в InputClause. Каждый список выходных значений ДОЛЖЕН быть экземпляром унарного теста (правило грамматики № 17). Проверяемым значением является выбранная выходная запись, содержащаяся в OutputClause. Каждая входная запись ДОЛЖНА быть экземпляром унарного теста (правило грамматики № 17). Аннотации к правилам игнорируются в семантике исполнения.

Компоненты таблицы принятия решений показаны на рисунке 8.5. Они также демонстрируются на метамодели в главе 8.3.

имя информационного элемента			
H	входное выражение 1	входное выражение 2	выходной маркер
	значение 1а, значение 1б	значение 2а, значение 2б	значение 1а, значение 1б
1	входная запись 1.1	входная запись 2.1	выходная запись 1.1
2		входная запись 2.2	выходная запись 1.2
3	входная запись 1.2	–	выходная запись 1.3

Семантика таблицы решений определяется путем перевода литеральных выражений и унарных тестов в логические выражения, которые затем сопоставляются с семантическим доменом и компилируются в набор подходящих правил, которые в свою очередь применяются и, после чего, выводятся как результаты таблицы решений. Наконец, некоторые из выходных выражений таблицы решений сопоставляются с

семантическим доменом и содержат результат интерпретации таблицы решений. Компоненты таблицы решений подробно описаны в таблице 46.

Таблица 46: Семантика таблиц решений

Имя параметра (* означает, что параметр опциональный)	**Описание**
input expression	Одно из входных выражений N> = 0; каждое входное выражение является литеральным выражением.
input values*	Одно из N-входных значений, соответствующих N-входным выражениям. Каждое из них представляет собой литерал унарных тестов (см. ниже).
output values *	Литерал унарных тестов для вывода. (В случае, если выходные компоненты M> 1 (см. Рисунок 8.12), у каждого выходного компонента могут быть собственные выходные значения)
rules	Список правил R> 0. Правило представляет собой список из N входных записей, за которыми следуют M выходные записи. Входная запись представляет собой литерал унарных тестов. Выходная запись является литеральным выражением.
hit policy*	Одно из следующих значений: "U", "A", “P”, “F”, "R", "O", "C", "C+", "C#", "C<", “C>” (по умолчанию "U").
default output value*	Выходное значение по умолчанию является одним из выходных значений. Если M>1, то выходное значение по умолчанию является контекстом с записями, состоящими из имен выходных компонентов и выходных значений.

Унарные тесты (правило грамматики № 15) используются для представления как входных значений, так и входных записей. Считается, что входное выражение *e удовлетворяет* входной записи *t* (с опциональными входными значениями *v*), в зависимости от синтаксиса *t*, следующим образом:

- правило грамматики 15.a: FEEL(*e in (t)*)=**true**;
- правило грамматики 15.b: FEEL(*e in (t)*)=**false;**
- правило грамматики 15.c, когда *v* не указывается: **e != null;**
- правило грамматики 15.c, когда *v* указывается: FEEL(*e in (v)*)=**true.**

Считается, что правило с входными записями $t_1,t_2,...,t_N$ *соответствует* списку входных выражений $[e_1,e_2,...,e_N]$ (с опциональным списком входных значений $[v_1,v_2,...v_N]$) если e_i *удовлетворяет* t_i (с необязательными входными значениями v_i) для всех *i* в 1..N.

Правило применяется, если оно подошло, и политика выбора указывает, что выходное значение подходящего правила должно быть включено в результат таблицы решений. Каждое примененное правило дает одно выходное значение (несколько выходов объединяются в одно значение контекста). Поэтому множество примененных правил требуют агрегации.

Политика выбора обозначается первой буквой одного из следующих названий, набранных полужирным шрифтом.

Политика выбора одного результата:

- **Unique** – только одно правило применимо;
- **Any** – несколько правил могут применяться, но все они имеют одинаковый выход;
- **Priority** – могут применяться несколько правил с разными выходами. Возвращается выход, который находится на первом месте в определенном списке *выходных значений;*
- **First** – возвращает первый результат согласно порядку применения правил.

Политика выбора нескольких результатов:

- **Collect** – возвращает список выходов в произвольном порядке;
- **Rule order** – возвращает список выходов в порядке применения правил;
- **Output order** – возвращает список выходов в порядке, установленном списком *выходных значений*.

Политика выбора Collect может дополнительно задавать агрегацию *aggregation*:

- **C ++** – возвращает сумму выходов;
- **C #** – возвращает количество выходов;
- **C <** – возвращает минимальное значение;
- **C >** – возвращает максимальное значение.

Aggregation определяется с использованием следующих встроенных функций, описанных в подразделе 10.3.4.4. «Функции списка»: *sum, count, minimum, maximum*. Для простоты понимания таблицы решений со сложным выходом не поддерживают агрегацию. К таким таблицам можно применять только следующие политики выбора: *Unique, Any, Priority, First, Collect without operator*, и *Rule order*.

Может быть так, что в таблице принятия решения ни одно правило не будет применяться к набору входных значений. В этом случае результат определяется выходным значением по умолчанию, или если оно не задано – значением **null**. Полная таблица принятия решения НЕ ДОЛЖНА задавать выходное значение по умолчанию.

Семантика вызова таблицы решений **DTI** выглядит следующим образом:

1. Каждое правило в списке правил сопоставляется со списком входных выражений. Сопоставление неупорядоченно.

2. Если ни одно из правил не применимо, то

    а. **DTI** = FEEL(*d*), если задано значение по умолчанию *d*;

    б. **DTI=null**, если значение по умолчанию не задано.

3. В другом случае, допустим, *m* является вспомогательным списком правил, которые соответствуют списку входных выражений. Если политикой выбора является "First" или "Rule order", *m* будет упорядочен в соответствии с номерами правил:

    а. пусть *o* – список выходных выражений, где выражение в индексе *i* является выходным выражением из правила *m[i]*. Выходное выражение правила в таблице решений с одним выходом – это просто выходная запись правила. Выходное выражение таблицы решений с несколькими выходами представляет собой контекст с записями, состоящими из выходных имен и соответствующих выходных записей правила. Если политикой выбора является "Output order", у таблицы решений ДОЛЖЕН быть только один выход, а *o* ДОЛЖЕН быть упорядочен в соответствии с порядком *выходных значений*. Аннотации правил игнорируются в целях определение значения выражения таблицы решений.

    б. если задана политика выбора нескольких результатов, DTI=FEEL(aggregation(o)), где агрегирование является одной из встроенных функций *sum, count, minimum*, как указано в разделе 10.3.4.4. «Функции списка»;

в. в другом случае **DTI**=FEEL(*o[1]*).

10.3.2.11 Область применения и стек контекста

Выражение FEEL *e* всегда оценивается в четко определенном наборе привязок имён, которые используются для распознавания КИ в *e*. Этот набор привязок имён называется областью применения *e*. Область применения моделируется как список контекстов. Область применения **s** содержит контексты с записями, которые находятся в области *e*. Последний контекст в области **s** – это *встроенный* контекст. Рядом с последним контекстом в области **s** находится *глобальный* контекст. Первый контекст в области **s** – это контекст, непосредственно содержащий **e** (если это применимо). Далее располагаются контексты, окружающие *e* (если это применимо).

Квалифицированное имя КИ выражения *e* – это КИ первого контекста в области **s**, с добавлением N, где N – это имя записи в первом контексте **s**, содержащее выражение *e*. КИ в выражении *e* распознаются путем просмотра контекстов в области *s*, начиная с первого и заканчивая последним.

10.3.2.11.1 Локальный контекст

Если выражение *e* обозначает значение записи контекста **m**, то **m** является локальным контекстом для *e*. Кроме того, **m** является первым элементом области **s**. В противном случае *e* не имеет локального контекста и первый элемент области **s** является глобальным контекстом. Однако в некоторых случаях, описываемых ниже, первым элементом **s** является особый контекст.

Все записи контекста **m** являются областью действия для *e*, но график *зависимости* (*depends on*) ДОЛЖЕН быть ациклическим. Это позволяет избегать путаницы с определениями выше: если **m** является результатом оценки контекстного выражения *m*, содержащего *e*, как мы можем узнать его значение, чтобы оценить *e*? Для этого необходимо оценить записи контекста в порядке *зависимости (depends on)*.

10.3.2.11.2 Глобальный контекст

Глобальный контекст – это контекст, который был создан до оценки *e*. Он содержит имена и значения переменных, определенных вне выражения *e*, но которые доступны в *e*. Например, когда *e* – это тело решения *D*, глобальный контекст содержит записи для требований к информации и требований к знаниям *D*. Иными словами он содержит имена и логику моделей бизнес-знаний, решений и сервисов принятия решений, требуемых *D*).

10.3.2.11.3 Встроенный контекст

Встроенный контекст содержит все встроенные функции.

10.3.2.11.4 Специальный контекст

Некоторые выражения FEEL интерпретируются в *специальном* контексте, который располагается перед **s.** Например, выражение фильтра многократно выполняется со специальным первым контекстом, содержащим имя 'item', связанное с последующими элементами списка. Функция выполняется с специальным первым контекстом, который содержит сопоставления: имя аргумента -> значение.

Квалифицированные имена (КИ) в выражениях FEEL интерпретируются относительно **s**. Значение выражения FEEL *e* в области **s** обозначается как FEEL (e, **s**). Можно также сказать, что *e* оценивается по **e** в области **s** или e = FEEL (e, **s**). Заметим, что **e** и **s** являются элементами области FEEL. **s** – список контекстов.

10.3.2.12 Сопоставление FEEL с другими языками

Выражение FEEL *e* обозначает значение **e** в семантической области. В некоторых случаях, возможен обмен типами значений между FEEL и внешними методами Java, моделями PMML и XML, как показано в Таблице 47. Пустая ячейка означает, что преобразование значений в этом случае не предусмотрено.

Таблица 47: Сопоставление между FEEL и другими языками

FEEL value	***Java***	***XML***	***PMML***
number	java.math.BigDecimal	decimal	decimal, PROB-NUMBER, PERCENTAGE-NUMBER
		integer	integer , INT-NUMBER
		double	double, REAL-NUMBER
string	java.lang.String	string	string, FIELD-NAME
date, time, date and time	javax.xml.datatype. XMLGregorianCalendar	date, dateTime, time, dateTimestamp	date, dateTime, time conversion required for dateDaysSince, *et. al.*
duration	javax.xml.datatype. Duration	yearMonthDuration, dayTimeDuration	
boolean	java.lang.Boolean	boolean	boolean
list	java.util.List	contain multiple child elements	array (homogeneous)
context	java.util.Map	contain attributes and child elements	

Иногда нет необходимости оценивать выражение FEEL *e*, но при этом нам требуется установить тип **e**. Следует заметить, что, если у *e* есть КИ, то для вывода типа может потребоваться контекст. Тип элемента области **FEEL**(e, **c**) записывается как **type(***e***)**.

10.3.2.13 Семантика функции

В FEEL различают следующие типы функций:

- встроенные функции, например,
  sum (см. 10.3.4.4), или

- функции, определяемые пользователем, например,
  function(age) age < 21, или

- функции, определяемые извне, например,
  function(angle) external {
  java: {
  class: “java.lang.Math”,
  method signature: “cos(double)”

}}

Встроенные функции FEEL описаны в разделе 10.3.4.

10.3.2.13.1 Встроенные функции

Встроенные функции подробно описаны в разделе 10.3.4. В частности, в нем приводятся сигнатуры функций и

домены параметров. У некоторых функций может быть несколько сигнатур одновремнно.

Встроенные функции вызываются с использованием того же синтаксиса, что и любые другие функции (правило грамматики № 40). Фактические параметры должны соответствовать доменам параметров как минимум в одной сигнатуре до или после применения неявных преобразований, иначе результат вызова будет **null**.

10.3.2.13.2 Функции, определяемые пользователем

Функции, определяемые пользователем (правило грамматики №55), имеют следующий формат:

function(X_1, ... X_n) e

X_1, ... X_n являются формальными параметрами. Каждый формальный параметр имеет вид n_i или n_i: t_i, где n_i – имена параметров, а t_i – их типы. Если тип не указан, то он интерпретируется как *Any*. Значение FEEL(*function(X_1, ... X_n) body*, **s**) – это элемент семантического домена FEEL, который обозначается, как **function(argument list: [**X_1, … X_n**], body:** *body***, scope: s)** (далее сокращенно **f**). Функции FEEL – это лексические замыкания, т. е. *body* – это выражение, которое ссылается на формальные параметры и любые другие имена в области **s**.

Вызов определяемой пользователем функции происходит с помощью того же синтаксиса, что и вызов других функции (правило грамматики № 38). Значение вызова *f* (n_1: e_1, …, n_n: e_n) в области **s** – это значение FEEL(*f*, **s**), применённое к аргументам n_1: FEEL (e_1, **s**)…, n_n: FEEL (e_n, **s**). Это также можно записать как **f (n**$_1$: **e**$_1$…, **n**$_n$: **e**$_n$).

Аргументы **n**$_1$: **e**$_1$…, **n**$_n$: **e**$_n$ соответствуют списку аргументов **[**X_1, … X_n**]**, если **type**(**e**$_i$) соответствует t_i до или после применения неявных преобразований; или t_i не указано в X_i, для всех *i* в *1 ..n*. Результат применения **f** к интерпретируемым аргументам **n**$_1$: **e**$_1$…, **n**$_n$: **e**$_n$ определяется следующим образом. Если **f** не является функцией, или если аргументы не соответствуют списку аргументов, результат вызова будет **null**. В противном случае, пусть **c** будет контекстом с записями **n**$_1$: **e**$_1$…, **n**$_n$: **e**$_n$. Результатом вызова является FEEL(*body*, ***s'***), где **s'**= insert before (**s**, 1, **c**) (см. 10.3.4.4).

Элементы `Invocable` (модели бизнес-знаний `Business Knowledge Models` или службы принятия решений `Decision Services`) вызываются с использованием того же синтаксиса, что и другие функции (правило грамматики № 38). Элемент `Invocable` тождественен функции FEEL, параметры которой являются входными данными для `Invocable` (см. 10.4)

10.3.2.13.3 Функции, определяемые извне

Функции FEEL, определяемые извне, имеют следующий вид:

function(X_1, … X_n) external mapping-information

Информация об отображении данных – это контекст, который ДОЛЖЕН быть представлен одним из следующих способов:

{

java: {class: class-name, method signature: method-signature}

}

или

{

pmml: {document: IRI, model: model-name}

}

Значение функции, определяемой извне, является элементом семантической области, который обозначается как **function(argument list: [**X_1, … X_n**], external: mapping-information)**.

Если информация об отображении данных указана в форме *java*, то внешняя функция должна быть доступна как метод класса Java. *class-name* ДОЛЖНО быть строковым именем класса Java в пути к классам. Конфигурация Classpath определена в соответствии с реализацией. *method-signature* ДОЛЖНА быть строкой, состоящей из имени публичного статического метода в именованном классе, за которым следует список аргументов, содержащий только имена типов аргументов Java. Информацию типа аргумента СЛЕДУЕТ использовать для распознавания перегруженных методов; ее также МОЖНО использовать для обнаружения ошибок вне основной области до выполнения.

Если информация об отображении данных указана в форме *pmml*, то внешняя функция должна быть доступна как модель PMML. *IRI* ДОЛЖЕН быть идентификатором ресурса для документа PMML. Имя модели *model-name* необязательно. Если *model-name* указано, оно ДОЛЖНО быть именем модели в документе, на который ссылается *IRI*. Если *model-name* не указано, внешняя функция ДОЛЖНА быть первой моделью в документе.

Когда вызывается определенная извне функция, фактические значения аргументов и значение результата преобразуются, когда это возможно, с использованием таблицы сопоставления типов для Java или PMML [см. таблицу 47]. Если преобразование невозможно, то значение **null** заменяется. Если результат не может быть получен, например, генерируется исключение, значение результата вызова будет **null**. Если тип функции, определенной извне – PMML, а результатом вызова PMML является один выход предиктора, то результатом функции, определенной извне будет выходное значение одного предиктора.

Передача значений параметров внешнему методу или модели требует знания ожидаемых типов параметров. Чтобы получить эту информацию в Java используется отражение. В PMML эту информацию можно получить из элементов схемы сбора и словаря данных, связанных с независимыми переменными выбранной модели.

Обратите внимание, что **DMN** не полностью определяет семантику модели принятия решений, которая использует внешние функции. НЕЖЕЛАТЕЛЬНО наличие побочных эффектов у функций, определяемых извне. ЖЕЛАТЕЛЬНО, чтобы функции, определяемые извне, были детерминированными.

10.3.2.13.4 Имя функции

Чтобы назвать функцию, ее следует определить, как запись контекста. Например,

```
{
    isPositive : function(x) x > 0,
    isNotNegative : function(x)
    isPositive(x+1), result:
    isNotNegative(0)
}
```

10.3.2.13.5 Позиционные и именованные параметры

Вызов любой функции FEEL (встроенной, определяемой пользователем или определяемой извне) может использовать позиционные или именованные параметры. В первом случае все параметры ДОЛЖНЫ передаваться. Во втором неизменяемые параметры привязаны к **null**.

10.3.2.14 Выражение цикла for

Выражение цикла for перебирает списки элементов или диапазоны чисел. Общий синтаксис:

for i_1 *in* ic_1 *[,* i_2 *in* ic_2 *[, ...]] return e*

где:

- ic_1, ic_2, …, ic_n являются *контекстами итерации*
- i_1, i_2, …, i_n – это переменные, связанные с каждым элементом в *контексте итерации*
- *e* – это возвращаемое выражение

Контекстом итерации может быть либо выражение, которое возвращает список элементов, либо два выражения, которые возвращают целые числа, связанные знаком “..”. Примеры допустимых контекстов итерации:

- [1, 2, 3]
- список
- 1..10
- 50..40
- x..x+10

Выражение цикла for итерирует каждый элемент в контексте итерации, связывая элемент с соответствующей переменной i_n и оценивая выражение *e* в этой области.

Когда контекст итерации представляет собой диапазон чисел, *выражение цикла for* будет последовательно перебирать весь диапазон, увеличивая или уменьшая значение i_n на 1, в зависимости от того, является ли диапазон восходящим (когда результирующее целое число из первого выражения меньше второго) или нисходящим (когда полученное целое число из первого выражения больше второго).

Результатом *выражения цикла for* является список, содержащий результат вычисления выражения *e* для каждой отдельной итерации по порядку.

Выражение *e* также может ссылаться на неявно определенную переменную, называемую «частичной», которая представляет собой список, содержащий все результаты предыдущих итераций выражения. Это неизменяемая переменная. Например, для вычисления факториального списка чисел от 0 до N, где N - неотрицательное целое число, можно записать следующее выражение:

*for i in 0..N return if i = 0 then 1 else i * partial[-1]*

Когда несколько итерационных контекстов определены в одном и том же *выражении цикла for*, результирующая итерация является перекрестным произведением элементов контекстов итерации. Порядок итераций - от внутреннего контекста итерации ко внешнему контексту итерации.

Например, результат *выражения цикла for*:

for i in [i_1,i_2], j in [j_1,j_2] return e = ***[r_1, r_2, r_3, r_4]***

Где:

r_1 = FEEL(e, { i: i_1, j: j_1, partial:[], … })

r_2 = FEEL(e, { i: i_1, j: j_2, partial:[r_1], …)

r_3 = FEEL(e, { i: i_2, j: j_2, partial:[r_1,r_2], … })

r_4 = FEEL(e, { i: i_2, j: j_2, partial:[r_1,r_2,r_3], … })

10.3.2.15 Семантическое отображение

Значение каждого основного правила грамматики объясняется ниже путем сопоставления синтаксиса со значением в семантической области. Значение может зависеть от определенных входных значений, которые сами были отображены в семантическую область. На входные значения могут распространяться дополнительные ограничения. Входная область(области) может (могут) быть подмножеством семантической области. Входы за пределами области приводят к нулевому значению (**null**), за исключением тех случаев, когда можно применить неявное преобразование *из одноэлементного списка* (10.3.2.9.4).

Таблица 48: Семантика функций FEEL

Правило грамматики	Синтаксис FEEL	Отображение в области
55	*function($n_1, \ldots n_N$) e*	**function(argument list: [$n_1, \ldots n_N$], body: *e*, scope: s)**
55	*function($n_1, \ldots n_N$) external e*	**function(argument list: [$n_1, \ldots n_N$], external: e)**

См. 10.3.2.7 «Диапазоны»

Таблица 49: Семантика других выражений FEEL

Правило грамматики	Синтаксис FEEL	Отображение в области
44	*for i_1 in ic_1, i_2 in ic_2, … return e*	**[FEEL(*e*, s'), FEEL(*e*, s''), …]**
45	*if e_1 then e_2 else e_3*	**if FEEL(e_1) is true then FEEL(e_2) else FEEL(e_3)**
46	*some n_1 in e_1, n_2 in e_2, … satisfies e*	**FEEL(*e*, s') or FEEL(*e*, s'') or …**
46	*every n1 in e1, n2 in e2, … satisfies e*	**false or FEEL(*e*, s') and FEEL(*e*, s'') and …**
47	*e_1 or e_2 or …*	**true and FEEL(e_1) or FEEL(e_2) or** …
48	*e_1 and e_2 and …*	**FEEL(e_1) and FEEL(e_2) and** …
49.a	*e = null*	**FEEL(*e*) is null**
49.a	*null = e*	**FEEL(*e*) is null**
49.a	*e != null*	**FEEL(*e*) is not null**
49.a	*null != e*	**FEEL(e) is not null**

Следует отметить, что мы используем полужирный шрифт для обозначения контекстов, списков, конъюнкций, дизъюнкций, условных выражений, истинных (true), ложных (false) и нулевых (null) значений в области FEEL.

Смысл конъюнкции **a и b** и дизъюнкции **a или b** определяется трехуровневой логикой. Поскольку это полные функции, входные данные могут быть истинными (**true**), ложными (**false**) или иным (**otherwise**) (что означает любой элемент **D**, отличный от **true** или **false**).

Условное выражение **if a then b else c**, равно **b**, если **a – true**, в противном случае оно равно **c**.

s'– это область **s** со специальным первым контекстом, содержащим ключи n_1, n_2 и т. д., привязанные к первому элементу декартова произведения FEEL(**e_1**) **x** FEEL(**e_2**) **x …;**

s'' – это **s** со специальным первым контекстом, содержащим ключи, привязанные ко второму элементу декартового произведения и т. д.

Когда декартово произведение пусто, квантификатор *some ... satisfies* возвращает значение **false**, а квантификатор *every ... satisfies* возвращает значение **true**.

Таблица 50: Семантика конъюнкций и дизъюнкций

a	**b**	**a and b**	**a or b**
true	true	true	true
true	false	false	true
true	otherwise	null	true
false	true	false	true
false	false	false	false
false	otherwise	false	null
otherwise	true	null	true
otherwise	false	false	null
otherwise	otherwise	null	null

Отрицание выполняется с использованием встроенной функции **not**. Трёхуровневая логика показана в таблице 51.

Таблица 51: Семантика отрицания

a	**not(a)**
true	false
false	true
otherwise	null

Равенство и неравенство сопоставляются с несколькими тестами, специфичными для определённого вида или типа данных, как показано в таблицах 52, 53 и 54. По определению FEEL(e_1 *!=* e_2) is FEEL(*not*(e_1=e_2)). Другие операторы сравнения определяются только для типов данных, перечисленных в таблице 54. Обратите внимание, что в таблице 54 приводится определение только для оператора ‘<’. Описание оператора ‘>’ аналогично описанию '<' и для краткости опущено; e_1<=e_2 определяется как e_1<e_2 *или* e_1=e_2.

Таблица 52: Общая семантика равенства и неравенства

Правило грамматики	**Синтаксис FEEL**	**Входная область**	**Результат**
49.a	*e1 = e2*	**e1 и e2 должны быть одинакового вида / иметь один тип данных, например оба должны являться числами, строками и т. д.**	*См. ниже*
49.a	$e_1 < e_2$	**e_1** и **e_2** должны быть одинакового вида / иметь один тип данных, например оба должны являться числами, строками *и т. д.*	*См. ниже*

Таблица 53: Семантика равенства

вид/тип данных	***e_1 = e_2***
list	списки должны быть одинаковой длины N и **e_1[i] = e_2[i]** для $1 \leq$ **i** $\leq$ N.
context	контексты должны иметь одинаковый набор ключей K и **e_1.k = e_2.k** для каждого значения k в K.
range	диапазоны должны указывать одну и ту же конечную точку (точки) и один и тот же оператор сравнения или флаг включенности конечной точки
function	внутренние функции должны иметь одинаковые параметры, тело и область применимости. Функции, определенные из вне, должны иметь одинаковые параметры и внешнее отображение информации о данных.
number	**value(e_1) = value(e_2)**. Значение определяется в подразделе 10.3.2.3.1 «number». Точность не учитывается.
string	последовательность символов **e_1** совпадает с **e_2**.
date	**value(e_1) = value(e_2).** Значение определяется в подразделе 10.3.2.3.5.
date and time	**value(e_1) = value(e_2).** Значение определяется в подразделе 10.3.2.3.6.
time	**value(e_1) = value(e_2).** Значение определяется в подразделе 10.3.2.3.4 .
days and time duration	**value(e_1) = value(e_2)**. Значение определяется в подразделе 10.3.2.3.7.
years and months duration	**value(e_1) = value(e_2)**. Значение определяется в подразделе 10.3.2.3.8 .
boolean	**e_1** и **e_2** должны быть оба истинными (**true)** или оба ложными (**false)**.

Таблица 54: Семантика неравенства

datatype	$e_1 < e_2$
number	**value(e_1) < value(e_2)**. **value** определяется в подразделе 10.3.2.3.1 «number». Точность не учитывается.
string	последовательность символов $\mathbf{e_1}$ лексикографически меньше последовательности символов $\mathbf{e_2}$. То есть, последовательности заполняются до достижения одинаковой длины, если необходимо, при помощи символов *\u0*, лишенных общих символов префикса, а затем сравнивается первый символ в каждой последовательности.
date	e1 <e2, если значение года e1 <значение года e2 e1 <e2, если значения года равны, а значение месяца e1 <значение месяца e2. e1 <e2, если значения года и месяца равны, а значение дня e1 <значение дня e2.
date and time	**$value_{dt}(e_1) < value_{dt}(e_2)$**. **$value_{dt}$** определяется в подразделе 10.3.2.3.5 «date». Если один вход имеет нулевое смещение временной зоны, этот вход использует смещение часового пояса другого входа.
time	**$value_t(e_1) < value_t(e_2)$**. **$value_t$** определяется в подразделе 10.3.2.3.4 «time». Если один вход имеет нулевое смещение временной зоны, этот вход использует смещение часового пояса другого входа.
days and time duration	**$value_{dtd}(e_1) < value_{dtd}(e_2)$**. **$value_{dtd}$** определяется в подразделе 10.3.2.3.7 «days and time duration».
years and months duration	**$value_{ymd}(e_1) < value_{ymd}(e_2)$**. **$value_{ymd}$** определяется в подразделе 10.3.2.3.8 «years and months duration».

FEEL позволяет использовать синтаксический сахар для сравнения. Обратите внимание, что правила грамматики (раздел 10.3.1.2 «Правила грамматики») используются в ячейках условия таблицы решений. Синтаксис таблиц принятия решений приводится в таблице 55.

Таблица 55: Семантика синтаксиса таблицы решений

Правило грамматики	**Синтаксис FEEL**	**Эквивалентный синтаксис FEEL**	**применимость**
49.b	e_1 *between* e_2 *and* e_3	$e_1 >= e_2$ *and* $e_1 <= e_3$	
49.c	e_1 *in* $[e_2,e_3,\ldots]$	$e_1 = e_2$ *or* $e_1 = e_3$ *or*…	e_2 и e_3 являются конечными точками
49.c	e_1 *in* $[e_2,e_3,\ldots]$	e_1 *in* e_2 *or* e_1 *in* e_3 *or*…	e_2 и e_3 являются диапазонами
49.c	e_1 *in* $<=e_2$	$e_1 <= e_2$	
49.c	e_1 *in* $<e_2$	$e_1 < e_2$	
49.c	e_1 *in* $>=e_2$	$e_1 >= e_2$	

49.c	e_1 *in* <e_2	$e_1 < e_2$	
49.c	e_1 *in* (e_2..e_3)	$e_1 > e_2$ *and* $e_1 < e_3$	
49.c	e_1 *in* (e_2..e_3]	$e_1 > e_2$ *and* $e_1 <= e_3$	
49.c	e_1 *in* [e_2..e_3)	$e_1 >= e_2$ *and* $e_1 < e_3$	
49.c	e_1 *in* [e_2..e_3]	$e_1 >= e_2$ *and* $e_1 <= e_3$	
49.c	*e1 in e2*	*e1= e2*	e2 - квалифицированное имя, которое не оценивается как список
49.c	*e1 in e2*	*list contains(e2, e1)*	e1 - это простое значение, которое не является списком, а e2 - это квалифицированное имя, которое оценивается как список
49.c	*e1 in e2*	*{? : e1, r : e2}.r*	e2 - логическое выражение, которое использует специальную переменную "?"

Сложение и вычитание определены в таблице 56 и 57. Следует отметить, что, если входные значения не относятся к перечисленным типам, результат будет **null**.

Таблица 56: Общая семантика сложения и вычитания

Правило граммат ики	**FEEL**	**Входная область и результат**
19	$e_1 + e_2$	*См. ниже*
20	$e_1 - e_2$	*См. ниже*

Таблица 57: Специфическая семантика сложения и вычитания

type(e_1)	**type(e_2)**	***$e_1 + e_2$, $e_1 - e_2$***	**тип результата**
number	number	Допустим, что $\mathbf{e_1=(p_1,s_1)}$ и $\mathbf{e_2=(p_2,s_2)}$ согласно определению, приводимому в подразделе 10.3.2.3.1 «number». Если значение $\mathbf{value(p_1,s_1) +/- value(p_2,s_2)}$ требует шкалы вне диапазона допустимых значений, результат будет равен **null**. В других случаях результат будет равен **(p,s)**, где: • $\mathbf{value(p,s) = value(p_1,s_1) +/- value(p_2,s_2) + \varepsilon}$ • $\mathbf{s \leq max(s_1,s_2)}$ • **s** максимимзируется, поскольку длина **p** не может превышать 34 разряда. • **ε** – возможная ошибка округления.	number
date and time	date and time	Сложение не определено. Вычитание определяется как $\mathbf{value_{dtd\text{-}1}(value_{dt}(e_1)\text{-}value_{dt}(e_2))}$, где $\mathbf{value_{dt}}$ определяется в подразделе 10.3.2.3.5 «date» и $\mathbf{value_{dtd\text{-}1}}$ определяется в подразделе 10.3.2.3.7 «days and time duration». Если для какого-то из значений указан тип date, оно неявно преобразуется в дату и время UTC («00:00:00»), как определено в подразделе 10.3.2.3.6. Вычитание требует, чтобы оба значения имели часовой пояс или оба не имели часового пояса. Вычитание не определено для случая, когда только у одного из значений есть часовой пояс.	days and time duration
time	time	Сложение не определено. Вычитание определяется как $\mathbf{value_{dtd\text{-}1}(value_t(e_1)\text{-}value_t(e_2))}$, где $\mathbf{value_t}$ определено в подразделе 10.3.2.3.4,а $\mathbf{value_{dtd}^{-1}}$ определяется в подразделе 10.3.2.3.7 «days and time duration».	days and time duration
years and months duration	years and months duration	$\mathbf{value_{ymd}^{-1}(value_{ymd}(e_1) +/- value_{ymd}(e_2))}$ где $\mathbf{value_{ymd}}$ и $\mathbf{value_{ymd}^{-1}}$ определяется в подразделе 10.3.2.3.8 «years and months duration».	years and months duration
days and time duration	days and time duration	$\mathbf{value_{dtd\ -1}(value_{dtd}(e_1) +/- value_{dtd}(e_2))}$ где $\mathbf{value_{dtd}}$ и $\mathbf{value_{dtd\ -1}}$ определяется в подразделе 10.3.2.3.7 «days and time duration».	days and time duration

date and time	years and months duration	date and time (date(**e**$_1$.year +/– **e**$_2$.years + floor((**e**$_1$.month +/– **e**$_2$.months)/12), **e**$_1$.month +/– **e**$_2$.months – floor((**e**$_1$.month +/– **e**$_2$.months)/12) * 12, **e**$_1$.day), time(**e**$_1$)), где именованные свойства определены в таблице 53 ниже, а функции даты, даты и времени, времени и округления в меньшую сторону определены в разделе 10.3.4. Значения, **value**$_{dt}$ и **value**$_{dt}^{-1}$ определены в подразделе 10.3.2.3.5 «date», а **value**$_{ymd}$ – в подразделе 10.3.2.3.8 «years and months duration».	date and time
years and months duration	date and time	Вычитание не определено. Сложение является коммутативным и определяется предыдущим правилом.	date and time
date and time	days and time duration	**value**$_{dt}^{-1}$**(value**$_{dt}$**(e**$_1$**) +/- value**$_{dtd}$**(e**$_2$**)),** где значения **value**$_{dt}$ и **value**$_{dt}^{-1}$ определены в подразделе 10.3.2.3.5, а значение **value**$_{dtd}$ определено в подразделе 10.3.2.3.7 «days and time duration».	date and time
days and time duration	date and time	Вычитание не определено. Сложение является коммутативным и определяется предыдущим правилом.	date and time
time	days and time duration	**value**$_t^{-1}$**(value**$_t$**(e**$_1$**) +/- value**$_{dtd}$**(e**$_2$**)),** где значения **value**$_t^{-1}$ и **value**$_t$ определено в подразделе 10.3.2.3.4 , а значение **value**$_{dtd}$ определено в подразделе 10.3.2.3.7 «days and time duration».	time
days and time duration	time	Вычитание не определено. Сложение является коммутативным и определяется предыдущим правилом.	time
String	string	Вычитание не определено. Сложение объединяет строки. Результатом является строка, содержащая последовательность символов в e1, за которой следует последовательность символов в e2.	string
date	years and months duration	**date(e**$_1$**.year +/– e**$_2$**.years + floor((e**$_1$**.month +/– e**$_2$**.months)/12), e**$_1$**.month +/– e**$_2$**.months – floor((e**$_1$**.month +/– e**$_2$**.months)/12) * 12, e**$_1$**.day)**, где именованные свойства определены в таблице 65 ниже, а функции date и floor определены в подразделе 10.3.4.	date
years and months duration	date	Вычитание не определено. Дополнение коммутативно и определяется предыдущим правилом.	date
date	days and time duration	date(value$_{dt-1}$ (value$_{dt}$(e$_1$) +/- value$_{dtd}$(e$_2$))), где value$_{dt}$ и value $_{dt-1}$ определены в 10.3.2.3.5 и value$_{dtd}$ определено в 10.3.2.3.7	date
days and time duration	date	Вычитание не определено. Дополнение коммутативно и определяется предыдущим правилом.	date

Умножение и деление определяются в таблицах 58 и 59. Стоит отметить, что, если входные значения не относятся к перечисленным типам, результат равен **null**.

Таблица 58: Общая семантика умножения и деления

Правило грамматики	FEEL	Входная область и результат
21	$e_1 * e_2$	*См. ниже*
22	e_1 / e_2	*См. ниже*

Таблица 59: Специфическая семантика умножения и деления

type(e_1)	type(e_2)	$e_1 * e_2$	e_1 / e_2	Тип результата
number $e_1=(p_1,s_1)$	number $e_2=(p_2,s_2)$	Если **value(p_1,s_1) * value(p_2,s_2)** требует шкалы вне диапазона допустимых значений, результат будет равен **null**. В других случаях результат будет равен **(p,s)**, где • **value(p,s) = value(p_1,s_1) * value(p_2,s_2) + ε** • **s ≤ s_1+s_2** • **s** максимимзируется, поскольку длина **p** не может превышать 34 разряда • **ε** возможная ошибка округления	Если **value(p_2,s_2)=0** или **value(p_1,s_1) / value(p_2,s_2)** требует шкалы вне диапазона допустимых значений, результат будет равен **null**. В других случаях результат будет равен **(p,s)**, где • **value(p,s) = value(p_1,s_1) / value(p_2,s_2) + ε** • **s ≤ s_1-s_2** • **s** максимимзируется, поскольку длина **p** не может превышать 34 разряда • **ε** возможная ошибка округления	number
years and months duration	number	**$value_{ymd}^{-1}$(value$_{ymd}$(e_1) * value(e_2)),** где **value$_{ymd}$** и **value$_{ymd}$** определяются в подразделе 10.3.2.3.7 «days and time duration».	Если **value(e_2)=0**, результат будет **null**. В противном случае – **$value_{ymd}^{-1}$(value$_{ymd}$(e_1) / value(e_2)),** где **value$_{ymd}$** и **value$_{ymd}^{-1}$** определяются в подразделе 10.3.2.3.8 «years and months duration».	years and months duration
number	years and months duration	Как в примере, приведенном выше, однако значения e1 и e2 меняются местами	Не разрешено	years and months duration
years and months duration	number	Не допускается	Если **valuey$_{ymd}$(e_2)=0**, результат будет **null**. В противном случае – **value$_{ymd}$(e_1) / value(e_2),** где значение **value$_{ymd}$** определяется в подразделе 10.3.2.3.8 «years and months duration»	number
days and time duration	number	**$value_{dtd}^{-1}$(value$_{dtd}$(e_1) * value(e_2)),** где значения **value$_{dtd}$** и **$value_{dtd}^{-1}$** определяются в подразделе 10.3.2.3.7 «days and time duration».	Если **value(e_2)=0**, результат будет **null**. В противном случае результат будет **value$_{dtd}^{-1}$(value$_{dtd}$(e_1) * value(e_2)),** где значения **value$_{dtd}$** и **$value_{dtd}^{-1}$** определяются в подразделе 10.3.2.3.7 «days and time duration».	days and time duration

number	days and time duration	Как в примере, приведенном выше, однако значения e1 и e2 меняются местами	Не допускается	days and time duration
days and time duration	number	Не разрешено	Если $\mathbf{value_{dtd}(e_2)=0}$, результат будет **null**. В противном случае результат будет $\mathbf{value_{dtd}(e_1)/\ value(e_2)}$, где значение $\mathbf{value_{dtd}}$ определяется в подразделе 10.3.2.3.7 «days and time duration».	number

Таблица 60: Семантика возведения в степень

Правило грамматики	Синтаксис FEEL	Входная область	Результат
23	e_1 ** e_2	$\mathbf{type(e_1)}$ является числом. $\mathbf{value(e_2)}$ является числом в диапазоне [-999,999,999..999,999,999].	Если $\mathbf{value(e_1)_{value(e2)}}$ требует шкалы вне диапазона допустимых значений, результат будет равен **null**. В других случаях результат будет равен **(p,s),** где • $\mathbf{value(p,s)= value(e_1)_{value(e2)} + \varepsilon}$ • длина **p** не может превышать 34 разряда • **ε** - ошибка округления

Проверка типов определяется в таблице 61. Следует отметить, что *type* не отображается в области, а **null** является единственным значением в типе Null (см. 10.3.2.1).

Перед оценкой *экземпляра* оператора оба операнда сопоставляются с решеткой типов **L** (см. 10.3.2.9).

Таблица 61: Семантика проверки типов

Правило грамматики	Синтаксис FEEL	Отображение в домене	Примеры
51	***e1 instance of e2***	**Если *e2* нельзя сопоставить с узлом решетки L, то результат будет null** Если **e1** равно **null** и **type(e2)** – *Null*, то тогда результат (**true)** Если **type(e1)** соответствует **type(e2)** (см. раздел 10.3.2.9) и **e1** не **null**, то результат истина (**true).** В противном случае результат **false.**	*Экземпляр списка [123] – true* *Экземпляр строки "abc" – true* *Экземпляр строки 123 – false* *Экземпляр списка 123 – null* *т.к. для типа list требуется указать параметры (см. правило № 54).*

Отрицательные числа определяются в таблице 62.

Таблица 62: Семантика отрицательных чисел

Правило грамматики	Синтаксис FEEL	Эквивалентный синтаксис FEEL
24	*-e*	*0-e*

Вызов определяется в таблице 63. В вызове могут использоваться позиционные и именованные аргументы. В первом случае, передаваться должны все аргументы. Во втором – непередаваемые аргументы привязываются к нулевому значению (**null**). Обратите внимание, что **e** может быть функцией, определяемой пользователем, а также внешней или встроенной функцией. Аргументы подлежат неявными преобразованиями (10.3.2.9.4). Если типы аргументов до или после преобразования не соответствуют типам параметров, результат вызова будет нулевым (**null**).

Таблица 63: Семантика вызовов

Правило грамматики	FEEL	Отображение в домене	Применимость
38, 39, 42	*e*(e_1,..)	**e**(e_1,…)	**e** – это функция с подходящей арностью и соответствующими типами параметров
38, 39, 40, 41	*e*(n_1:e_1,…)	**e**(n_1:e_1,…)	**e** – это функция с подходящими именами и соответствующими типами параметров

Свойства определяются в таблице 64 и 65. Если **type(*e*)** – это date and time, time, или duration, а **name** - имя свойства, то значение указано в таблице 65 и таблице 66. Например, FEEL(*date and time("2012-03-07Z").year*) = **2012**.

Таблица 64: Общая семантика свойств

Правило грамматики	FEEL	Отображение в домене	Применимость
18	*e.name*	**e."name"**	**type(e)** является контекстом
18	*e.name*	*См. ниже*	**type(e)** является date/time/duration

Таблица 65: Список свойств для каждого типа

type(*e*)	***e* . name**	**name =**
date	результатом является именованный компонент объекта date **e**. Действительные имена показаны справа.	year, month, day, weekday
date and time	результатом является именованный компонент объекта date and time **e**. Действительные имена показаны справа.	year, month, day, weekday, hour, minute, second, time offset, timezone
time	результатом является именованный компонент объекта time **e**. Действительные имена показаны справа.	hour, minute, second, time offset, timezone

years and months duration	результатом является именованный компонент объекта years and months duration **e**. Действительные имена показаны справа.	years, months
days and time duration	результатом является именованный компонент объекта days and time duration **e**. Действительные имена показаны справа.	days, hours, minutes, seconds
range	результатом является именованный компонент объекта range **e**. Действительные имена показаны справа.	start, end, start included, end included

Таблица 66: Специфическая семантика свойств даты, времени и продолжительности

name	***type*(name)**	**Описание**
year	number	Номер года в виде целого числа в интервале [-999,999,999 .. 999999999]
month	number	Номер месяца в виде целого числа в интервале [1..12], где 1 – это январь и 12 – это декабрь
day	number	День месяца как целое число в интервале [1..31]
weekday	number	День недели как целое число в интервале [1..7], где 1 – это понедельник и 7 – воскресенье (соответствует определению ISO 8601)
hour	number	Час дня как целое число в интервале [0..23]
minute	number	Минута часа как целое число в интервале [0..59]
second	number	Секунда в виде десятичной дроби в интервале [0..60)
time offset	days and time duration	Смещение времени, соответствующее часовому поясу, который представляет значение date или date and time. Продолжительность временного смещения должна быть в интервале **[duration("-PT14H")..duration("PT14H")]**, согласно XML Schema. Часть 2: dateTime datatype. Свойство **time offset** возвращает ноль, если объект не имеет установленного временного смещения.
timezone	string	Идентификатор часового пояса, определенный в базе данных часовых поясов IANA. Свойство **timezone** возвращает ноль, если для объекта не определен часовой пояс IANA.
years	number	Нормализованный компонент лет в значении years and months duration представляет собой целое число. Это свойство возвращает ноль, если вызывается для значения days and time duration.
months	number	Нормализованный компонент месяцев в значении years and months duration. Поскольку значение нормализовано, это свойство должно возвращать целое число в интервале [0..11]. Свойство возвращает ноль, если вызывается для значения days and time duration.
days	number	Нормализованный компонент дней в значении days and time duration в виде целого числа. Свойство возвращает ноль, если вызывается для значения years and months duration.
hours	number	Нормализованный компонент часов в значении days and time duration. Поскольку значение нормализовано, это свойство должно возвращать целое число в интервале [0..23]. Свойство возвращает ноль, если вызывается для значения years and months duration.

minutes	number	Нормализованный компонент минут в значении days and time duration. Поскольку значение нормализовано, это свойство должно возвращать целое число в интервале [0..59]. Свойство возвращает ноль, если вызывается для значения years and months duration.
seconds	number	Нормализованный компонент минут в значении days and time duration. Поскольку значение нормализовано, это свойство должно возвращать число с десятичным разделителем в интервале [0..60]. Свойство возвращает ноль, если вызывается для значения years and months duration.

Таблица 67: Специфическая семантика свойств диапазонов

name	***type*(name)**	**описание**
start	Тип начальной точки диапазона	Начальная точка диапазона
end	Тип конечной точки диапазона	Конечная точка диапазона
start included	boolean	Начальная точка включена в диапазон
end included	boolean	Конечная точка включена в диапазон

Списки определяются в таблице 68.

Таблица 68: Семантика списков

Правило грамматики	**Синтаксис FEEL**	**Отображение в домене (область s)**	**Применимость**
54	$e_1[e_2]$	$\mathbf{e_1[e_2]}$	$\mathbf{e_1}$ является списком и $\mathbf{e_2}$ является целым числом (0 номер шкалы)
54	$e_1[e_2]$	$\mathbf{e_1}$	$\mathbf{e_1}$ не является списком и не **null,** а **value(**$\mathbf{e_2}$**)** = **1**
54	$e_1[e_2]$	список элементов **e** таков, что **i** принадлежит **e**, если **i** находится в $\mathbf{e_1}$, а FEEL(e_2, **s'**) имеет значение **true**, где **s'** – область s со специальным первым контекстом, содержащим запись контекста ("**item**", **i**) и, если **i** является контекстом. Специальный контекст также содержит все записи контекста **i**.	$\mathbf{e_1}$ является списком, и **type(**FEEL(e_2, **s'**)**)** является булевым значением

54	$e_1[e_2]$	**[e₁]** если FEEL(e_2, **s'**) имеет значение **true**, где **s'** – это область **s** со специальным первым контекстом, содержащим запись контекста ("**item**", **e₁**), и если **e₁** является контекстом, специальный контекст также содержит все элементы контекста **e₁**. Else **[]**.	**e₁** не является списком и не **null,** а **type(**FEEL(e_2, **s'**)**)** является булевым значением

Контексты определены в таблице 69.

Таблица 69: Семантика контекста

Правило грамматики	**Синтаксис FEEL**	**Отображение в домене (область s)**
57	$\{ n_1 : e_1, n_2 : e_2, \ldots \}$	**{** "**n₁**"**:** FEEL(e_1, **s₁**)**,** "**n2**"**:** FEEL(e_2, **s₂**)**, …},** при этом **sᵢ** – являются всеми **s** со специальным первым контекстом **cᵢ**, содержащим подмножество записей контекста результата. Если **cᵢ** содержит запись для **nⱼ**, то **cⱼ** не содержит записи для **nᵢ**.
	$\{ "n_1" : e_1, "n_2" : e_2, \ldots \}$	
54	$[e_1, e_2, \ldots]$	**[** FEEL(e_1)**,** FEEL(e_2)**, …]**

10.3.2.16 Обработка ошибок

Когда встроенная функция обрабатывает вход, который находится за пределами определенного домена, ЖЕЛАТЕЛЬНО, чтобы функция сообщала или записывала диагностическую информацию, если это необходимо, при этом функция ДОЛЖНА возвращать **null**.

10.3.3 Данные XML

FEEL поддерживает XML-данные в контексте FEEL путем сопоставления данных XML с семантическим доменом FEEL. Допустим, что XE(*e*, **p**) является функцией, преобразующей XML-элемент *e* и родительский контекст FEEL **p** в контексте FEEL, в соответствии с таблицами, приведенными ниже. XE использует другую функцию XV(*v*), которая преображает XML-значение *v* в семантическую область FEEL.

Семантика пространства имен XML не поддерживается сопоставлением. Например, с учетом представлений префикса пространства имен *xmlns:p1="http://example.org/foobar"* и *xmlns:p2="http://example.org/foobar"*, теги *p1:myElement* и *p2:myElement* будут одним и тем же элементом, если используются семантика пространства имён XML; однако они будут являться разными элементами, если используется XML без семантики пространства имён.

10.3.3.1 Семантическое сопоставление для элементов XML (XE)

Таблица 70, *e* – это имя элемента XML, *a* – имя одного из его атрибутов, *c* – дочерний элемент, а *v* – значение. Родительский контекст **p** изначально пуст.

Таблица 70: Семантика элементов XML

XML	**Запись контекста в p**	**Примечание**
<e />	**"e" : null**	пустой элемент → запись со значением **null** в **p**
<q:e />	**"e" : null**	пространства имен игнорируются.
<e>v</e>	**"e":**XV(*v*)	неповторяющийся элемент без атрибутов
<e>v_1</e> <e>v_2</e>	**"e": [** XV(v_1)**,** XV(v_2) **]**	повторяющийся элемент без атрибутов
<e a="v"/> *<c_1>v_1</c_1>* *<c_n>v_2</c_n><c_n>v_3</c_n>* *</e>*	**"e": { "a":** XV(*v*), **"c_1":** XV(v_1), **"cn": [** XV(v_2), XV(v_3)] **}**	Элемент, содержащий атрибуты или дочерние элементы → контекст
<e a="v_1">v_2</e>	**"e": {** "**@a**": XV(v_1), "**$content**": XV($v_2$) **}**	v_2 is содержится в сгенерированной записи $content

Такая запись в столбце **context entry in p**, как **"e" : null** указывает на запись контекста со строковым ключом **"e"** и значением **null**. Записи контекста содержатся в контексте **p**, который соответствует XML-элементу, или самому XML-документу.

При сопоставлении префиксы пространства имен не заменяются на пространства имен IRI. Правила FEEL требуют только того, чтобы ключи внутри контекста были различны, а префиксы пространства имен были достаточными.

10.3.3.2 Семантическое сопоставление для значений XML (XV)

Если XML-документ был проанализирован с использованием XML-схемы, то типы данных некоторых атомарных значений не обязательно являются строками. Таблица 71 определяет, как типизированное значение XML *v* отображается в FEEL.

Таблица 71: Семантика значений XML

Тип *v*	**Семантический домен FEEL**
number	FEEL(*v*)
string	FEEL(*"v"*)
date	FEEL(*date("v")*)
dateTime	FEEL(*date and time("v")*)

time	FEEL(*time("v")*)
duration	FEEL(*duration("v")*)
list, *например "v_1 v_2"*	[XV(v_1), XV(v_2)]
element	XE(*v*)

10.3.3.3 Пример XML

Ниже приводит пример схемы и экземпляра в XML и их эквиваленты в FEEL:

10.3.3.3.1 Схема

```
<xsd:schema
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns="http://www.example.org"
        targetNamespace="http://www.example.org"
        elementFormDefault="qualified">
 <xsd:element name="Context">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="Employee">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="salary" type="xsd:decimal"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="Customer" maxOccurs="unbounded">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="loyalty_level" type="xsd:string"/>
       <xsd:element name="credit_limit" type="xsd:decimal"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

10.3.3.3.2 Экземпляр

```
<Context
      xmlns:tns="http://www.example.org"
      xmlns="http://www.example.org">
 <tns:Employee>
  <tns:salary>13000</tns:salary>
 </tns:Employee>
 <Customer>
  <loyalty_level>gold</loyalty_level>
  <credit_limit>10000</credit_limit>
 </Customer>
```

```
 <Customer>
  <loyalty_level>gold</loyalty_level>
  <credit_limit>20000</credit_limit>
 </Customer>
 <Customer>
  <loyalty_level>silver</loyalty_level>
  <credit_limit>5000</credit_limit>
 </Customer>
</Context>
```

10.3.3.3.3 Эквивалент в FEEL – табличный контекст

Контекст		
Employee	salary	13000
Заказчик	loyalty_level	credit_limit
	золото	10000
	золото	20000
	серебро	5000

Когда проверяется модель решения, ее входные данные, описываемые определением элемента XML Schema (раздел 7.3.2. Метамодель «ItemDefinition»), привязаны к данным экземпляра, отображаемым в домене FEEL. Данные экземпляра могут быть представлены в различных форматах, например, XML. Данные экземпляра можно обозначить как эквивалентный табличный контекст (см. таблицу выше). Логика принятия решения может ссылаться на записи в контексте, используя такие выражения, как *Context.tns$Employee.tns$salary*, со значением 13000.

10.3.4 Встроенные функции

Для обеспечения интероперабельности FEEL включает библиотеку встроенных функций. Синтаксис и семантика встроенных модулей необходимы для совместимой реализации FEEL.

Во всех таблицах в этом разделе надстрочный указатель ссылается на дополнительное ограничение домена, указанное в соответствующей сноске к таблице. Всякий раз, когда параметр находится за пределами домена, результатом встроенного значения является **null**.

10.3.4.1 Функции преобразования

FEEL поддерживает преобразование различных типов данных. Особое значение имеет преобразование строк в даты, время и продолжительность. FEEL не поддерживает буквенное представление даты, времени или продолжительности. Кроме того, отформатированные числа, такие как *1,000.00*, должны преобразовываться из строк с указанием разделителя групп разрядов и десятичного разделителя.

Встроенные функции перечисляются в таблицы 72. В первом столбце указаны имя и параметры. Вопросительный знак (?) обозначает необязательный параметр. Второй столбец определяет домен для параметров. Домен параметра указан как

- тип, например, число, строка;

- any – любой элемент из семантической области, включая **null;**
- not null – любой элемент из семантического домена, исключая **null**;
- date string – строковое значение в лексическом пространстве типа данных date, указанного в XML Schema Part 2 Datatypes;
- time string – строковое значение в лексическом пространстве типа данных time, указанного в XML Schema Part 2 Datatypes, или строковое значение, которое представляет собой расширенную форму локального представления time, согласно ISO 8601, за которым следует символ "@", а затем строковое значение, которое является идентификатором часового пояса в базе данных временных зон IANA (http://www.iana.org/time-zones);
- date time string – строковое значение, состоящее из значения строки даты, как указано выше, за которым опционально следует символ "T", а затем значение строки времени, как указано выше;
- duration string – строковое значение в лексическом пространстве типов xs: dayTimeDuration или xs: yearMonthDuration, заданных в XQuery 1.0 и модели данных XPath 2.0.

Таблица 72: Семантика функции преобразования

Имя(параметры)	**Домен параметров**	**Описание**	**Пример**
date(*from*)	date string	конвертирует *from* в date	*date("2012-12-25") – date("2012-12-24") = duration("P1D")*
date(*from*)	date and time	конвертирует *from* в date (устанавливает компоненты времени на null)	*date(date and time("2012-12-25T11:00:00Z")) = date("2012-12-25")*
date(*year*, *month*, *day*)	*year*, *month*, *day* являются числами	создает значения даты из значений года, месяца, дня	*date(2012, 12, 25) = date("2012-12-25")*
date and time(*date*, *time*)	*date* является date or date time; *time* is a time	создает дату из указанной даты (игнорируя любой компонент времени) и заданного времени	*date and time ("2012-12- 4T23:59:00") = date and time (date("2012-12-24"), time("T23:59:00"))*
date and time(*from*)	date time string	конвертирует *from* в date and time	*date and time("2012-12-24T23:59:00") + duration("PT1M") = date and time("2012-12- 25T00:00:00")*
time(*from*)	time string	конвертирует *from* в time	*time("23:59:00z") + duration("PT2M") = time("00:01:00@Etc/UTC")*
time(*from*)	time, date and time	конвертирует *from* в time (игнорируя компонент даты)	*time(date and time("2012-12-25T11:00:00Z")) = time("11:00:00Z")*
time(*hour*, *minute*, *second*, *offset?*)	*hour*, *minute*, *second*, являются числами, *offset* являются days and time duration, or	создает время из заданных значений компонента	*time("T23:59:00z") = time(23, 59, 0, duration("PT0H"))*

	null		
number(*from*, *grouping separator*, *decimal separator*)	string[1], string, string	конвертирует *from* в number	*number("1 000,0", " ", ",") = number("1,000.0", ",", ".")*
string(*from*)	non-null	конвертирует *from* в string	*string(1.1) = "1.1"* *string(null) = null*
duration(*from*)	duration string	конвертирует *from* в days and time duration или years and months duration	*date and time("2012-12-24T23:59:00") - date and time("2012-12-22T03:45:00") = duration("P2DT20H14M")* *duration("P2Y2M") = duration("P26M")*
years and months duration(*from*, *to*)	Оба являются date and time	возвращает years and months duration между *from* и *to*	*years and months duration(date("2011-12-22"), date("2013-08-24")) = duration("P1Y8M")*

1. *Разделитель групп разрядов* ДОЛЖЕН обозначаться одним из следующих способов: пробел (' '), запятая (','), точка ('.') или **null**.

Десятичный разделитель ДОЛЖЕН обозначаться одним из следующих способов: точка, запятая, null. Десятичный разделитель НЕ ДОЛЖЕН совпадать с разделителем групп разрядов, если только они оба не имеют значение null.

from ДОЛЖНО соответствовать правилу грамматики 37 после удаления всех вхождений разделителя групп разрядов, если таковые имеются, и после изменения десятичного разделителя, если таковой имеется, на точку.

10.3.4.2 Булевы функции

Таблица 73 определяет булевы функции.

Таблица 73: Семантика булевых функций

Имя(параметры)	**Домен параметров**	**Описание**	**Пример**
not(*negand*)	boolean	Логическое отрицание	*not(true) = false* *not(null) = null*

10.3.4.3 Строковые функции

Таблица 74 определяет строковые функции.

Таблица 74: Семантика строковых функций

Имя(параметры)	**Домен параметров**	**Описание**	**Пример**
substring(*string, start position, length?*)	string, number[1]	возвращает *length* или все символы в *string*, начиная со *start position*. 1-я позиция – 1, последняя позиция –1	*substring("foobar",3) = "obar"* *substring("foobar",3,3) = "oba"* *substring("foobar", -2, 1) = "a"* *substring("\U01F40Eab ", 2) = "ab" где "\U01F40Eab " является представлением* 𐎠ab
string length(*string*)	string	Возвращает количество символов (или кодовых точек) в строке *string*	*string length("foo") = 3* *string length("\U01F40Eab") = 3*
upper case(*string*)	string	Возвращает *string* в верхнем регистре	*upper case("aBc4") = "ABC4"*
lower case(*string*)	string	Возвращает *string* в нижнем регистре	*lower case("aBc4") = "abc4"*
substring before (*string, match*)	string, string	Возвращает подстроку *string* до *match* в *string*	*substring before("foobar", "bar")* *= "foo" substring* *before("foobar", "xyz") = ""*
substring after (*string, match*)	string, string	Возвращает подстроку *string* после *match* в *string*	*substring after("foobar", "ob")* *= "ar" substring after("", "a")* *= ""*
replace(*input, pattern, replacement, flags?*)	string[2]	сопоставление и замена шаблонов регулярных выражений	*replace("abcd", "(ab)|(a)", "[1=$1][2=$2]") = "[1=ab][2=]cd"*
contains(*string, match*)	string	Строка *string* содержит *match*?	*contains("foobar", "of") = false*
starts with(*string, match*)	string	Строка *string* начинается с *match*?	*starts with("foobar", "fo") = true*
ends with(*string, match*)	string	Строка *string* заканчивается с *match*?	*ends with("foobar", "r") = true*
matches(*input, pattern, flags?*)	string[2]	Вход *input* соответсвует шаблону регулярного выражения *pattern*?	*matches("foobar", "^fo*b") = true*
split(string, delimiter)	*string* является строкой string, а разделитель – это шбалон pattern2	Разбивает строку *string* на список подстрок, который прерывается при наличии шаблона разделителя *delimiter pattern*.	*split("John Doe", "\\s") = ["John", "Doe"] split("a;b;c;;", ";") = ["a","b","c","",""]*

1. *start position* должна быть отличным от нуля целым числом (0 на шкале) в диапазоне [-L..L], где L – длина строки. *length* должна находиться в диапазоне [1..E], где E - L – *start position* + *1*, если значение *start position* положительное, и – *start position* в противном случае.

2. *pattern*, *replacement*, и *flags* ДОЛЖНЫ соответствовать синтаксису и ограничениям, указанным в разделе 7.6 XQuery 1.0 и XPath 2.0. Функции и операторы. Обратите внимание: там, где XPath указывает результат ошибки, FEEL указывает нулевой результат (null).

10.3.4.4 Функции списка

Таблица 75 определяет функции списка.

Таблица 75: Семантика функций списка

Имя(параметры)	**Домен параметров**	**Описание**	**Пример**
list contains(*list*, *element*)	list, any element of the semantic domain including **null**	Список *list* содержит *element*?	*list contains([1,2,3], 2) = true*
count(*list*)	list	Возвращает размер списка *list*, или ноль, если список *list* пустой	*count([1,2,3]) = 3* *count([]) = 0* *count([1,[2,3]]) = 2*
min(*list*) min($c_1, \ldots, c_N$), $N > 0$ max(*list*) max($c_1, \ldots, c_N$), $N > 0$	non-empy list of comparable items or argument list of one or more comparable items	Возвращает минимальный (максимальный) элемент, или **null** если список *list* пустой	*min([1,2,3]) = 1* *max(1,2,3) = 3* *min(1) = min([1]) = 1* *max([]) = null*
sum(*list*) sum($n_1, \ldots, n_N$), $N > 0$	list of 0 or more numbers or argument list of one or more numbers	возвращает сумму чисел, или **null** если список *list* пустой	*sum([1,2,3]) = 6* *sum(1,2,3) = 6* *sum(1) = 1* *sum([]) = null*
mean(*list*) mean($n_1, \ldots,$ n_N), $N > 0$	non-empty list of numbers or argument list of one or more numbers	Возвращает среднее арифметическое	*mean([1,2,3]) = 2* *mean(1,2,3) = 2* *mean(1) = 1* *mean([]) = null*
all(*list*) all($b_1, \ldots, b_N$), $N > 0$	list of Boolean items or argument list of one or more Boolean items	возвращает *false*, если один из элементов имеет значение *false*, или *true*, если список пустой или значение всех элементов *true*, в других случаях возвращает *null*	*all([false,null,true]) = false* *all(true) = all([true]) = true* *all([]) = true* *all(0) = null*

any(*list*) any($b_1, \ldots, b_N$), $N > 0$	list of Boolean items or argument list of one or more Boolean items	возвращает *true*, если один из элементов имеет значение *true*, или *false*, если список пустой или значение всех элементов *false*, в других случаях возвращает *null*	*any([false,null,true]) = true* *any(false) = false* *any([]) = false* *any(0) = null*
sublist(*list, start position, length?*)	list, number[1], number[2]	Возвращает список *length* или все элементы *list*, начиная с *list[start position]*. 1-я позиция –1, последняя позиция –1	*sublist([4,5,6], 1, 2) = [4,5]*
append(*list, item…*)	list, any element including **null**	Возвращает новый *list* с добавленными элементами *item*	*append([1], 2, 3) = [1,2,3]*
concatenate(*list…*)	list	Возвращает новый *list*, который является конкатенацией аргументов	*concatenate([1,2],[3]) = [1,2,3]*
insert before(*list, position, newItem*)	list, number[1], any element including **null**	Возвращает новый *list* с *newItem*, вставленными на место *position*	*insert before([1,3],1,2) = [2,1,3]*
remove(*list, position*)	list, number[1]	*list* с позицией в *position* удален	*remove([1,2,3], 2) = [1,3]*
reverse(*list*)	list	Отображает *list* в обратном порядке	*reverse([1,2,3]) = [3,2,1]*
index of(*list, match*)	list, any element including **null**	Возвращает *list* позиций, содержащих *match*, в порядке возрастания	*index of([1,2,3,2],2) = [2,4]*
union(*list…*)	list	Соединяет значения, удаляет повторяющиеся значения	*union([1,2],[2,3]) = [1,2,3]*
distinct values(*list*)	list	Удаление повторяющихся значений	*distinct values([1,2,3,2,1] = [1,2,3]*
flatten(*list*)	list	Выполняет сведение вложенных списков	*flatten([[1,2],[[3]], 4]) = [1,2,3,4]*
product(*list*) product($n_1, \ldots, n_n$)	list is a list of numbers. $n_1 \ldots n_n$ are numbers.	Возвращает произведение чисел	*product(2, 3, 4) = 24*
median(*list*) median($n_1, \ldots, n_n$)	list is a list of numbers. $n_1 \ldots n_n$ are numbers.	Возвращает медианный элемент списка чисел. Т.е. после сортировки списка, если список имеет нечетное количество элементов, он возвращает средний элемент. Если список имеет четное количество элементов, возвращает среднее значение двух средних элементов. Если список пуст, возвращает ноль.	*median(8, 2, 5, 3, 4) = 4* *median([6, 1, 2, 3]) = 2.5* *median([ ]) = null* *median([6, 1, 2, 3]) = 2.5* *median([ ]) = null*

stddev(*list*) stddev(n_1, …, n_n)	list is a list of numbers. n_1 … n_n are numbers.	Возвращает пример стандартного отклонения списка чисел. Если список *list* пуст или содержит только один элемент, функция возвращает ноль.	*stddev(2, 4, 7, 5) = 2.0816659994661327352822 97706979931* *stddev([47]) = null* *stddev(47) = null* *stddev([ ]) = null*
mode (*list*) mode (n_1, …, n_n)	list is a list of numbers. n_1 … n_n are numbers.	Возвращает режим списка номеров. Если результат содержит несколько элементов, они возвращаются в порядке возрастания. Если список пуст, возвращается пустой список.	*mode(6, 3, 9, 6, 6) = [6]* *mode([6, 1, 9, 6, 1]) = [1, 6]* *mode([ ]) = [ ]*

1. *position* должно быть отличным от нуля целым числом (0 на шкале) в диапазоне [-L..L], где L – длина списка.
2. *length* должна находится в диапазоне [1..E], где E is L – *start position +1*, если *start position* положительное число, и –*start position* в других случаях.

10.3.4.5 Числовые функции

Таблица 76 определяет числовые функции.

Таблица 76: Семантика числовых функций

Имя(параметры)	**Домен параметров**	**Описание**	**Пример**
decimal(*n*, *scale*)	number, number[1]	возвращает *n* с заданной шкалой *scale*	*decimal(1/3, 2) = .33* *decimal(1.5, 0) = 2* *decimal(2.5, 0) = 2*
floor(*n)*	number	возвращать наибольшее целое число<= *n*	*floor(1.5) = 1* *floor(-1.5) = -2*
ceiling(*n)*	number	возвращать наименьшее целое число>= *n*	*ceiling(1.5) = 2* *ceiling(-1.5) = -1*
abs(*n*)	*n* может быть number, days and time duration или year and month duration.	Возвращает абсолютное значение *n*.	*abs(10) = 10* *abs(-10) = 10* *abs(@"PT5H") = @"PT5H"* *abs(@"-PT5H") = @"PT5H"*
modulo(*dividend*, *divisor*)	*dividend* и *divisor* являются числами, при этом делитель *divisor* не может быть равным 0 (нулю). Возвращает остаток от деления *dividend* на *divisor*. Если *dividend* или *divisor* отрицательные числа, то	Возвращает остаток от деления делимого на делитель.	*modulo(12, 5) = 2* *modulo(-12,5)= 3* *modulo(12,-5)= -3* *modulo(-12,-5)= -2* *modulo(10.1, 4.5)= 1.1* *modulo(-10.1, 4.5)= 3.4*

	результат записывается со знаком *divisor*. Функция `modulo` может быть выражена следующим образом: `modulo(dividend, divisor) = dividend - divisor*floor(dividend/divisor.`		*modulo(10.1, -4.5)= -3.4* *modulo(-10.1, -4.5)= -1.1*
sqrt(*number*)	*number* является числом.	Возвращает квадратный корень заданного числа. Если число отрицательное, то возвращается null.	*sqrt(16) = 4*
log(*number*)	*number* является числом.	Возвращает натуральный логарифм (основание **e**) параметра *number*.	*log(10) = 2.30258509299*
exp(*number*)	*number* является числом.	Возвращает число Эйлера **e**, возведенное в степень *number*.	*exp(5) = 148.413159102577*
odd(*number*)	*number* является числом.	Возвращает true, если число *number* нечетное, false, если оно четное.	*odd(5) = true* *odd(2) = false*
even(*number*)	*number* является числом.	Возвращает true, если число *number* четное, false, если оно нечетное.	*even(5) = false* *even (2) = true*

1. Шкала находится в диапазоне [−6111..6176]

10.3.4.6 Функции даты и времени

Функции даты и времени определяются в таблице 77.

Таблица 77: Семантика функций даты и времени

Имя(параметры)	**Домен параметров**	**Описание**	**Пример**
is(value1, value2)	Оба являются элементами **D**	Возвращает **true**, если оба значения являются одним и тем же элементом в семантическом домене FEEL **D** (см. 10.3.2.2)	*is(date("2012-12-25"), time("23:00:50”)) is* ***false*** *is(date("2012-12-25"), date("2012-12-25")) is* ***true*** *is(time("23:00:50z"), time("23:00:50”)) is* ***false*** *is(time("23:00:50z"), time("23:00:50+00:00”)) is* ***false***

10.3.4.7 Функции диапазона

Функции, представленные в списке ниже, устанавливают связи между одиночными скалярными значениями и

диапазонами таких значений. Все упомянутые функции принимают два аргумента и возвращают True, если связь между аргументами выполняется, в противном случае они возвращают False.

Спецификация функций в значительной степени основана на эквивалентных функциях стандартной версии 1.4 языка HL7 CQL (Clinical Quality Language).

На рисунке ниже наглядно отображены связи, определяемые функциями в этой главе, однако полная семантика функций приводится в таблице 78.

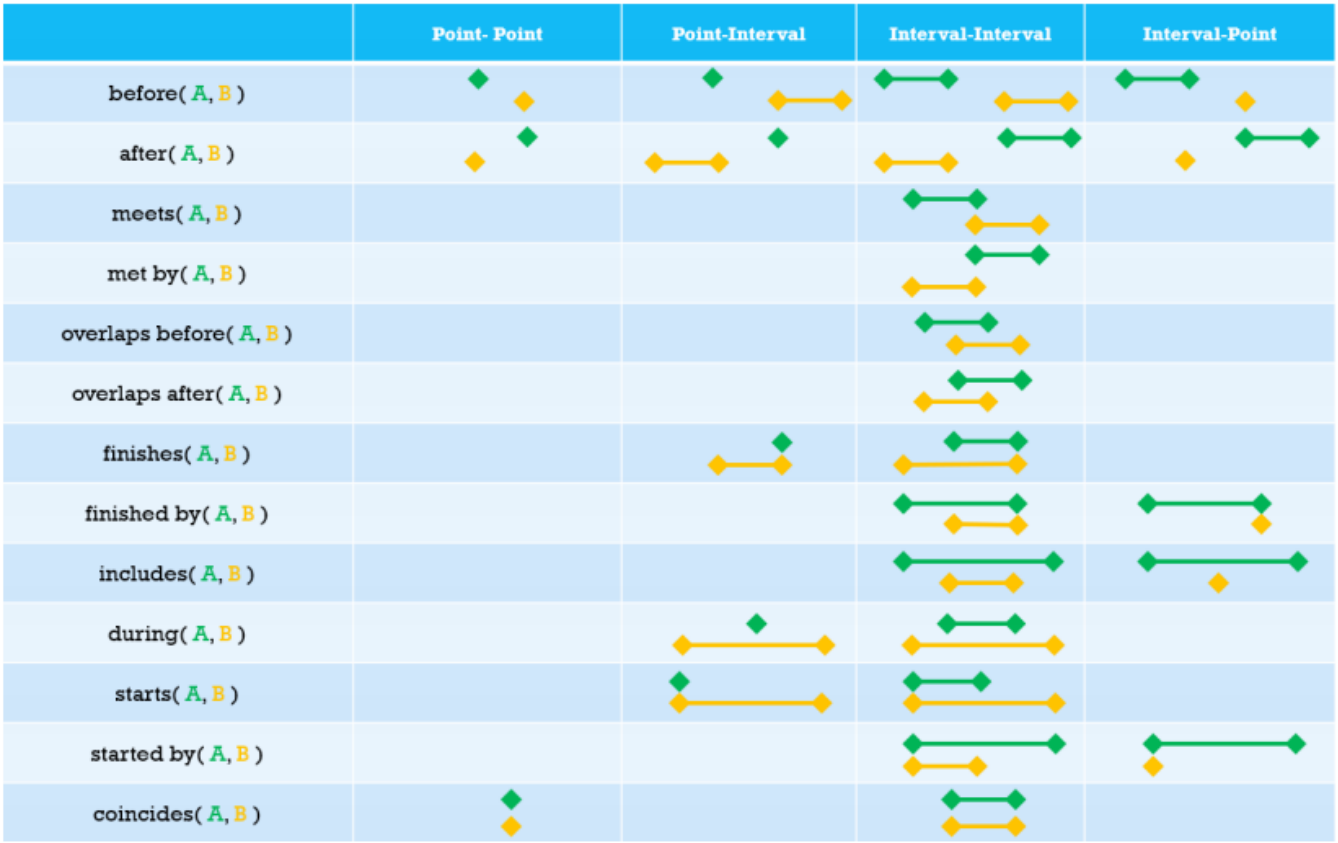


Таблица 78: Семантика функций диапазона

Имя(параметры)	**Значение true if and only if (для каждой сигнатуры соответственно)**	**Пример**
(a) before*(point1, point2)* (b) before*(point, range)* (c) before*(range, point)* (d) before*(range1,range2)*	(a) point1 < point2 (b) point < range.start or (point = range.start and not(range.start included)) (c) range.end < point	before(1, 10) = true before(10, 1) = false before(1, [1..10]) = false before(1, (1..10]) = true before(1, [5..10]) = true before([1..10], 10) = false before([1..10), 10) = true before([1..10], 15) = true

	or (range.end = point and not(range.end included)) (d) range1.end < range2.start or ((not(range1.end included) or not(range2.start included)) and range1.end = range2.start)	before([1..10], [15..20]) = true before([1..10], [10..20]) = false before([1..10), [10..20]) = true before([1..10], (10..20]) = true
(a) after(point1, point2) (b) after(point, range) (c) after(range, point) (d) after(range1, range2)	(a) point1 > point2 (b) point > range.end or (point = range.end and not(range.end included)) (c) range.start > point or (range.start = point and not(range.start included)) (d) range1.start > range2.end or ((not(range1.start included) or not(range2.end included)) and range1.start = range2.end)	after(10, 5) = true after(5, 10) = false after(12, [1..10]) = true after(10, [1..10)) = true after(10, [1..10]) = false after([11..20], 12) = false after([11..20], 10) = true after((11..20], 11) = true after([11..20], 11) = false after([11..20], [1..10]) = true after([1..10], [11..20]) = false after([11..20], [1..11)) = true after((11..20], [1..11]) = true

(a) meets(range1, range2)	(a) range1.end included and range2.start included and range1.end = range2.start	meets([1..5], [5..10]) = true meets([1..5), [5..10]) = false meets([1..5], (5..10]) = false meets([1..5], [6..10]) = false
(a) met by(range1, range2)	(a) range1.start included and range2.end included and range1.start = range2.end	met by([5..10], [1..5]) = true met by([5..10], [1..5)) = false met by((5..10], [1..5]) = false met by([6..10], [1..5]) = false
(a) overlaps(range1, range2)	(a) (range1.end > range2.start or (range1.end = range2.start and (range1.end included or range2.end included))) and (range1.start < range2.end or (range1.start = range2.end and range1.start included and range2.end included))	overlaps([1..5], [3..8]) = true overlaps([3..8], [1..5]) = true overlaps([1..8], [3..5]) = true overlaps([3..5], [1..8]) = true overlaps([1..5], [6..8]) = false overlaps([6..8], [1..5]) = false overlaps([1..5], [5..8]) = true overlaps([1..5], (5..8]) = false overlaps([1..5), [5..8]) = false overlaps([1..5), (5..8]) = false overlaps([5..8], [1..5]) = true overlaps((5..8], [1..5]) = false overlaps([5..8], [1..5)) = false overlaps((5..8], [1..5)) = false
(a) overlaps before(range1, range2)	(a) (range1.start < range2.start or (range1.start = range2.start and range1.start included	overlaps before([1..5], [3..8]) = true overlaps before([1..5], [6..8]) = false overlaps before([1..5], [5..8]) = true overlaps before([1..5], (5..8]) = false overlaps before([1..5), [5..8]) = false overlaps before([1..5), (1..5]) = true

	and range2.start included)) and (range1.end > range2.start or (range1.end = range2.start and range1.end included and range2.start included)) and (range1.end < range2.end or (range1.end = range2.end and (not(range1.end included) or range2.end included)))	overlaps before([1..5], (1..5]) = true overlaps before([1..5), [1..5]) = false overlaps before([1..5], [1..5]) = false
(a) overlaps after(range1, range2)	(a) (range2.start < range1.start or (range2.start = range1.start and range2.start included and not(range1.start included))) and (range2.end > range1.start or (range2.end = range1.start and range2.end included and range1.start included))	Overlaps after([3..8], [1..5]) = true Overlaps after([6..8], [1..5]) = false Overlaps after([5..8], [1..5]) = true Overlaps after((5..8], [1..5]) = false Overlaps after([5..8], [1..5)) = false Overlaps after((1..5], [1..5)) = true Overlaps after((1..5], [1..5]) = true Overlaps after([1..5], [1..5)) = false Overlaps after([1..5], [1..5]) = false

	and (range2.end < range1.end or (range2.end = range1.end and (not(range2.end included) or range1.end included)))	
(a) finishes(point, range) (b) finishes(range1, range2)	(a) range.end included and range.end = point (b) range1.end included = range2.end included and range1.end = range2.end and (range1.start > range2.start or (range1.start = range2.start and (not(range1.start included) or range2.start included)))	finishes(10, [1..10]) = true finishes(10, [1..10)) = false finishes([5..10], [1..10]) = true finishes([5..10), [1..10]) = false finishes([5..10), [1..10)) = true finishes([1..10], [1..10]) = true finishes((1..10], [1..10]) = true
(a) finished by(range, point) (b) finished by(range1, range2)	(a) range.end included and range.end = point (b) range1.end included = range2.end included and range1.end = range2.end and (range1.start < range2.start	finished by([1..10], 10) = true finished by([1..10), 10) = false finished by([1..10], [5..10]) = true finished by([1..10], [5..10)) = false finished by([1..10), [5..10)) = true finished by([1..10], [1..10]) = true finished by([1..10], (1..10]) = true

	or (range1.start = range2.start and (range1.start included or not(range2.start included))))	
(a) includes(range, point) (b) includes(range1, range2)	(a) (range.start < point and range.end > point) or (range.start = point and range.start included) or (range.end = point and range.end included) (b) (range1.start < range2.start or (range1.start = range2.start and (range1.start included or not(range2.start included)))) and (range1.end > range2.end or (range1.end = range2.end and (range1.end included or not(range2.end included))))	includes([1..10], 5) = true includes([1..10], 12) = false includes([1..10], 1) = true includes([1..10], 10) = true includes((1..10], 1) = false includes([1..10), 10) = false includes([1..10], [4..6]) = true includes([1..10], [1..5]) = true includes((1..10], (1..5]) = true includes([1..10], (1..10)) = true includes([1..10), [5..10)) = true includes([1..10], [1..10)) = true includes([1..10], (1..10]) = true includes([1..10], [1..10]) = true
(a) during(point, range) (b) during(range1, range2)	(a) (range.start < point and range.end > point)	during(5, [1..10]) = true during(12, [1..10]) = false during(1, [1..10]) = true

	or (range.start = point and range.start included) or (range.end = point and range.end included) (b) (range2.start < range1.start or (range2.start = range1.start and (range2.start included or not(range1.start included)))) and (range2.end > range1.end or (range2.end = range1.end and (range2.end included or not(range1.end included))))	during(10, [1..10]) = true during(1, (1..10]) = false during(10, [1..10)) = false during([4..6], [1..10]) = true during([1..5], [1..10]) = true during((1..5], (1..10]) = true during((1..10), [1..10]) = true during([5..10), [1..10)) = true during([1..10), [1..10]) = true during((1..10], [1..10]) = true during([1..10], [1..10]) = true
(a) starts(point, range) (b) starts(range1, range2)	(a) range.start = point and range.start included (b) range1.start = range2.start and range1.start included = range2.start included and (range1.end < range2.end or (range1.end = range2.end and (not(range1.end included)	starts(1, [1..10]) = true starts(1, (1..10]) = false starts(2, [1..10]) = false starts([1..5], [1..10]) = true starts((1..5], (1..10]) = true starts((1..5], [1..10]) = false starts([1..5], (1..10]) = false starts([1..10], [1..10]) = true starts([1..10), [1..10]) = true starts((1..10), (1..10)) = true

	or range2.end included)))	
(a) started by(range, point) (b) started by(range1, range2)	(a) range.start = point and range.start included (b) range1.start = range2.start and range1.start included = range2.start included and (range2.end < range1.end or (range2.end = range1.end and (not(range2.end included) or range1.end included)))	tarted by([1..10], 1) = true started by((1..10], 1) = false started by([1..10], 2) = false started by([1..10], [1..5]) = true started by((1..10], (1..5]) = true started by([1..10], (1..5]) = false started by((1..10], [1..5]) = false started by([1..10], [1..10]) = true started by([1..10], [1..10)) = true started by((1..10), (1..10)) = true
(a) coincides(point1, point2) (b) coincides(range1, range2)	(a) point1 = point2 (b) range1.start = range2.start and range1.start included = range2.start included and range1.end = range2.end and range1.end included = range2.end included	coincides(5, 5) = true coincides(3, 4) = false coincides([1..5], [1..5]) = true coincides((1..5), [1..5]) = false coincides([1..5], [2..6]) = false

10.3.4.8 Временные встроенные функции

Функции, перечисленные ниже, предоставляют общие вспомогательные утилиты при работе с значениями date или date and time; приводятся в таблице 1

Таблица 79: Семантика временных встроенных функций

Имя(параметры)	**Домен параметров**	**Описание**	**Пример**
day of year(date)	date или date and time	Возвращает порядковый номер дня года по григорианскому календарю	day of year(date(2019, 9, 17)) = 260
day of week(date)	date или date and time	Возвращает день недели в соответствии с нумерацией григорианского календаря: «понедельник», «вторник», «среда», «четверг», «пятница», «суббота», «воскресенье»	day of week(date(2019, 9, 17)) = "Tuesday"
month of year(date)	date или date and time	Возвращает порядковый номер недели в году по григорианскому календарю	week of year(date(2019, 9, 17)) = 38 week of year(date(2003, 12, 29)) = 1 week of year(date(2004, 1, 4)) = 1 week of year(date(2005, 1, 1)) = 53 week of year(date(2005, 1, 3)) = 1 week of year(date(2005, 1, 9)) = 1

10.3.4.9 Сортировка

Сортировка списка с помощью функции упорядочивания. Например,

sort(list: [3,1,4,5,2], precedes: function(x,y) x < y) = [1,2,3,4,5]

Таблица 80: Семантика функции сортировки

Имя параметра (* означает, что параметр опциональный)	**Домен**
list	Список любого элемента. Следует с осторожностью применять значение null
precedes	Булева функция с 2 аргументами, которые определяются для каждой пары элементов списка

10.3.4.10 Функция контекста

Таблица 81 определяет функции контекста.

Таблица 81: Семантика функции сортировки

Имя(параметры)	Домен параметров	Описание	Пример
get value(m, key)	context, string	Выбор значения записи с именем ключа из контекста m	*get value({key1 : "value1"}, "key1") = "value1"* *get value({key1 : "value1"}, "unexistent-key") = null*
get entries(m)	context	Создает список пар ключ-значение из контекста m	*get entries({key1 : "value1", key2 : "value2"}) = [{ key : "key1", value : "value1" }, {key : "key2", value : "value2"}]*

10.4. Семантика выполнения службы решений

В FEEL предусмотрена семантика выполнения служб решений, которые определяются в моделях принятия решений, где FEEL – это язык выражений. Служба принятия решений семантически эквивалентна функции FEEL, параметрами которой являются входные данные решения, а логика представляет собой контекст, собранный из решений службы решений и требований к знаниям.

Реализации службы решений ДОЛЖНЫ возвращать результат, как описано выше, и МОГУТ возвращать дополнительную информацию, такую как промежуточные результаты, записи журнала, информацию об отладке, сообщения об ошибках, аннотации правил и т. д. Формат любой дополнительной информации не указывается.

Каждое выражение FEEL в модели принятия решений имеет семантику выполнения. `LiteralExpression` (текст FEEL), определенное в разделе 10.3. «Табличные выражения», в подразделе 10.2.2 FEEL, может быть преобразовано в текст FEEL и, следовательно, также имеет семантику выполнения.

Напомним, что `DecisionService` определяется четырьмя списками: `inputData`, `inputDecisions`, `outputDecisions`, и `encapsulatedDecisions`. Списки не являются независимыми и, следовательно, не требуется указывать их все. Например, каждое требуемое решение (прямое и косвенное) из `outputDecisions` должно быть `encapsulatedDecision`, `inputDecision` или обязательным элементом `inputDecision`. Для простоты в дальнейшем мы предполагаем, что все четыре списка правильно и полностью определены.

Семантика выполнения `DecisionService` определяется путем сопоставления `DecisionService` с функцией FEEL *F*. Допустим S – `DecisionService` с входными данными id_1, id_2, ..., входными решениями di_1, di_2, ..., инкапсулированными решениями de_1, de_2, ... , и выходными решениями do_1, do_2, ... У входных данных id_i есть квалифицированное имя $id_i.n$. У каждого решения d_i есть квалифицированное имя n_{di} и выражение логики решения e_d. В решение могу входить требования к знаниям. В частности, для решений может требоваться `BusinessKnowledgeModels` bkm_1, bkm_2, ... и `DecisionServices` s_1, s_2, ... `BusinessKnowledgeModels`. У модели бизнес-знаний есть следующие квалифицированные имена n_{bkm_i}. и encapsulatedLogic $bkm_i.f$. У `DecisionServices` есть квалифицированные имена ns_i и эквивалентная логика f_{si}, где эквивалентная логика определяется

рекурсивно, связывая s_i с S.

Синтаксис функции FEEL F: *function*(n_{id1}, n_{id2}, ..., n_{di1}, n_{di2} ...) *C.result*, где *C* – контекст

{

$n_{s1} : f_{s1}$, $n_{s2} : f_{s2}$, ...,

$n_{bkm1} : f_{bkm1}$, $n_{bkm2} : f_{bkm2}$, ...,

$n_{de1} : e_{de1}$, $n_{de2} : e_{de2}$, ...,

result: { $n_{do1} : e_{do1}$, $n_{do2} : e_{do2}$, ...*}*

},

таким образом *s1*, bkm_i, de_i и do_i частично упорядочены по требованиям (например, контекстная запись для требуемого решения располагается перед решением, которое ссылается на эту запись).

Если элемент E (решение, входные данные, служба решений или модель бизнес-знаний) определяется в той же модели принятия решений, что и S, его квалифицированное имя будет записываться, как E. В противном случае, квалифицированное имя элемента записывается, как IE, где I - имя импортируемого элемента, ссылающегося на модель, в которой определяется E.

Семантика выполнения S – это функция FEEL (*F*). При вызове значений из семантического домена FEEL, привязанных к параметрам, которые представляют входные данные и принимаемые решения, функция возвращает:

- значение одного выходного решения (в случае одного выходного решения);
- контекст, состоящий из всех выходных значений выходных решений (в случае с множеством выходных решений).

Элементы XML ДОЛЖНЫ сопоставляться с семантическим доменом FEEL, как указано в разделе 10.3.3 «Данные XML». В противном случае детали синтаксиса входных/выходных данных и конвертация в / из FEEL не определены.

10.5 Метамодель

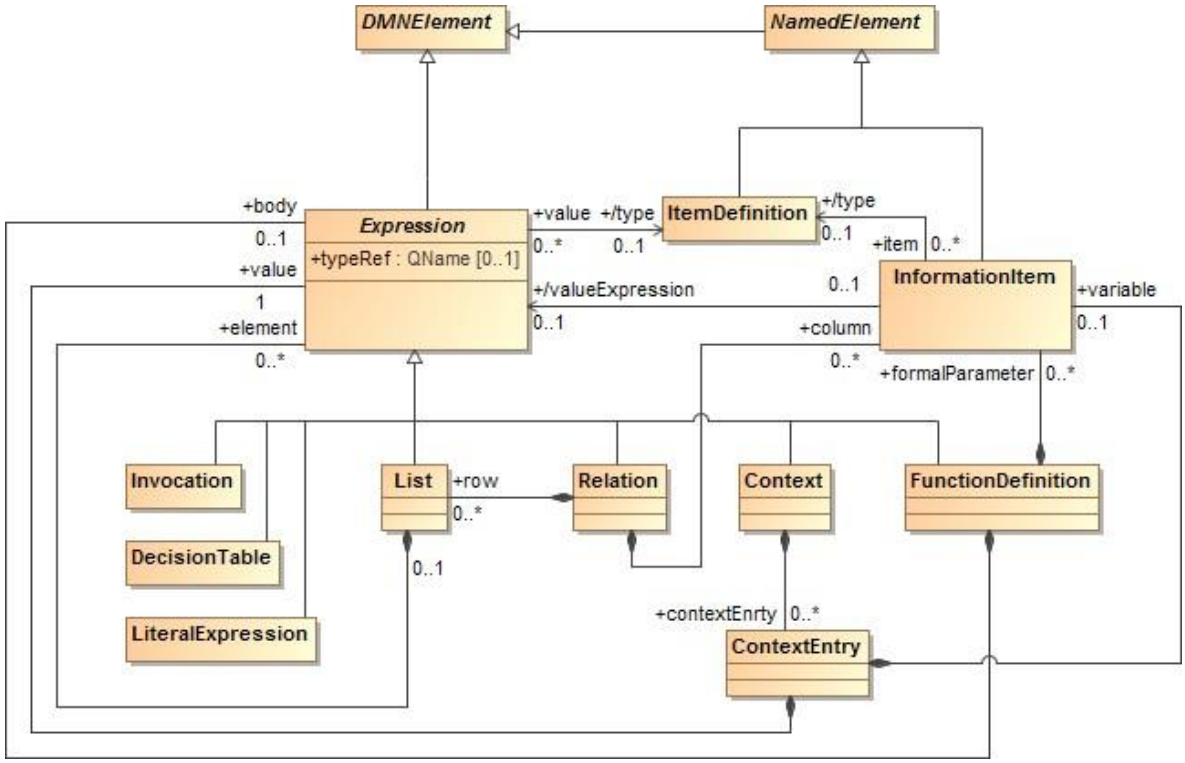


Рисунок 10.17. Диаграмма класса Expression

Класс `Expression` был расширен для обеспечения поддержки четырех новых видов табличных выражений, введенных FEEL, а именно: `Context`, `FunctionDefinition`, `Relation` и `List`.

Табличные выражения – это выражения, которые имеют стандартное схематическое представление (см. раздел 7.2.1 «Выражения» и 10.2.1 «Табличные выражения«). *Контексты*, *определения функций, отношения* и *списки* FEEL ДОЛЖНЫ быть смоделированы как элементы `Context`, `FunctionDefinition`, `Relation` и `List`, соответственно, и представлены по мере возможности в виде табличного выражения, то есть в тех случаях, когда они являются выражениями верхнего уровня, поскольку экземпляр `LiteralExpression` не может содержать другой элемент `Expression`.

10.5.1 Метамодель Context

`Context` состоит из любого количества `contextEntry`, которые являются экземплярами `ContextEntry`.

Элемент `Context` представлен схематически как **табличный контекст** (раздел 10.2.1.4. «Табличный контекст»). *Context* в FEEL (правило грамматики 57 и раздел 10.3.2.6 «Контекст») ДОЛЖЕН по возможности быть смоделирован как элемент `Context`.

`Context` наследует все атрибуты и ассоциации моделей от `Expression`. В таблице 82 представлены дополнительные атрибуты и ассоциации модели элемента `Context`.

Таблица 82. Дополнительные атрибуты и ассоциации модели `Context`

Атрибут	Описание
contextEntry: `ContextEntry` [*]	В этих атрибутах перечислены экземпляры `ContextEntry`, из которых состоит `Context`.

10.5.2 Метамодель ContextEntry

Класс `ContextEntry` используется для моделирования *записей контекста* FEEL, когда *контекст* моделируется как элемент `Context`. `ContextEntry` – это особый тип `DMNElement`, от которого он наследует необязательные атрибуты `id`, `description` и `label`.

Экземпляр `ContextEntry` состоит из необязательной переменной `variable`, которая является элементом `InformationItem`, чье имя `name` является *ключом* в *записи контекста*, и значения `value`, которое является экземпляром `Expression`, моделирующим *выражение* в *записи контекста*.

В таблице 83 представлены атрибуты и ассоциации модели элемента `ContextEntry`.

Таблица 83. Атрибуты и ассоциации модели `ContextEntry`

Атрибут	Описание
variable: `InformationItem` [0..1]	Содержащийся в `ContextEntry` экземпляр `InformationItem`, чье имя является *ключом* в смоделированной *записи контекста.*
value: `Expression`	Экземпляр `Expression`, который является *выражением* в `ContextEntry`.

10.5.3 Метамодель FunctionDefinition

У `FunctionDefinition` есть `formalParameters` и `body`. Элемент `FunctionDefinition` схематически представлен как **табличная функция** в соответствии с разделом 10.2.1.7 «Табличная функция». *Определение функции* FEEL (правило грамматики 55 и раздел 10.3.2.12 «Семантическое отображение») ЖЕЛАТЕЛЬНО моделировать как элемент `FunctionDefinition`, если это возможно.

`FunctionDefinition` наследует все атрибуты и ассоциации модели из `Expression`. В таблице 84 представлены дополнительные атрибуты и ассоциации модели элемента `FunctionDefinition`.

Таблица 84. Атрибуты и ассоциации модели `FunctionDefinition`

Атрибут	Описание
FormalParameter: `InformationItem` [*]	В этих атрибутах перечислены экземпляры `InformationItem`, которые являются параметрами `Context`.

body: Expression [0..1]	Экземпляр Expression, являющегося телом в FunctionDefinition.
kind: FunctionKind = FEEL { FEEL | Java | PMML }	Атрибут kind определяет тип FunctionDefinition. Значением по умолчанию является FEEL. Также поддерживаются значения Java и PMML

10.5.4 Метамодель List

List представляет собой простой список элементов element, которые являются экземплярами Expression. Элемент List представлен схематически в виде **табличного списка**, см. раздел 10.2.1.5 «Табличный список». *Список* FEEL (правило грамматики 54 и раздел 10.3.2.12 «Семантическое отображение») по возможности ЖЕЛАТЕЛЬНО моделировать как элемент List.

List наследует все атрибуты и ассоциации модели от Expression. В таблице 85 представлены дополнительные атрибуты и ассоциации модели элемента List.

Таблица 85. Атрибуты и ассоциации модели List

Атрибут	Описание
element: Expression [*]	В этих атрибутах перечислены экземпляры Expression, которые являются элементами List.

10.5.5 Метамодель Relation

Relation – это краткое обозначение списка одинаковых контекстов. Вместо повторяющихся ContextEntrys, в Relation используется столбец column, а List используется для каждой строки row с одним из List's expression для каждого значения столбца.

Relation наследует все атрибуты и ассоциации модели от Expression. В таблице 86 представлены дополнительные атрибуты и ассоциации модели элемента Relation.

Таблица 86. Атрибуты и ассоциации модели Relation

Атрибут	Описание
row: List [*]	В этих атрибутах перечислены экземпляры List, из которых состоят строки Relation.
column: InformationItem [*]	В этих атрибутах перечислены экземпляры InformationItem, которые определяют столбцы в Relation.

10.6 Примеры

В целях быстрого ознакомления с FEEL приведем несколько примеров.

Выражения FEEL могут ссылаться на другие выражения FEEL по имени. Именованные выражения содержатся в контексте. Выражения оцениваются в области, которая представляет собой список контекстов для распознавания имен. Результатом оценки является элемент семантического домена FEEL.

10.6.1 Контекст

На рисунке 10.18 в качестве примера показан табличный контекст. Такой контекст может возникнуть в нескольких случаях. Он может быть частью логики принятия решения для одного сложного решения. Или это может быть контекст, эквивалентный части ГТР (см. раздел 10.4. «Семантика выполнения службы решений»), где *заявитель*, *запрашиваемый продукт* и *кредитная история* являются экземплярами входных данных; *ежемесячный доход* и *ежемесячные расходы* – суб-решениями, а *РПП* – моделью бизнес-знаний.

<table>
<tr><td rowspan="6">заявитель</td><td>возраст</td><td colspan="3">51</td></tr>
<tr><td>семейное положение maritalStatus</td><td colspan="3">"ЗАМУЖЕМ"</td></tr>
<tr><td>существующий клиент existingCustomer</td><td colspan="3">false</td></tr>
<tr><td rowspan="3">ежемесячно</td><td>доходы</td><td colspan="2">10000</td></tr>
<tr><td>погашения</td><td colspan="2">2500</td></tr>
<tr><td>затраты</td><td colspan="2">3000</td></tr>
<tr><td rowspan="4">запрошенный продукт</td><td colspan="2">Тип продукта</td><td>"СТАНДАРТНЫЙ КРЕДИТ"</td></tr>
<tr><td colspan="2">ставка</td><td>0.25</td></tr>
<tr><td colspan="2">срок</td><td>36</td></tr>
<tr><td colspan="2">размер</td><td>100000.00</td></tr>
<tr><td>ежемесячный доход</td><td colspan="3">applicant.monthly.income
(заявитель.ежемесячно.доходы)</td></tr>
<tr><td>ежемесячные расходы</td><td colspan="3">applicant.monthly.repayments
(заявитель.ежемесячно.погашения)

applicant.monthly.expenses
(заявитель.ежемесячно.затраты),</td></tr>
<tr><td rowspan="3">кредитная история</td><td>дата регистрации</td><td>событие</td><td>вес</td></tr>
<tr><td>дата("2008-03-12")</td><td>"ипотека"</td><td>100</td></tr>
<tr><td>дата("2011-04-01")</td><td>"предупреждение о потере права выкупа заложенного имущества "</td><td>150</td></tr>
<tr><td rowspan="2">размер периодического платежа (РПП)</td><td colspan="3">(ставка, срок, сумма)</td></tr>
<tr><td colspan="3">(сумма *ставка/12) / (1 – (1 + ставка/12)**- срок)</td></tr>
</table>

Рисунок 10.18: Пример контекста

Обратите внимание, что есть 6 записей контекста верхнего уровня, представленных шестью строками таблицы. Значение записи контекста с именем 'заявитель' само по себе является контекстом, как и значение записи контекста с именем 'ежемесячно'. Значение записи контекста с именем 'ежемесячные расходы' представляет собой список, значение записи контекста с именем 'кредитная история' – это отношение, то есть список из двух контекстов, один контекст для каждой строки. Значение записи контекста с именем

'РПП' – это функция с параметрами 'ставка', 'срок' и 'сумма'.

В следующих примерах используется приведенный выше контекст. Каждый пример имеет пару эквивалентных выражений FEEL, разделенных горизонтальной линией. Оба выражения обозначают один и тот же элемент в семантической области. Второе выражение, «ответ», является буквальным значением.

10.6.2 Расчеты

```
monthly income * 12
```

```
120000
```

Контекст определяет *ежемесячный доход* как *applicant.monthly.income (заявитель.ежемесячно.доходы)*, который также определяется в контексте как 10 000. Ежемесячный доход в двенадцатикратном размере составляет 120 000.

10.6.3 If, In

```
if applicant.maritalStatus in ("M","S") then "valid" else "not
valid"
```

```
"valid"
```

Тест *in* определяет, будет ли выражение левой стороны соответствовать списку значений или диапазонов с правой стороны. Если оно соответствует, выражение *if* возвращает значение выражения *then*. В противном случае возвращается значение выражения *else*.

10.6.4 Сумма записей в списке

```
sum(monthly) outgoings)
```

```
5500
```

Ежемесячные расходы вычисляются в контексте в виде списка *[applicant.monthly.repayments, applicant.monthly.expenses (заявитель.ежемесячно.рекламы), (заявитель. ежемесячно.расходы)]* или [2500, 3000]. Квадратные скобки не требуется записывать в табличный контекст.

10.6.5 Вызов функции РПП, определяемой пользователем

С помощью функции РПП, определенной в контексте, вычисляются ежемесячные платежи для заданной процентной ставки, количества месяцев и суммы кредита.

```
PMT(requested product .rate,
    requested product .term,
    requested product .amount)
```

```
3975.98259012555233827844010011243 1
```

Функция вызывается текстовым способом с использованием списка аргументов в скобках после имени функции. Аргументы определяются в контексте и равны 0,25, 36 и 100 000 соответственно.

10.6.6 Суммарный вес недавней кредитной истории

```
sum(credit history[record date > date("2011-01-01")].weight)
```

```
150
```

Это сложный «однострочный скрипт», который будет полезен для включения в составные подвыражения:

- built-in: *sum*
    - path expression ending in *.weight*
        - filter: *[record date > date("2011-01-01")]*
            - name resolved in context: *credit history*

Выражение в квадратных скобках, следующее за выражением списка, фильтрует список. *Кредитная история* определяется в контексте как отношение, то есть список подобных контекстов. Только последний элемент в отношении удовлетворяет фильтру. Первый элемент неактуален. Выражение пути, заканчивающееся на *.weight*, выбирает значение записи *weight* из контекста или списка контекстов, которым удовлетворяет фильтр. Вес *weight* последнего элемента в кредитной истории равен 150. Это единственный элемент, который удовлетворяет фильтру, поэтому сумма также равна 150.

10.6.7 Определить, содержит ли кредитная история событие банкротства

```
some ch in credit history satisfies ch.event = "bankruptcy"
```

```
false
```

Выражение *some* определяет, удовлетворяет ли тесту хотя бы один элемент в списке или отношение.

В контексте кредитной истории нет событий банкротства.

Эта страница намеренно оставлена пустой.

11. Примеры DMN

11.1 Пример 1: Происхождение

11.1.1 Введение

В данном разделе приводятся примеры использования **DMN** для моделирования и принятия решений в простом бизнес-процессе, созданном при помощи нотации **BPMN**, включая решения, которые должны быть автоматизированы в службах решений, вызываемых из системы управления бизнес-процессами.

11.1.2 Модель бизнес-процесса

На рис. 11.1 показан простой процесс выдачи кредитов, смоделированный в **BPMN 2.0**. Процесс включает в себя следующие этапы: заявка на получение кредита, получение данных из кредитного бюро только в случае необходимости и автоматическое принятие решения о том, следует ли утвердить, отклонить заявку или направить ее на рассмотрение человеку. Если заявка отправляется на рассмотрение, осуществляется сбор документов заявителя и кредитный специалист выносит решение по этому делу. Процесс состоит из следующих компонентов:

- задача «**Сбор данных по заявке**», в ходе выполнения которой собираются данные о запрашиваемом продукте и о Заявителе (например, через форму онлайн-заявки);
- задача «**Определить стратегию бюро**» вызывает службу принятия решений, передавая данные о Запрошенном продукте и Заявителе. Служба возвращает два решения: Стратегия и Тип запроса в Бюро;
- **шлюз** использует значение Стратегии для маршрутизации процесса по одной из следующих веток: Отклонить заявку, Сбор данных бюро или определить маршрут;
- задача «**Получить данные бюро**» собирает данные из кредитного бюро в соответствии с решением «Тип запроса в Бюро», затем процесс идет по ветке Определить маршрут;
- задача «**Определить маршрут**» вызывает службу принятия решений, передавая Запрошенный продукт, Данные заявителя и Данные Бюро (если задача «Получить данные бюро» не была выполнена, значение Данные Бюро равно null). Служба возвращает одно решение: Маршрут;
- **шлюз** использует значение «Маршрут» для маршрутизации процесса по одной из следующих веток: Утвердить заявку, Отправить на рассмотрение или Отклонить заявку;
- в ходе выполнения задачи «**Сбор документов**» запрашиваются и загружаются документы заявителя для оформления заявки;
- задача «**Рассмотреть заявку**» позволяет кредитному менеджеру рассмотреть дело и решить, следует ли принять или отклонить заявку;
- **шлюз** использует решение кредитного менеджера для маршрутизации процесса по одной из следующих веток: Утвердить заявку или Отклонить заявку;
- в ходе выполнения задачи «**Утвердить заявку**» заявитель информируется о том, что его заявка была принята. Происходит выдача кредитного продукта;
- в ходе выполнения задачи «**Отклонить заявку**» заявитель информируется о том, что его заявка была отклонена.

Обратите внимание, что в этом примере в **BPMN 2.0** в качестве бизнес-правил представлены две точки принятия решения (автоматизированные вызовы служб принятия решений); третья точка принятия решения (принятие решения человеком) представляется в виде пользовательской задачи.

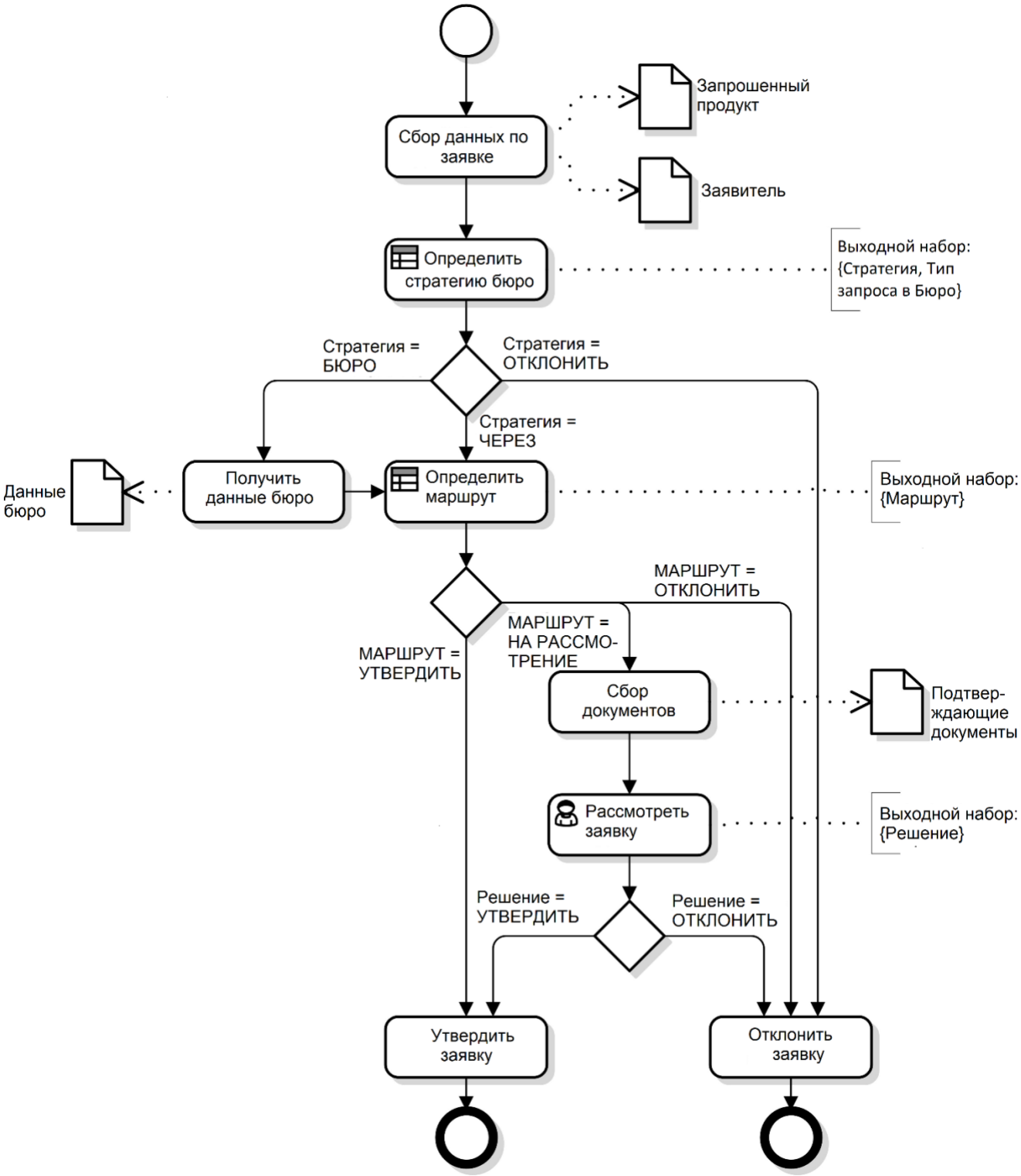


Рисунок 11.1: Пример бизнес-процесса

11.1.3 Уровень требований к решению

Примеры в этой главе были подготовлены с использованием программного обеспечения, которое отображает иконки к элементам. Хотя добавление таких иконок допустимо, оно не является нормативным.

11.1.3.1 Диаграммы требований к решениям

На рисунке 11.2 показана ДТР всех решений в этом бизнес-процессе. На ней изображены четыре источника входных данных для принятия решений (запрошенный продукт, данные заявителя, данные бюро и подтверждающие документы) и четыре решения, результаты которых используются в бизнес-процессе (Стратегия, Тип запроса в Бюро, Маршрут и Решение). Между ними находятся промежуточные решения: оценки риска, доступности и соответствия критериям. К особенностям данной ДТР можно отнести:

- она описывает как автоматизированные, так принимаемые человеком решения;
- некоторые решения (например, категория риска до оценки бюро) и входные данные (например, данные заявителя) требуются для принятия нескольких решений, то есть сеть требований к информации не имеет древообразную структуру;
- модели бизнес-знаний (см. определение доступности) могут быть вызваны несколькими решениями;
- модели бизнес-знаний (см. Коэффициент непредвиденных обстоятельств) могут использоваться другими моделями бизнес-знаний;
- в некоторых решениях отсутствуют модели бизнес-знаний;
- источники знаний могут предоставлять полномочия для принятия нескольких решений и / или моделей бизнес-знаний.

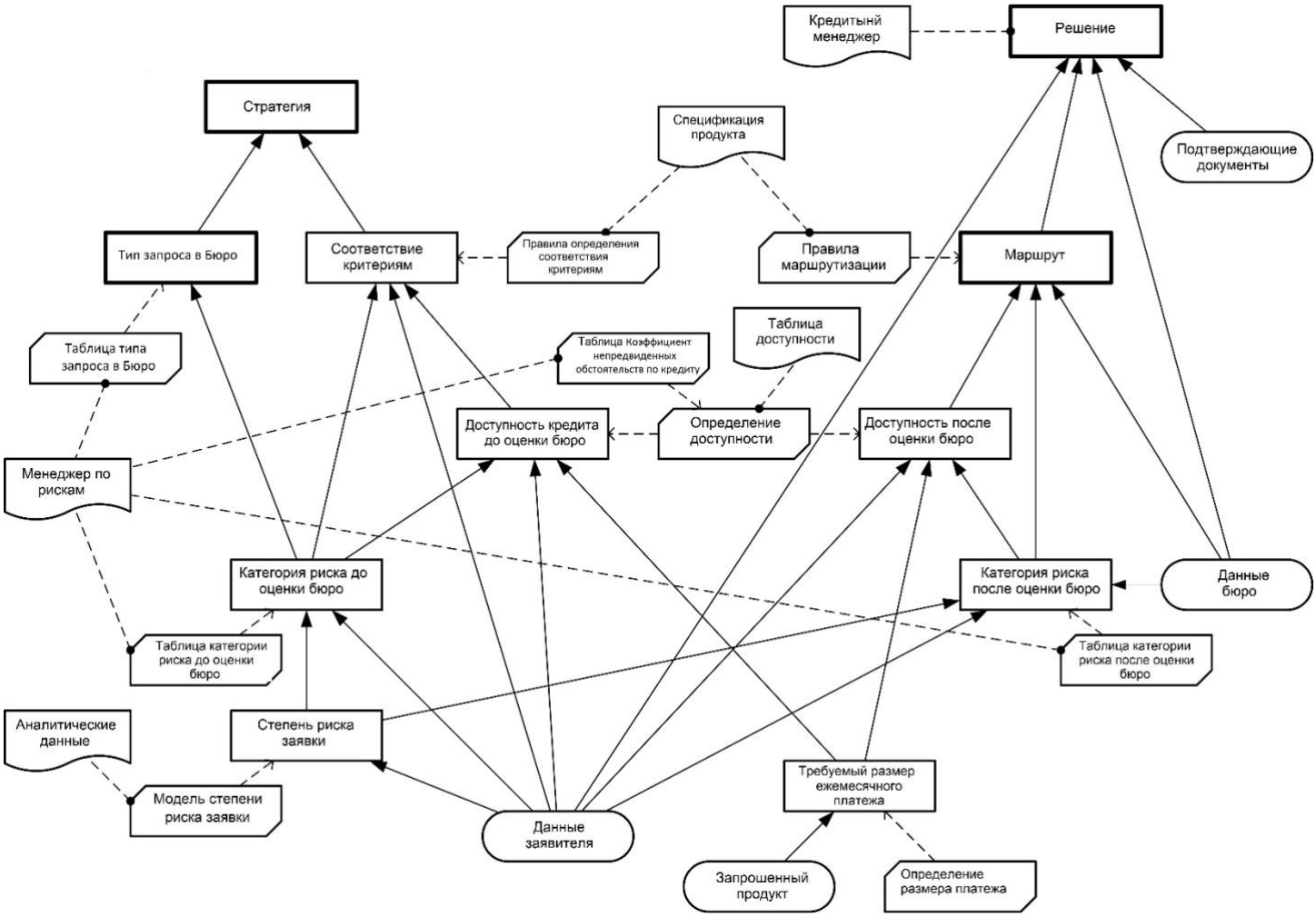


Рисунок 11.2: ДТР для всех автоматизированных решений

Чтобы информацию было удобнее воспринимать, в качестве альтернативы, можно составить отдельные (но перекрывающиеся) ДТР для трех точек принятия решения:

- на рис. 11.3 показана ДТР решений, которые являются обязательными для точки принятия решения «Определить стратегию бюро». Иными словами, на рисунке 11.3 изображен подграфик требований для решений с типом вызова Стратегия и Тип запроса в Бюро. Эти решения автоматизируются посредством инкапсуляции в службе принятия решений, которая вызывается в этой точке. Необходимо, чтобы логика решений была полностью определена;

- на рисунке 11.4 показана ДТР для точки принятия решения «Определить маршрут». Иными словами, на рисунке 72 изображен подграфик требований для решения по маршрутизации («Маршрут»). Эти решения также автоматизируются с помощью службы принятия решений, и поэтому их логика должна быть полностью определена. Обратите внимание, что некоторые элементы отображаются как на рис. 11.3, так и на рисунке 11.4;

- на рис. 11.5 показана ДТР для точки принятия решения «Рассмотреть заявку». Иными словами, на рисунке 11.5 изображен подграфик требований для Решения по заявке. Это принимаемое человеком решение, которое не связано со спецификацией логики принятия решения. Однако

ДТР указывает, что кредитный менеджер учитывает результаты автоматизированного решения по маршрутизации («Маршрут») наряду с данными экземпляра, включая подтверждающие документы. (Подграфик требований решения по маршрутизации был скрыт на этой ДТР и обозначен многоточием …).

- На рисунке 11.6 изображена дополнительная ДТР для источника знаний «Аналитические данные по кредитным рискам». Иными словами, на рисунке 11.6 показаны требования, связывающие этот источник знаний с другими элементами. ДТР могут использоваться для формирования представлений, отличных от конкретного решения.

Все пять ДТР – рис. 11.2, рис. 11.3, рис. 11.4, рис. 11.5 и рис. 11.6 – являются различными представлениями одного и того же ГТР.

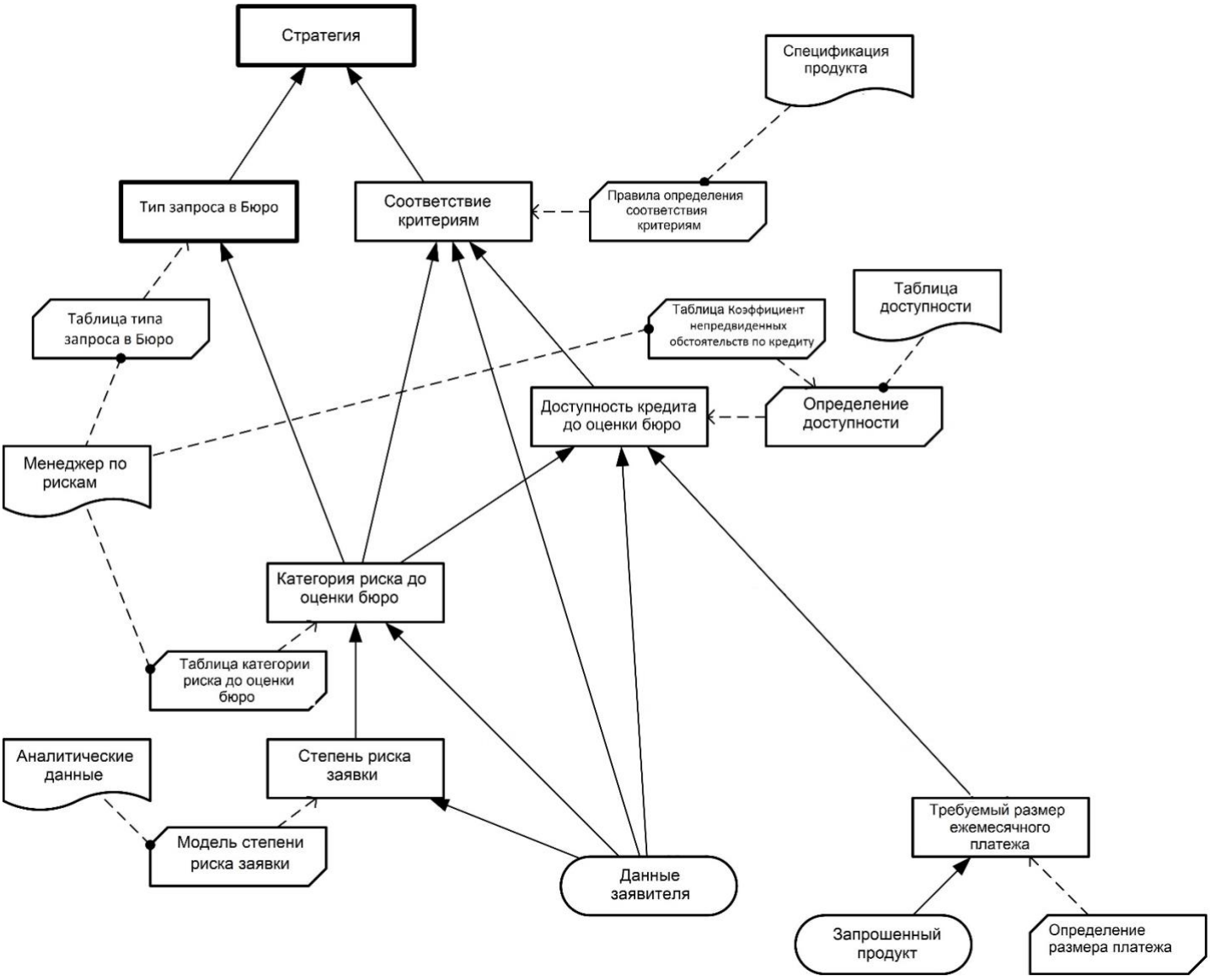


Рисунок 11.3: ДТР для точки принятия решения «Определить стратегию бюро».

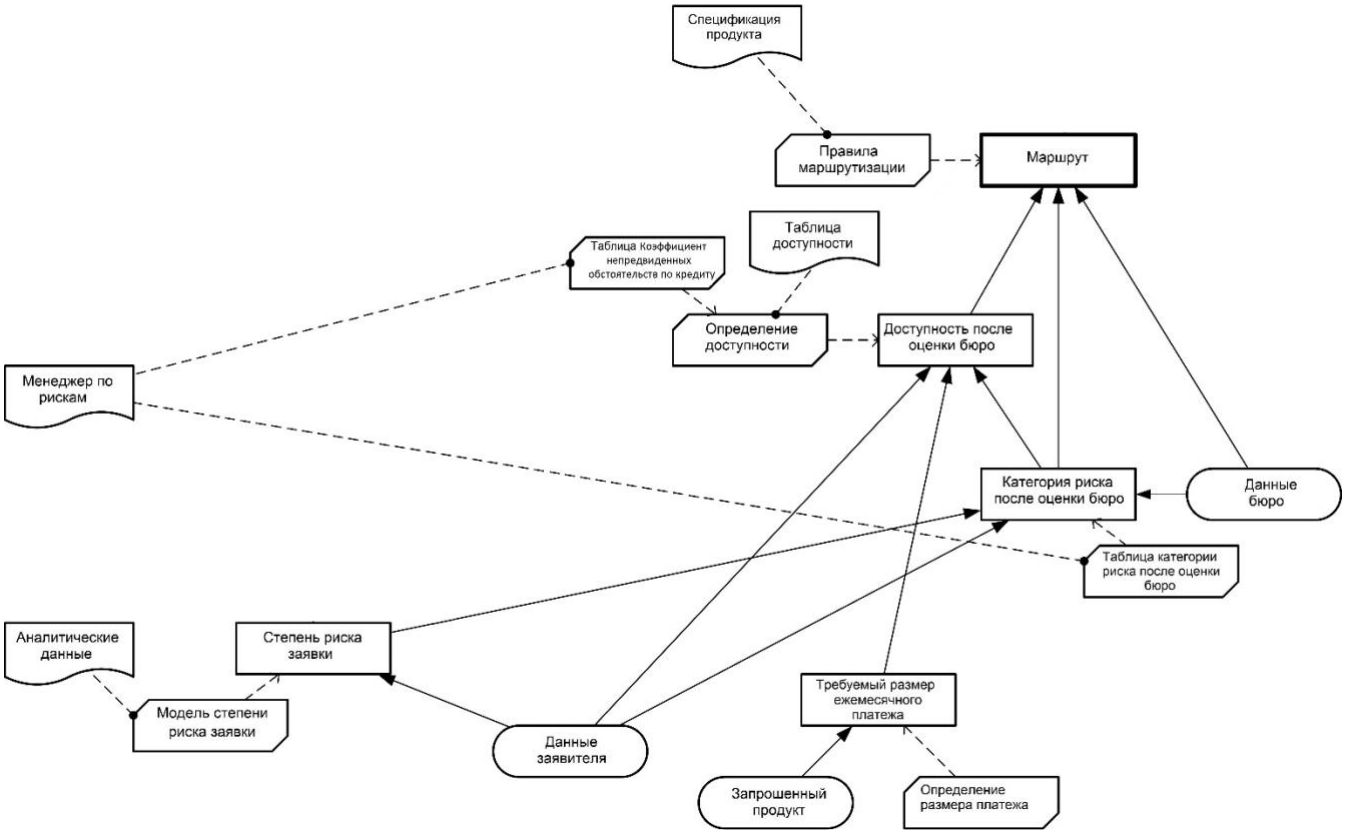


Рисунок 11.4: ДТР для точки принятия решения «Маршрут».

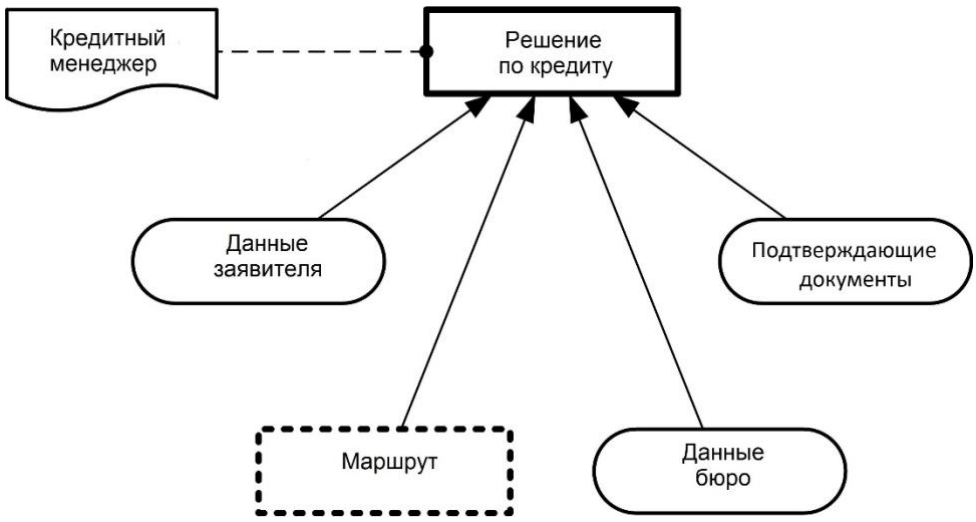


Рисунок 11.5: ДТР для точки принятия решения «Рассмотреть заявку».

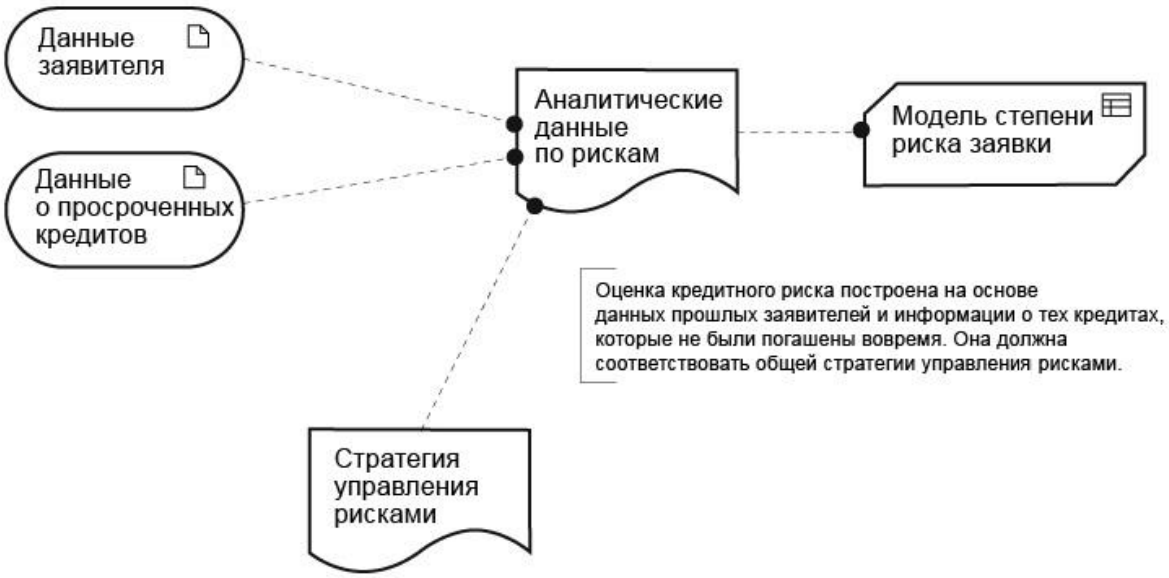


Рисунок 11.6: ДТР для источника знаний «Аналитические данные по кредитным рискам».

11.1.3.2 Элементы ГТР

11.1.3.2.1 Решения

ГТР, изображенный с помощью этих ДТР, показывает зависимости между следующими решениями:

- решение «**Стратегия**», вызывает таблицу «Стратегия», показанную на рисунке 11.9 (без инкапсуляции таблицы в модель бизнес-знаний). Решения «Тип запроса в Бюро» и «Соответствие критериям до оценки бюро» являются обязательными для решения «**Стратегия**»;
- решение «**Тип запроса в Бюро**» вызывает таблицу Типов запросов в Бюро, показанную на рисунке 11.11. Решение «Категория риска до оценки бюро» является обязательным для решения «**Тип запроса в Бюро**»;
- решение «**Соответствие критериям**» вызывает правила Соответствия критериям, показанные на рисунке 11.13. Наличие Данных заявителя, а также решения по категории риска до оценки бюро и по соответствию критериям до оценки бюро являются обязательными для решения «**Соответствие критериям**»;
- решение «**Доступность до оценки Бюро**» вызывает табличное выражение «Вычисление доступности», см. рис 11.24, которое, в свою очередь, вызывает таблицу коэффициентов непредвиденных обстоятельств, рис. 11.25. Наличие Данных заявителя, Категории риска до оценки Бюро, а также наличие решения «Требуемый размер ежемесячного платежа» является обязательным для принятия решения «**Доступность до оценки Бюро**»;
- решение «**Категория риска до оценки Бюро**» вызывает таблицу категорий рисков до оценки Бюро, показанную на рисунке 11.15. Данные заявителя и решение Степень риска заявки являются обязательными для «**Категория риска до оценки Бюро**»;
- решение «**Степень риска заявки**», требующее Данных заявителя, вызывает модель степени рисков заявки, см. рис. 11.17;
- решение «**Маршрут**» вызывает Правила маршрутизации, показанные на рисунке 11.19. Данные бюро, Доступность после оценки Бюро и решение «Категория риска после оценки Бюро» являются обязательными для решения «**Маршрут**»;

- решение «**Доступность после оценки Бюро**» вызывает табличное выражение вычисления «Доступность», см. рис 11.24, которое, в свою очередь, вызывает таблицу коэффициентов непредвиденных обстоятельств, рис. 11.25. Наличие Данных заявителя, Категории риска после оценки Бюро, а также решения «Требуемый размер ежемесячного платежа» является обязательным для принятия решения «**Доступность после оценки Бюро**»;
- решение «**Категория риска после оценки Бюро**» вызывает таблицу категорий рисков после оценки Бюро, показанную на рисунке 11.21. Данные заявителя, данные Бюро и решение Степень риска заявки являются обязательными для «**Категория риска до оценки Бюро**»;
- решение «**Требуемый размер ежемесячного платежа**», требующее данные по Запрошенному продукту, вызывает табличное выражение вычисления размера Платежа, см. рис. 11.27;
- **Решение по заявке**, требующее наличия Данных заявителя, Данных Бюро, подтверждающих документов и решения по маршрутизации, не имеет связанной логики принятия решения.

Для перечисленных выше решений указываются вопросы и допустимые ответы. Такой подход обычно применяется при моделировании решений, для которых не задается логика. Также вопросы и ответы могут использоваться для решений до тех пор, пока их логика не будет детально определена.

Вопросы, разрешенные ответы и описания для каждого решения приведены ниже.

Решение по заявке

Вопрос: Следует ли принять заявку, которая была направлена на рассмотрение?

Допустимые ответы: Да/Нет

Описание: Определите, следует ли принять или отклонить заявку, требующую рассмотрения, с учетом имеющихся данных и подтверждающих документов.

Степень риска заявки

Вопрос: Какова оценка риска для этой заявки?

Допустимые ответы: Число больше 70 и меньше 150

Описание: Логика принятия решения по «**Степени риска заявки»** вызывает модель бизнес-знаний «Модель степени рисков заявки», передавая данные следующим образом: Applicant data.Age – как параметр «Возраст», Applicant data.MaritalStatus – как параметр «Семейное положение» и Applicant data.EmploymentStatus – как параметр «Статус занятости».

Тип запроса в Бюро

Вопрос: Сколько данных необходимо запросить у кредитного бюро по этой заявке?

Допустимые ответы: Значение из явного списка «Запрос всех данных», «Упрощенный», «Без запроса»

Описание: Логика решения «**Тип запроса в Бюро»** вызывает таблицу типов запросов в Бюро, передавая данные следующим образом: результат решения «Категория риска до оценки Бюро» – как параметр «Категория риска до оценки Бюро».

Соответствие критериям

Вопрос: Соответствует ли данный заявитель требованиям, предъявляемым банком к заёмщикам для получения запрошенного кредита, учитывая только данные его заявления?

Допустимые ответы: Значение из явного списка «Соответствует», «Не соответствует»

Описание: Логика решения «**Соответствие критериям»** вызывает модель бизнес-знаний «Правила соответствия критериям», передавая данные следующим образом: Applicant data.Age – как параметр «Возраст», результат решения по категории риска до оценки бюро – как параметр «Категория риска до оценки бюро», результат решения по доступности до оценки Бюро – как параметр «Доступность до оценки Бюро».

Доступность до оценки Бюро

Вопрос: В состоянии ли заявитель выплатить запрошенный кредит, учитывая только данные его заявления?

Допустимые ответы: Да/Нет

Описание:

Логика решения «**Доступность до оценки Бюро**» вызывает модель бизнес-знаний «Вычисление доступности», передавая данные следующим образом: Applicant data.Monthly.Income – как параметр «Ежемесячный доход», Applicant data.Monthly.Repayments – как параметр «Ежемесячные выплаты», Applicant data. Monthly.Expenses – как параметр «Ежемесячные расходы», результат решения по категории риска до оценки Бюро – как параметр «Категория риска», результат решения по требуемому размеру ежемесячного платежа – как параметр «Требуемый размер ежемесячного платежа».

Доступность после оценки Бюро

Вопрос: В состоянии ли заявитель выплатить запрошенный кредит, учитывая все доступные данные?

Допустимые ответы: Да/Нет

Описание: Логика решения «**Доступность после оценки Бюро**» вызывает модель бизнес-знаний «Вычисление доступности», передавая данные следующим образом: Applicant data.Monthly.Income – как параметр «Ежемесячный доход», Applicant data.Monthly.Repayments – как параметр «Ежемесячные выплаты», Applicant data. Monthly.Expenses – как параметр «Ежемесячные расходы», результат решения по категории риска после оценки Бюро – как параметр «Категория риска», результат решения по требуемому размеру ежемесячного платежа – как параметр «Требуемый размер ежемесячного платежа».

Категория риска до оценки Бюро

Вопрос: Какую категорию риска можно присвоить данному заявителю, учитывая только данные его заявки?

Допустимые ответы: Значение из явного списка «Отказать», «Высокий риск», «Средний риск», «Низкий риск», «Очень низкий».

Описание: Логика решения «**Категория риска до оценки Бюро**» вызывает модель бизнес-знаний «Таблица категорий рисков до оценки Бюро», передавая данные следующим образом: Applicant data.ExistingCustomer – как параметр «Существующий клиент» и результат решения по степени риска заявки – как параметр «Степень риска заявки».

Категория риска после оценки Бюро

Вопрос: Какую категорию риска можно присвоить данному заявителю, учитывая все доступные данные?

Допустимые ответы: Значение из явного списка «Отказать», «Высокий риск», «Средний риск», «Низкий риск», «Очень низкий».

Описание: Логика решения «**Категория риска после оценки Бюро**» вызывает модель бизнес-знаний «Категория рисков после оценки Бюро», передавая данные следующим образом: Applicant data.ExistingCustomer – как параметр «Существующий клиент», Bureau data.CreditScore – как параметр «Кредитный оценка» и результат решения по степень риска заявки – как параметр «Степень риска заявки».

Обратите внимание, если «Данные Бюро» возвращают null (в виду выбора стратегии «ЧЕРЕЗ», которая предполагает пропуск задачи «Сбор данных бюро»), параметр «Кредитная оценка» будет null;

Требуемый размер ежемесячного платежа

Вопрос: Минимальный размер платежа, требуемый для этого кредитного продукта?

Допустимые ответы: Сумма в долларах больше нуля

Описание: Логика решения «**Требуемый размер ежемесячного платежа**» вызывает модель бизнес-знаний «Вычисление размера платежа», передавая данные следующим образом: Requested product.ProductType – как параметр «Тип продукта», Requested product.Rate – как параметр «Ставка», Requested product.Term – как параметр «Срок» и Requested product.Amount – как параметр «Сумма».

Маршрут

Вопрос: Какой маршрут необходимо выбрать для данного заявителя, учитывая все доступные данные?

Допустимые ответы: Значение из явного списка «Отказать», «Отправить на рассмотрение», «Принять без проверки»

Описание: Логика решения «**Маршрут**» вызывает модель бизнес-знаний «Правила маршрутизации», передавая данные следующим образом: Bureau data. Bankrupt передается как параметр «Банкрот», Bureau data . CreditScore – как параметр «Кредитная оценка», результат решения «Категория риска после оценки Бюро» – как параметр «Категория риска после оценки Бюро», а результат решения по доступности после оценки Бюро – как параметр «Доступность после оценки Бюро». Обратите внимание, если данные Бюро возвращают null (в виду выбора стратегии «ЧЕРЕЗ», которая предполагает пропуск задачи «Сбор данных бюро»), то параметры «Банкрот» и «Кредитная оценка» будут null;

Стратегия

Вопрос: Какая стратегия подходит для обработки данной заявки?

Допустимые ответы: Значение из явного списка «Отказать», «Бюро», «Через»

Описание: определяет полную таблицу решений с уникальным результатом. Значение «Стратегия» выводится из значений «Соответствие критериям» и «Тип запроса в Бюро».

11.1.3.2.2 Источники знаний

ГТР содержит следующие источники знаний:

Таблица доступности

Описание: Внутренняя электронная таблица, показывающая соотношение доходов, платежей, расходов, риска и доступности.

Тип: Политика

Опыт сотрудника кредитного отдела

Описание: База знаний с практическими рекомендациями по организации эффективной работы, отражающая накопленный опыт сотрудников кредитного отдела.

Тип: Опыт

Аналитические данные по кредитным рискам

Описание: Анализ показателей кредитного риска, позволяющий определить релевантные факторы для оценки риска заявки.

Тип: Аналитические выводы

Спецификация продукта

Описание: Описания продуктов, их стоимостная структура и критерии соответствия установленным требованиям.

Тип: Политика

Стратегия управления рисками

Описание: Общий подход к управлению рисками для финансового учреждения, включая его подход к риску заявки, кредитным непредвиденным обстоятельствам и оценке кредитного риска.

Тип: Политика

11.1.3.2.3 Входящие данные

ГТР содержит следующие входящие данные:

Данные заявителя

Описание: Информация о заявителе, включая личную информацию, семейное положение и доходы / расходы домохозяйства.

Данные Бюро

Описание: Внешняя кредитная оценка и информация о банкротстве, предоставленная бюро.

Данные о просроченных кредитах

Описание: Информация о просроченных ранее кредитах.

Запрашиваемый продукт

Описание: Подробное описание кредита, на который была подана заявка.

Подтверждающие документы

Описание: Документы, связанные с кредитом, которые не обрабатываются в электронном виде, но доступны для ознакомления при вынесении решения человеком.

11.1.3.2.4 Модель бизнес-знаний

Наконец, ГТР содержит следующие модели бизнес-знаний:

Правила соответствия критериям

Описание: логика решения «Правила соответствия критериям» определяет полную, упорядоченную в соответствии с приоритетом, таблицу решений с одним результатом. Значение Соответствия критериям выводится из категории риска до оценки бюро, доступности до оценки бюро и возраста.

Правила маршрутизации

Описание: логика решения «Правила маршрутизации» определяет полную, упорядоченную в соответствии с приоритетом, таблицу решений с одним результатом. Значение «Маршрут» выводится из категории риска после оценки бюро, доступности после оценки бюро, параметров Банкрот и Кредитная оценка.

Таблицы типа запроса в Бюро

Описание: логика решения «Таблицы типа запроса в Бюро» определяет полную таблицу решений с уникальным результатом. Тип запроса в Бюро выводится из Категории риска до оценки Бюро.

Таблица коэффициентов непредвиденных обстоятельств по кредиту

Описание: логика решения **«Коэффициент непредвиденных обстоятельств по кредиту»** определяет полную таблицу решений с уникальным результатом, которая выводит значение «Коэффициент непредвиденных обстоятельств по кредиту» из значения «Категория риска».

Расчет доступности

Описание: логика решения «Расчет доступности» определяет табличную функцию, которая выводит значение Доступности на основании значений Ежемесячный доход, Ежемесячные погашения, Ежемесячные расходы и Требуемый размер ежемесячного платежа. На одном из шагов данного вычисления выводится коэффициент непредвиденных обстоятельств по кредиту путем вызова модели бизнес-знаний для таблицы «Коэффициент непредвиденных обстоятельств по кредиту»;

Таблица категории риска до оценки Бюро

Описание: логика ращения «Таблица категории риска до оценки Бюро» определяет полную таблицу решений с уникальным результатом. Значение «Категория риска до оценки Бюро» выводится из значений «Существующий клиент» и «Степень риска заявления»

Таблица категории риска после оценки Бюро

Описание: логика решения «Таблица категории риска после оценки Бюро» определяет полную таблицу решений с уникальным результатом. Значение «Категория риска после оценки Бюро» выводится из значений «Существующий клиент», «Степень риска заявления» и «Кредитная оценка».

Модель степени риска заявления

Описание: логика решения «Модель степени риска заявления» определяет полную, неупорядоченную таблицу с множественными результатами и агрегацией. Значение Степени риска заявления выводится из «Возраста», «Семейного положения» и «Статуса занятости» в виде суммы значений всех подходящих строк (следовательно, это прогнозируемая система показателей, представленная в виде таблицы решений).

Определение размера платежа

Описание: логика решения «Определение размера платежа» определяет табличную функцию, которая выводит размер ежемесячного платежа из значений Тип продукта, Ставка, Срок и Сумма.

Функция РПП

Описание: Стандартное вычисление размера ежемесячного платежа на основании Ставки, Срока и Суммы.

11.1.3.3 Бизнес-контекст

Наряду со всей прочей информацией, представленной на ДТР, вы можете указать бизнес-контекст принятия решения. Например, можно определить показатели, используемые для отслеживания эффективности принятия решений; цели, которые организация стремится достичь с помощью своего подхода к принятию решений; организационные подразделения, которые принимают решения или владеют подходом к принятию решений. Решения имеют перекрестную ссылку на показатели эффективности и цели, на которые они влияют, и на организационные единицы, которые либо принимают решение, либо определяют, как решение должно быть принято.

Показатели эффективности:

Ежемесячные расходы на Бюро	Общая сумма, взимаемая Бюро за предоставление запрошенных данных при выдаче кредита, за календарный месяц.
Процент одобренных заявок за месяц	Процент одобренных кредитных заявок за календарный месяц.

Процент кредитных заявок в месяц по которым решение было принято автоматически	Процент кредитных заявок за календарный месяц, рассмотрение которых не требовало прямого учаестия сотрудников кредитного отдела.
Объем кредитов, выданных в течение месяца	Общий объем кредитования за календарный месяц.
90% кредитов рассматриваются автоматически	К концу текущего года достигнуть показателей автоматического рассмотрения кредитов в 90%.

Решения сопоставляются с показателями эффективности и целями, на которые они влияют, следующим образом:

	Процент утвержденных кредитных заявок за месяц	**Объем кредитов, выданных в течение месяца**	**Ежемесячные расходы на Бюро**	**90% кредитов рассматриваются автоматически**	**Процент кредитных заявок в месяц по которым решение было принято автоматически**
Решение по кредиту	Да	Да			
Степень риска заявки			Да		
Тип запроса в Бюро			Да		
Маршрут	Да	Да		Да	Да
Стратегия	Да	Да		Да	да

Организационные подразделения

Сотрудник кредитного отдела	Физические лица в розничной банковской организации, ответственные за ручное рассмотрение кредитных заявок.
Менеджер по продукту	Организационное подразделение, определяющее ценообразование для кредитных и иных банковских продуктов, а также решающее, как продавать и отслеживать рентабельность данных продуктов.
Группа аналитики кредитного риска	Организационное подразделение, ответственное за создание моделей кредитного риска и использование данных в целях прогнозирования кредитного риска для клиентов и заявителей.
Розничные банковские услуги	Организационное подразделение, ориентированное на продажу банковских продуктов для потребителей
Подразделение кредитных рисков	Подразделение в банке, ответственное за определение стратегий и политик кредитного риска и предоставление инструментов для предотвращения таких рисков.

Сотрудники кредитного отдела, вероятно, будут частью подразделения по предоставлению розничных банковских услуг, группа аналитики кредитного риска и менеджер по управлению рисками входят в состав подразделения кредитных рисков, однако эти взаимоотношения не управляются DMN.

Эти организационные подразделения являются владельцами решений, принимают решения и владеют источниками знаний следующим образом:

	Владельцы решений	Принимают решения	Источники знаний
Сотрудник кредитного отдела		Решение по заявке	Опыт сотрудника кредитного отдела
Менеджер по продукту	Степень риска заявки		Аналитические данные по кредитным рискам
Подразделение кредитных рисков	Принятие решения Типа запроса в Бюро Соответствие критериям Категория риска до запроса в Бюро Категория риска после запроса в Бюро Маршрут Стратегия		Стратегия управления рисками
Менеджер по продукту	Доступность до запроса в Бюро Доступность после запроса в Бюро Размер требуемого ежемесячного платежа		

11.1.3.4 Службы принятия решений

Две службы решений, которые являются обязательными для модели бизнес-процесса, определяются для модели принятия решения. **Служба принятия решения «Стратегия Бюро»**, вызываемая задачей **«Определить стратегию Бюро»**, на выходе дает следующие решения: {Тип запроса в Бюро, Стратегия}, см. рис. 11.7. **Служба решения «Маршрут»**, вызываемая задачей «**Определить маршрут**», на выходе дает решение {Маршрут}, см. рис. 11.8.

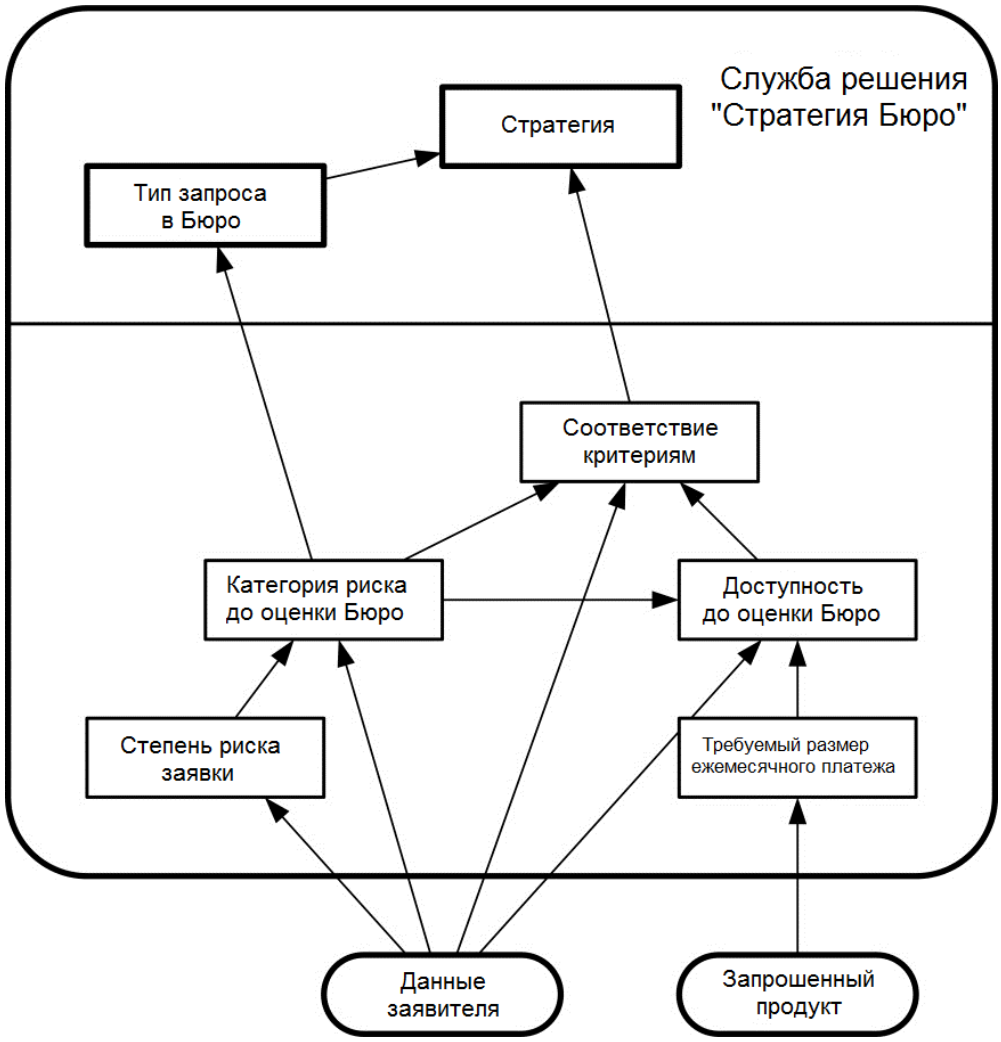


Рисунок 11.7. Служба решения «Стратегия Бюро»

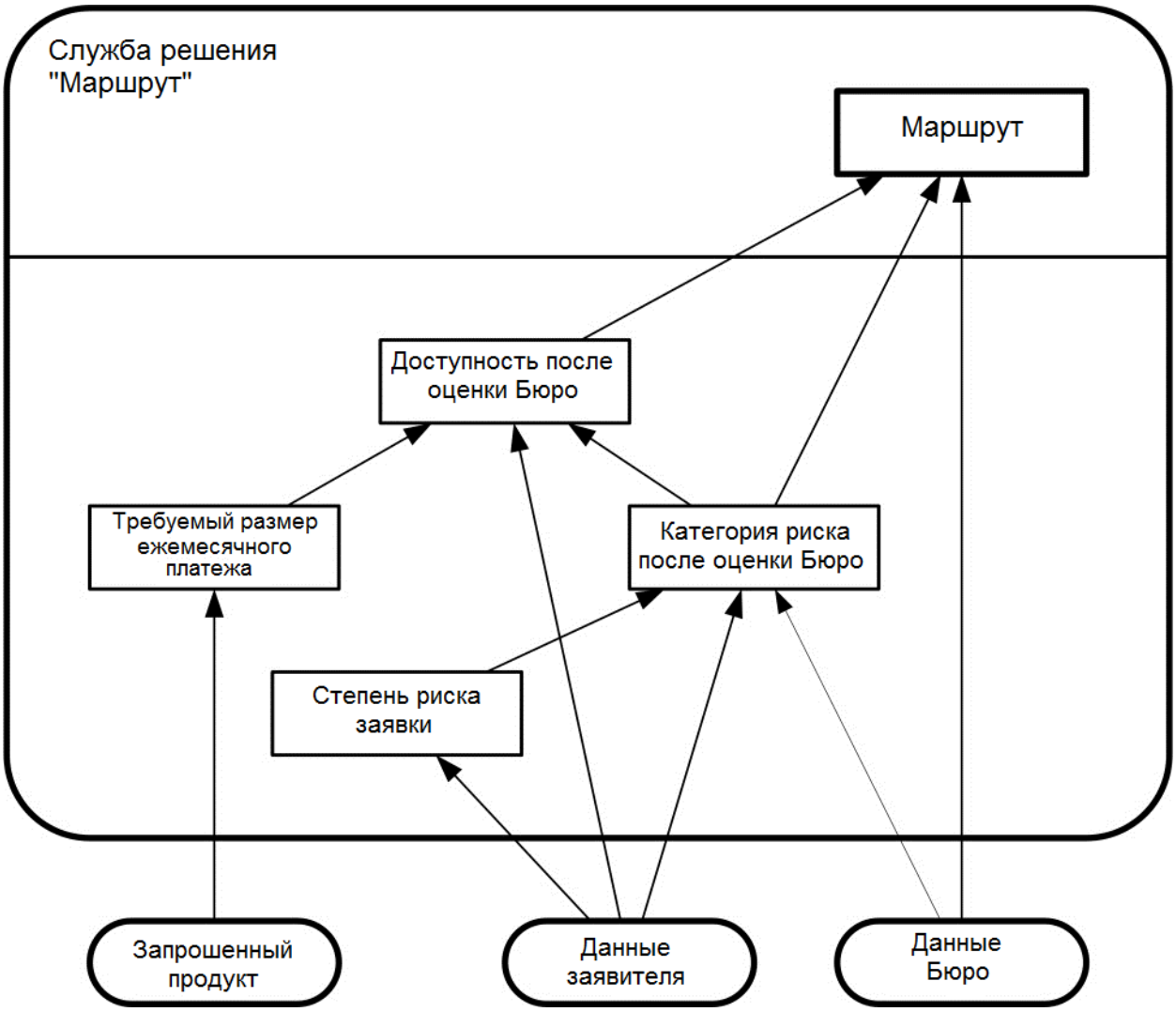


Рисунок 11.8. Служба решения «Маршрут»

11.1.4 Уровень логики решения

ГТР на рисунке 11.2 более подробно описывается в следующих спецификациях выражений значений, связанных с решениями и моделями бизнес-знаний:

- логика решения «**Стратегия**» (рис. 11.8) определяет полную таблицу решений с уникальным результатом. Значение «**Стратегия**» выводится из значений Соответствия критериям и Типа запроса в Бюро;
- логика решения **«Тип запроса в Бюро»** (изображенная в виде вызова, см. рис. 11.9) вызывает таблицу Тип запроса в Бюро, передавая результат решения «Категория риска до оценки Бюро» в виде параметра «Категория риска до оценки Бюро»;
- логика решения **«Таблицы типа запроса в Бюро»** (рис. 11.10) определяет полную таблицу решений с уникальным результатом. Тип запроса в Бюро выводится из Категории риска до оценки Бюро;
- логика принятия решения «**Соответствие критериям**» (изображенная в виде табличного вызова

на рис. 11.11), передавая Данные заявителя, вызывает модель бизнес-знаний для Правил соответствия критериям. Возраст передается как параметр «Возраст», результат решения по категории риска до оценки Бюро – как параметр «Категория риска до оценки бюро», а результат решения о доступности до оценки бюро – как параметр «Доступность до оценки Бюро»;

- логика решения «**Правила соответствия критериям»** (рис. 11.12) определяет полную, упорядоченную в соответствии с приоритетом, таблицу решений с одним результатом. Значение Соответствия критериям выводится из категории риска до оценки бюро, доступности до оценки бюро и возраста;
- логика решения «**Категория риска до оценки Бюро**» (изображенная в виде табличного вызова на рис. 11.13), передавая Данные заявителя, вызывает модель бизнес-знаний для таблицы, определяющей категорию риска до оценки Бюро. Данные передаются следующим образом: Данные заявителя. СуществующийКлиент (Applicant data. ExistingCustomer) передаются как параметр «Существующий клиент», результат решения «Степень риска заявления» – как параметр «Степень риска заявления»;
- логика ращения «**Таблица категории риска до оценки Бюро**» (рисунок 11.14) определяет полную таблицу решений с уникальным результатом. Значение «Категория риска до оценки Бюро» выводится из значений «Существующий клиент» и «Степень риска заявления»;
- логика решения «**Степень риска заявления**» (изображенная в виде табличного вызова на рис. 11.15), передавая Данные заявителя, вызывает модель бизнес-знаний для модели степени риска заявления. Данные передаются следующим образом: возраст передается как параметр «Возраст», Данные заявителя. СемейноеПоложение (Applicant data . MaritalStatus) – как параметр «Семейное положение», Данные заявителя, а СтатусЗанятости (Applicant data . EmploymentStatus) – как параметр «Статус занятости»;
- логика решения «**Модель степени риска заявления**» (рис. 11.16) определяет полную, неупорядоченную таблицу с множественными результатами и агрегацией. Значение Степени риска заявления выводится из «Возраста», «Семейного положения» и «Статуса занятости» в виде суммы значений всех подходящих строк (следовательно, это прогнозируемая система показателей, представленная в виде таблицы решений);
- логика решения «**Маршрут**» (изображенная в виде табличного вызова на рис. 11.17), передавая Данные бюро, вызывает модель бизнес-знаний для Правил маршрутизации. Данные передаются следующим образом: Банкрот передается как параметр «Банкрот», Данные Бюро. КредитнаяОценка (Bureau data . CreditScore) – как параметр «Кредитная оценка», результат решения «Категория риска до оценки Бюро» – как параметр «Категория риска до оценки Бюро», а результат решения «Доступность после оценки Бюро» – как параметр «Доступность после оценки Бюро». Обратите внимание, что, если Данные Бюро равны null (в виду выбора стратегии «ЧЕРЕЗ», которая предполагает пропуск задачи Сбора данных бюро), параметры «Банкрот» и «Кредитная оценка» будут равны null;
- логика решения «**Правила маршрутизации**» (рисунок 11.18) определяет полную, упорядоченную в соответствии с приоритетом, таблицу решений с одним результатом. Значение «Маршрут» выводится из категории риска после оценки бюро, доступности после оценки бюро, параметров Банкрот и Кредитная оценка;
- логика решения «**Категория риска после оценки Бюро**» (изображенная в виде табличного вызова на рис. 11.19), передавая Данные заявителя, вызывает модель бизнес-знаний для таблицы, определяющей категорию риска до оценки Бюро. Данные передаются следующим образом: Данные заявителя. СуществующийКлиент (Applicant data . ExistingCustomer) передается как параметр «Существующий клиент», КредитнаяОценка (Bureau data . CreditScore) – как параметр «Кредитная оценка», результат решения «Степень риска заявления» - как параметр «Степень риска заявления». Обратите внимание, что, если Данные Бюро равны null (в виду выбора

стратегии «ЧЕРЕЗ», которая предполагает пропуск задачи Сбора данных бюро), параметр «Кредитная оценка» будет равны null;

- логика решения «**Таблица категории риска после оценки Бюро**» (рисунок 11.20) определяет полную таблицу решений с уникальным результатом. Значение «Категория риска после оценки Бюро» выводится из значений «Существующий клиент», «Степень риска заявления» и «Кредитная оценка»;
- логика решения «**Доступность до оценки Бюро**» (изображенная в виде табличного вызова на рис. 11.21) вызывает модель бизнес-знаний для расчета Доступности, при этом данные передаются следующим образом: Данные заявителя. Ежемесячно. Доход (Applicant data.Monthly.Income) как параметр «Ежемесячный доход», Данные заявителя.Ежемесячно.Погашение (Applicant data.Monthly.Repayments) – как «Ежемесячные погашения», Данные заявителя.Ежемесячно.Расходы (Applicant data.Monthly.Expenses) как параметр «Ежемесячные расходы», результат принятия решения по категории риска до оценки Бюро передается как параметр «Категория риска», а результат решения «Требуемый размер ежемесячного платежа» – как параметр «Требуемый размер ежемесячного платежа»;
- логика решения «**Доступность после оценки Бюро**» (изображенная в виде табличного вызова на рис. 11.22) вызывает модель бизнес-знаний для расчета Доступности, при этом данные передаются следующим образом: Данные заявителя. Ежемесячно. Доход (Applicant data.Monthly.Income) как параметр «Ежемесячный доход», Данные заявителя.Ежемесячно.Погашение (Applicant data.Monthly.Repayments) – как «Ежемесячные погашения», Данные заявителя.Ежемесячно.Расходы (Applicant data.Monthly.Expenses) как параметр «Ежемесячные расходы», результат принятия решения по категории риска после оценки Бюро передается как параметр «Категория риска», а результат решения «Требуемый размер ежемесячного платежа» – как параметр «Требуемый размер ежемесячного платежа»;
- логика решения **«Расчет доступности»** (рисунок 11.23) определяет табличную функцию, которая выводит значение Доступности на основании значений Ежемесячный доход, Ежемесячные погашения, Ежемесячные расходы и Требуемый размер ежемесячного платежа. На одном из шагов данного вычисления выводится коэффициент непредвиденных обстоятельств по кредиту путем вызова модели бизнес-знаний для таблицы «Коэффициент непредвиденных обстоятельств по кредиту», при этом результат решения по категории риска передается как параметр «Категория риска»;
- логика решения **«Коэффициент непредвиденных обстоятельств по кредиту»** (рисунок 11.24) определяет полную таблицу решений с уникальным результатом, которая выводит значение «Коэффициент непредвиденных обстоятельств по кредиту» из значения «Категория риска»;
- логика решения **«Требуемый размер ежемесячного платежа»** (изображенная в виде табличного вызова на рис. 11.25) вызывает модель бизнес-знаний для «Определение размера платежа», при этом данные передаются следующим образом: Запрошенный продукт.Тип продукта (Requested product. ProductType) передается как параметр «Тип продукта», Запрошенный продукт. Ставка (Requested product . Rate) – как параметр «Ставка», Запрошенный продукт.Срок (Requested product.Term) – как параметр «Срок», и Запрошенный продукт.Сумма как параметр «Сумма»;
- логика решения «**Определение размера платежа**» (рисунок 11.26) определяет табличную функцию, которая выводит размер ежемесячного платежа из значений Тип продукта, Ставка, Срок и Сумма. На одном из шагов данного вычисления вызывается внешняя функция РПП (размер периодического платежа), импортируемая из файла DMN XML. На рисунке 11.27 показана логика функции РПП.

Стратегия

U	Соответствие критериям	Тип запроса в Бюро	Стратегия
	СООТВЕТСТВУЕТ, НЕ СООТВЕТСТВУЕТ	*ЗАПРОС ВСЕХ ДАННЫХ, УПРОЩЕННЫЙ, БЕЗ ЗАПРОСА*	*ОТКАЗАТЬ, БЮРО, ЧЕРЕЗ*
1	*НЕ СООТВЕТСТВУЕТ*	–	*ОТКАЗАТЬ*
2	*СООТВЕТСТВУЕТ*	*ЗАПРОС ВСЕХ ДАННЫХ, УПРОЩЕННЫЙ*	*БЮРО*
3		*БЕЗ ЗАПРОСА*	*ЧЕРЕЗ*

Рисунок 11.9. Логика решения «Стратегия»

Тип запроса в Бюро

Таблица «Тип запроса в Бюро»

Категория риска до оценки Бюро	`Категория риска до оценки Бюро`

Рисунок 11.10: Логика решения «Тип запроса в Бюро»

Таблица «Тип запроса в Бюро»

U	Категория риска до оценки Бюро	Тип запроса в Бюро
	ВЫСОКИЙ, СРЕДНИЙ, НИЗКИЙ, ОЧЕНЬ НИЗКИЙ, ОТКАЗАТЬ	*ЗАПРОС ВСЕХ ДАННЫХ, УПРОЩЕННЫЙ, БЕЗ ЗАПРОСА*
1	*ВЫСОКИЙ, СРЕДНИЙ*	*ЗАПРОС ВСЕХ ДАННЫХ*
2	*НИЗКИЙ*	*УПРОЩЕННЫЙ*
3	*ОЧЕНЬ НИЗКИЙ, ОТКАЗАТЬ*	*БЕЗ ЗАПРОСА*

Рисунок 11.11: Логика решения для таблицы «Тип запроса в Бюро»

Соответствие критериям

Правила соответствия критериям	
Возраст	`Данные заявителя. Возраст (Applicant data. Age)`
Категория риска до оценки Бюро	`Категория риска до оценки Бюро`
Доступность до оценки Бюро	`Доступность до оценки Бюро`

Рисунок 11.12: Логика решения «Соответствие критериям»

Правила соответствия критериям

P	Категория риска до оценки Бюро	Доступность до оценки Бюро	Возраст	Соответствие критериям
	ОТКАЗАТЬ, ВЫСОКИЙ, СРЕДНИЙ, НИЗКИЙ, ОЧЕНЬ НИЗКИЙ			НЕ СООТВЕТСТВУЕТ, СООТВЕТСТВУЕТ
1	ОТКАЗАТЬ	–	–	НЕ СООТВЕТСТВУЕТ
2	–	false	–	НЕ СООТВЕТСТВУЕТ
3	–	–	< 18	НЕ СООТВЕТСТВУЕТ
4	–	–	–	СООТВЕТСТВУЕТ

Рисунок 11.13: Логика решения для Правил соответствия критериям

Категория риска до оценки Бюро

Таблица категории риска до оценки Бюро	
Существующий клиент	`Данные заявителя. Существующий клиент Applicant data. ExistingCustomer`
Степень риска заявления	`Степень риска заявления`

Рисунок 11.14: Логика решения «Категория риска до оценки»

Таблица Категория риска до оценки Бюро

U	Существующий клиент	Степень риска заявки	Таблица Категория риска до оценки Бюро
			ОТКАЗАТЬ, ВЫСОКИЙ, СРЕДНИЙ, НИЗКИЙ, ОЧЕНЬ НИЗКИЙ
1	false	< 100	*ВЫСОКИЙ*
2		[100..120)	*СРЕДНИЙ*
3		[120..130]	*НИЗКИЙ*
4		> 130	*ОЧЕНЬ НИЗКИЙ*
5	true	< 80	*ОТКАЗАТЬ*
6		[80..90)	*ВЫСОКИЙ*
7		[90..110]	*СРЕДНИЙ*
8		> 110	*НИЗКИЙ*

Рисунок 11.15: Логика решения для таблицы «Категория риска до оценки Бюро»

Степень риска заявления

Модель «Степень риска заявления»

Возраст	`Данные заявителя. Возраст` `Applicant data . Age`
Семейное положение	`Данные заявителя. Семейное положение` `Applicant data . MaritalStatus`
Статус занятости	`Данные заявителя. Статус занятости` `Applicant data . EmploymentStatus`

Рисунок 11.16: Логика решения «Степень риска заявления»

Модель «Степень риска заявления»

C+	Возраст	Семейное положение	Статус занятости	Частичная Оценка
	[18..120]	*S, M*	*БЕЗРАБОТНЫЙ, ТРУДОУСТРОЕННЫЙ, ИНДИВИДУАЛЬНЫЙ ПРЕДПРИНИМАТЕЛЬ, СТУДЕНТ*	
1	[18..21]	–	–	32
2	[22..25]	–	–	35
3	[26..35]	–	–	40
4	[36..49]	–	–	43
5	>=50	–	–	48
6	–	*S*	–	25
7	–	*M*	–	45
8	–	–	*БЕЗРАБОТНЫЙ*	15
9	–	–	*СТУДЕНТ*	18
10	–	–	*ТРУДОУСТРОЕННЫЙ*	45
11	–	–	*ИНДИВИДУАЛЬНЫЙ ПРЕДПРИНИМАТЕЛЬ*	36

Рисунок 11.17: Логика решения для модели «Степень риска заявления»

Маршрут	
Правила маршрутизации	
Банкрот	`Данные бюро. Банкрот` `Bureau data . Bankrupt`
Кредитная оценка	`Данные бюро. Кредитная оценка` `Bureau data . CreditScore`
Категория риска после оценки Бюро	`Категория риска после оценки Бюро`
Доступность после оценки Бюро	`Доступность после оценки Бюро`

Рисунок 11.18: Логика решения «Маршрут»

Правила маршрутизации

P	Категория риска после оценки Бюро	Доступность после оценки Бюро	Банкрот	Кредитная оценка	Маршрут
	ОТКАЗАТЬ, ВЫСОКИЙ, СРЕДНИЙ, НИЗКИЙ, ОЧЕНЬ НИЗКИЙ			*null, [0..999]*	*ОТКАЗАТЬ, НА РАССМОТРЕНИЕ, УТВЕРДИТЬ*
1	–	false	–	–	*ОТКАЗАТЬ*
2	–	–	true	–	*ОТКАЗАТЬ*
3	*ВЫСОКИЙ*	–	–	–	*НА РАССМОТРЕНИЕ*
4	–	–	–	< 580	*НА РАССМОТРЕНИЕ*
5	–	–	–	–	*УТВЕРДИТЬ*

Рисунок 11.19: Логика решения для Правил маршрутизации

Категория риска после оценки Бюро	
Таблица «Категория риска после оценки Бюро»	
Существующий клиент	Данные заявителя. Существующий клиент Applicant data . ExistingCustomer
Кредитная оценка	Данные бюро. Кредитная оценка Bureau data . CreditScore
Степень риска заявления	Степень риска заявления

Рисунок 11.20: Логика решения «Категория риска после оценки Бюро»

Таблица Категория риска после оценки Бюро

U	Существующий клиент	Степень риска заявления	Кредитная оценка	Категория риска после оценки Бюро
				ОТКАЗАТЬ, ВЫСОКИЙ, СРЕДНИЙ, НИЗКИЙ, ОЧЕНЬ НИЗКИЙ
1	false	< 120	< 590	*ВЫСОКИЙ*
2			[590..610]	*СРЕДНИЙ*
3			> 610	*НИЗКИЙ*
4		[120..130]	< 600	*ВЫСОКИЙ*
5			[600..625]	*СРЕДНИЙ*
6			> 625	*НИЗКИЙ*
7		> 130	–	*ОЧЕНЬ НИЗКИЙ*
8	true	<= 100	< 580	*ВЫСОКИЙ*
9			[580..600]	*СРЕДНИЙ*
10			> 600	*НИЗКИЙ*
11		> 100	< 590	*ВЫСОКИЙ*
12			[590..615]	*СРЕДНИЙ*
13			> 615	*НИЗКИЙ*

Рисунок 11.21: Логика решения для таблиц «Категория риска после оценки Бюро»

Доступность до оценки Бюро	
Расчет доступности	
Ежемесячный доход	`Данные заявителя. Ежемесячно . Доход` `Applicant data . Monthly . Income`
Ежемесячные выплаты	`Данные заявителя. Ежемесячно . Выплаты` `Applicant data . Monthly . Repayments`
Ежемесячные расходы	`Данные заявителя. Ежемесячно . Расходы` `Applicant data . Monthly . Expenses`
Категория риска	`Категория риска до оценки Бюро`
Требуемый размер ежемесячного платежа	`Требуемый размер ежемесячного платежа`

Рисунок 11.22: Логика решения Доступность до оценки Бюро

Доступность после оценки Бюро	
Расчет доступности	
Ежемесячный доход	`Данные заявителя. Ежемесячно . Доход` `Applicant data . Monthly . Income`
Ежемесячные выплаты	`Данные заявителя. Ежемесячно . Выплаты` `Applicant data . Monthly . Repayments`
Ежемесячные расходы	`Данные заявителя. Ежемесячно . Расходы` `Applicant data . Monthly . Expenses`
Категория риска	`Категория риска после оценки Бюро`
Требуемый размер ежемесячного платежа	`Требуемый размер ежемесячного платежа`

Рисунок 11.23: Логика решения «Доступность после оценки Бюро»

Расчет доступности

F	(Ежемесячный доход, Ежемесячные выплаты, Ежемесячные расходы, Категория риска, Требуемый размер ежемесячного платежа)
Чистый доход	Ежемесячный доход – (Ежемесячные выплаты + Ежемесячные расходы)
Таблица коэффициентов непредвиденных обстоятельств по кредиту	Таблица коэффициентов непредвиденных обстоятельств по кредиту
	Категория риска: Категория риска
Доступность	if Чистый доход * Коэффициенте непредвиденных обстоятельств по кредиту> Требуемый размер ежемесячного платежа then true else false
Доступность	

Рисунок 11.24: Логика решения «Расчет доступности

Таблица коэффициентов непредвиденных обстоятельств по кредиту

U	Категория риска	Коэффициенте непредвиденных обстоятельств по кредиту
	ОТКАЗАТЬ, ВЫСОКИЙ, СРЕДНИЙ, НИЗКИЙ, ОЧЕНЬ НИЗКИЙ	
1	*ВЫСОКИЙ, ОТКАЗАТЬ*	0.6
2	*СРЕДНИЙ*	0.7
3	*НИЗКИЙ, ОЧЕНЬ НИЗКИЙ*	0.8

Рисунок 11.25: Логика решения для таблицы коэффициентов непредвиденных обстоятельств по кредиту

Требуемый размер ежемесячного платежа

Определение размера платежа

Тип продукта	Запрошенный продукт. ТипПродукта Requested product . ProductType
Ставка	Запрошенный продукт. Ставка Requested product . Rate
Срок	Запрошенный продукт. Срок Requested product . Term
Сумма	Запрошенный продукт. Сумма Requested product . Amount

Рисунок 11.26: Логика решения «Требуемый размер ежемесячного платежа»

Определение размера платежа	
F	(Тип продукта, Ставка, Срок, Сумма)
Ежемесячная оплата за обслуживание счета	if Тип продукта = СТАНДАРТНЫЙ КРЕДИТ” then 20.00 else if Тип продукта = “СПЕЦИАЛЬНЫЙ КРЕДИТ” then 25.00 else null
Ежемесячное погашение	РПП (Ставка, Срок, Сумма)
Ежемесячное погашение + Ежемесячная оплата за обслуживание счета	

Рисунок 11.27: Логика решения «Определение размера платежа»

Функция РПП	
F	(Ставка, Срок, Сумма)
(Сумма * Ставка/12)/(1 – (1 + Ставка/12)** -Срок)	

Рисунок 11.28: Логика решения «Функция РПП»

11.1.5 Исполнение модели решения

Для исполнения модели решения (в данном случае путем вызова двух служб принятия решений) данные кейса должны быть связаны с входными данными, точно так же, как вызов связывает аргументы с параметрами функции. Однако привязка данных кейса к входным данным не является частью модели принятия решений, в отличие от вызова, который указывает, каким образом входы требования к принятию решения связаны с параметрами требуемых знаний этого решения.

FEEL позволяет использовать контексты и другие выражения для представления данных кейса (см. также разделы 10.3.3 «Данные XML» и 10.6.1 «Контекст»). Входные данные связаны с определением элемента (7.3.2 «Метамодель ItemDefinition»), а данные кейса должны иметь одинаковые типы и другие ограничения, указанные в определении элемента. Данные кейса должны быть сопоставлены с доменом FEEL. Например, данные экземпляра XML сопоставляются с доменом FEEL, как описано в разделе 10.3.3 «Данные XML».

Для удобства восприятия, в примерах данные кейса указываются в виде табличных выражений, вместо XML. На рис. 11.28, рис. 11.29 и на рис. 11.30 показаны табличные контексты, определяющие данные кейса для Данных заявителя, Запрошенного продукта и Данных Бюро.

Данные заявителя		
Возраст	51	
Семейное положение MaritalStatus	*ЖЕНАТ*	
Статус занятости EmploymentStatus	*ТРУДОУСТРОЕН*	
Существующий клиент ExistingCustomer	false	
Ежемесячно	Доходы	10,000.00
	Выплаты	2,500.00
	Расходы	3,000.00

Рисунок 11.29: Пример входных данных для Данных заявителя

Данные бюро	
Банкрот	false
КредитнаяОценка CreditScore	600

Рисунок 11.30: Пример входных данных для Запрошенного продукта

Стратегия	ЧЕРЕЗ
Тип запроса в Бюро	«БЕЗ ЗАПРОСА»

Рисунок 11.31: Пример входных данных для Данных Бюро

При вызове службы решения «Стратегия Бюро» с данными кейса для «Данные заявителя» и «Запрошенный продукт» возвращается контекст, показанный на рисунке 11.32:

Маршрут	*ПРИНЯТЬ*

Рисунок 11.32: Результат службы решения «Стратегия Бюро»

При вызове службы решения «Маршрут» с данными кейса для «Данные заявителя», «Запрошенный продукт» и «Данные Бюро» возвращается контекст, показанный на рисунке 11.33.

Запрошенный продукт	
ТипПродукта ProductType	*СТАНДАРТНЫЙ КРЕДИТ*
Ставка	0.08
Срок	36
Сумма	100,000.00

Рисунок 11.33: Результат службы решения «Маршрут»

11.2 Пример 2: Ранжирование кредитных продуктов

Во втором примере рассмотрим соответствие заёмщика критериям получения различных продуктов ипотечного кредитования на основе доходов, активов, обязательств и кредитной оценки Заемщика, и ранжирование этих продуктов согласно заданным условиям сортировки. Пример иллюстрирует большое разнообразие типов выражений DMN, включая контекст, вызов, отношение и определение функции, а также некоторые новые функции и операторы FEEL, включая импорт, вызов службы, расширенную итерацию, обобщенные унарные тесты и привязку Java. Представленная здесь логика является лишь одним из множества различных способов моделирования сценария.

ДТР для модели принятия решений показана на рисунке 11.34.

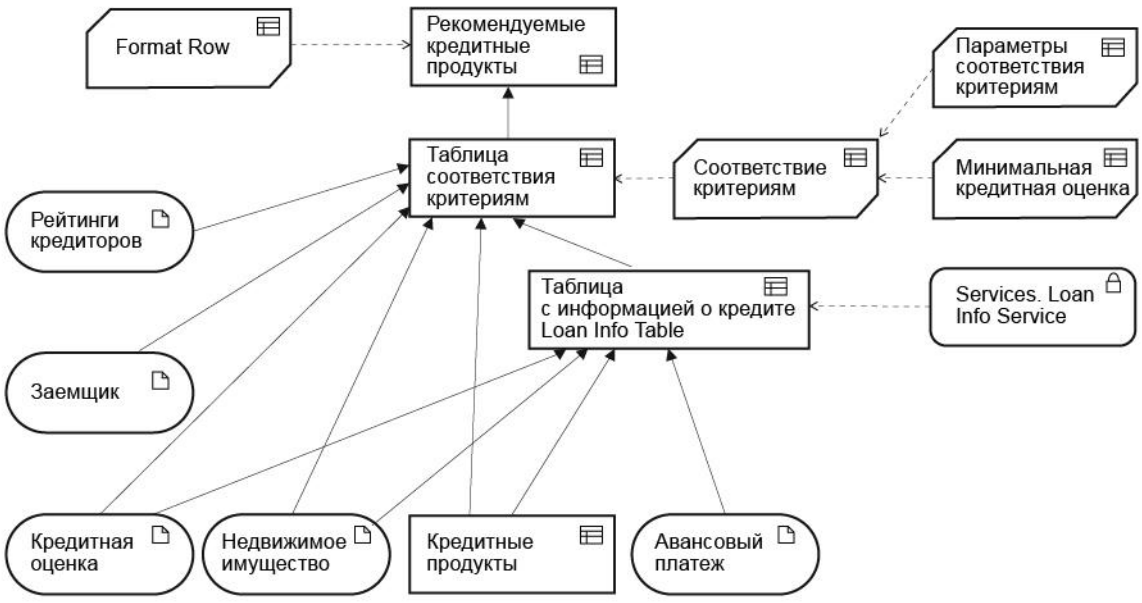


Рис. 11.34: ДТР для рекомендованных кредитных продуктов

Элементы входных данных включают в себя:

- **Кредитная оценка** – число от 300 до 850 включительно
- **Авансовый платеж** – число
- **Недвижимое имущество** – структура типа *tProperty* (рисунок 11.35)
- **Заемщик** – структура типа *tBorrower* (рисунок 11.37), и
- **Рейтинги кредиторов** – структура типа *tLenderRatings* (рисунок 11.38)

Формат табличного выражения для определений типов данных на Рисунке 11.35, Рисунке 11.37 и Рисунке 11.38 не является нормативным. Например, рисунок 11.35 представляет собой визуализацию XML представления, которое приводится на рисунке 11.36.

tProperty	1	Address (Адрес)	1	Street (Улица)	Text
			2	Unit (Дом, строение, корпус)	Text
			3	City (Город)	Text
			4	State (Республика, край область)	Text
			5	ZIP (Почтовый индекс)	Text
	2	Purchase Price (Цена покупки)	Number		
	3	Monthly Tax Payment (Размер ежемесячного налогового платежа)	Number		
	4	Monthly Insurance Payment (Размер ежемесячного страхового платежа)	Number		
	5	Monthly HOA Condo Fee (Ежемесячная плата за квартиру в ТСЖ)	Number		

Рисунок 11.35: Тип tProperty (ненормативное представление)

```
<semantic:itemDefinition name="tProperty" label="tProperty">
  <semantic:itemComponent id="_5e820b14-1f14-44e2-bee1-a35fbedc477f" name="Address">
    <semantic:itemComponent id="_d40919e3-168d-46dc-a7da-ccefeead8a49" name="Street">
      <semantic:typeRef>string</semantic:typeRef>
    </semantic:itemComponent>
    <semantic:itemComponent id="_a03ae467-fb6a-46f0-ab1a-dc0992d81095" name="Unit">
      <semantic:typeRef>string</semantic:typeRef>
    </semantic:itemComponent>
    <semantic:itemComponent id="_f902cd87-2c95-4b90-95cf-a2c6b1b87a1e" name="City">
      <semantic:typeRef>string</semantic:typeRef>
    </semantic:itemComponent>
    <semantic:itemComponent id="_97f12b0d-be5c-4d42-abb5-d565599fee87" name="State">
      <semantic:typeRef>string</semantic:typeRef>
    </semantic:itemComponent>
    <semantic:itemComponent id="_2fdc92bc-55da-4ff7-8a9d-c5213b69a0a8" name="ZIP">
      <semantic:typeRef>string</semantic:typeRef>
    </semantic:itemComponent>
  </semantic:itemComponent>
  <semantic:itemComponent id="_cc0e8c3f-ae44-4080-88db-555d8a2f8560" name="Purchase Price">
    <semantic:typeRef>number</semantic:typeRef>
  </semantic:itemComponent>
  <semantic:itemComponent id="_ce17ee0b-f1e1-43cf-8a5e-4a18390fc6d6" name="Monthly Tax Payment">
    <semantic:typeRef>number</semantic:typeRef>
  </semantic:itemComponent>
  <semantic:itemComponent id="_338c3f84-8ff7-404d-9b61-d211a5cebe4b" name="Monthly Insurance Payment">
    <semantic:typeRef>number</semantic:typeRef>
  </semantic:itemComponent>
  <semantic:itemComponent id="_fe427d63-1cf3-4d2d-b268-f7e01dccad59" name="Monthly HOA Condo Fee">
    <semantic:typeRef>number</semantic:typeRef>
  </semantic:itemComponent>
</semantic:itemDefinition>
```

Рисунок 11.36: Тип tProperty (представление XML)

tBorrower	1	Full Name (ФИО)	Text		
	2	Tax ID (Налоговый идентификатор)	Text		
	3	Employment Income (Доходы от трудовой деятельности)	Number		
	4	Other Income (Другие доходы)	Number		
	5	Assets (Активы) *tAssets* *tAsset*	1	Type (Тип)	tAssetType *Text* *“Checking Savings Brokerage account”, “Real Estate”, “Other Liquid”, “Other Non-Liquid”* *(«Сберегательный брокерский счет», «Недвижимость», «Прочие ликвидные активы», «Прочие неликвидные активы»)*
			2	Institution Account or Description (Институциональный счет или описание)	Text
			3	Value (Значение)	Number
	6	Liabilities (Пассивы) *tLiabilities* *tLiability*	1	Type (Тип)	tLiabilityType *Text* *“Credit card”, “Auto loan”, “Student loan”, “Lease”, “Lien”, “Real estate loan”, “Alimony child support”, “Other”* *(«Кредитная карта», «Автокредит», «Кредит на оплату обучения», «Аренда», «Обременение», «Кредит на недвижимость», «Алименты на содержание ребенка», «Другое»)*
			2	Payee (Получатель платежа)	Text
			3	Monthly payment (Размер ежемесячной выплаты)	Number
			4	Balance (Баланс)	Number
			5	To be paid off (К списанию)	Boolean

Рисунок 11.37: Тип tBorrower

tLendersRatings ||| *tLendersRating*	1	Lender Name (Наименование кредитора)	Text
	2	Customer Rating (Рейтинг клиента)	Number *[1…5]*

Рисунок 11.38: Тип tLenderRatings, коллекция tLenderRating

Кроме того, решение «Кредитные продукты» без входных данных и структура типа tLoanProducts являются отношением (рисунок 11.39). Ячейки в отношении – это выражения FEEL, но они часто содержат литеральные значения, что позволяет встроить таблицы статических данных в модель решения.

В этом случае модель решения представляет собой список продуктов ипотечного кредитования, предлагаемых различными кредиторами. В ней указывается наилучшая процентная ставка для заемщиков с низким уровнем риска, затраты на выдачу кредита в процентах от суммы кредита, и комиссионные (постоянное значение).

Кредитные продукты
tLoanProducts

	Наименование кредитора	Наименование продукта	Тип	Лучшая % ставка	Затраты на выдачу кредита, %	Комиссионные	Срок
	Text	*TProductName* *“Fixed30-NoPoints”, “Fixed30-Standard”, “Fixed15-NoPoints”, “Fixed15-Standard”, “ARM5/1-NoPoints”, “ARM5/1- Standard”*	*tAmortizationtype, “Fixed rate”, “Variable rate”*	*tPercent*	*tPercent*	*Number*	*Number*
1	“Кредитор А”	*Fixed30-NoPoints*	*Fixed rate*	3.95	0	1925	360
2	“Кредитор Б”	*Fixed30-Standard*	*Fixed rate*	3.75	0.972	1975	360
3	“Кредитор В”	*Fixed15-NoPoints*	*Fixed rate*	3.625	0	816	180
4	“Кредитор Г”	*Fixed15-Standard*	*Fixed rate*	3.25	0.767	1975	180
5	“Кредитор Д”	*ARM5/1-NoPoints*	*Variable rate*	3.875	0	1776	360
6	“Кредитор Е”	*ARM5/1- Standard*	*Variable rate*	3.625	0.667	1975	360

Рисунок 11.39: Кредитные продукты

tLoanProducts

|||

tLoanProduct

1	Lender Name (Наименование кредитора)	Text
2	Product Name (Наименование продукта)	TProductName *“Fixed30-NoPoints”, “Fixed30-Standard”, “Fixed15-NoPoints”, “Fixed15-Standard”, “ARM5/1-NoPoints”, “ARM5/1- Standard”*
3	Type (Тип)	tAmortizationtype, *“Fixed rate”, “Variable rate”*
4	Best Rate Pct (Лучшая % ставка)	tPercent *Number*
5	Points Pct (Затраты на выдачу кредита в %)	tPercent *Number*
6	Fees amount (Комиссионные)	Number
7	Term (Срок)	Number

Рисунок 11.40: Тип tLoanProducts, коллекция tLoanProduct

Модель «**Рекомендованные кредитные продукты**» импортирует модель решения «**Информация о кредите**» с ДТР (рис. 11.41), которая определяет службу **Loan Info Service**. Чтобы различать пространства имен импортированной и импортируемой моделей, первой присваевается префикс, выбираемый разработчиком модели по своему усмотрению. В данном случае был выбран префикс *Services*. В импортируемой ДТР (рис. 11.34) импортированная служба **Services.Loan Info Service** обозначена ненормативным значком замка. Это означает, что ее логика не может быть отредактирована в импортируемой модели. Параметры службы – это входные данные, показанные на рисунке 11.41: **Кредитная оценка, Недвижимость, Кредитный продукт** и **Авансовый платеж.** Их типы идентичны типам параметров импортируемой модели.

Services.Loan Info Service заполняет строку решения «**Таблица с информацией о продукте»**, представляющую собой коллекцию типа **tLoanInfoRow** (рис. 11.39). Таким образом происходит расчет деталей выбранного кредитного продукта для заданной стоимости недвижимого имущества (покупной цены) и размера авансового платежа.

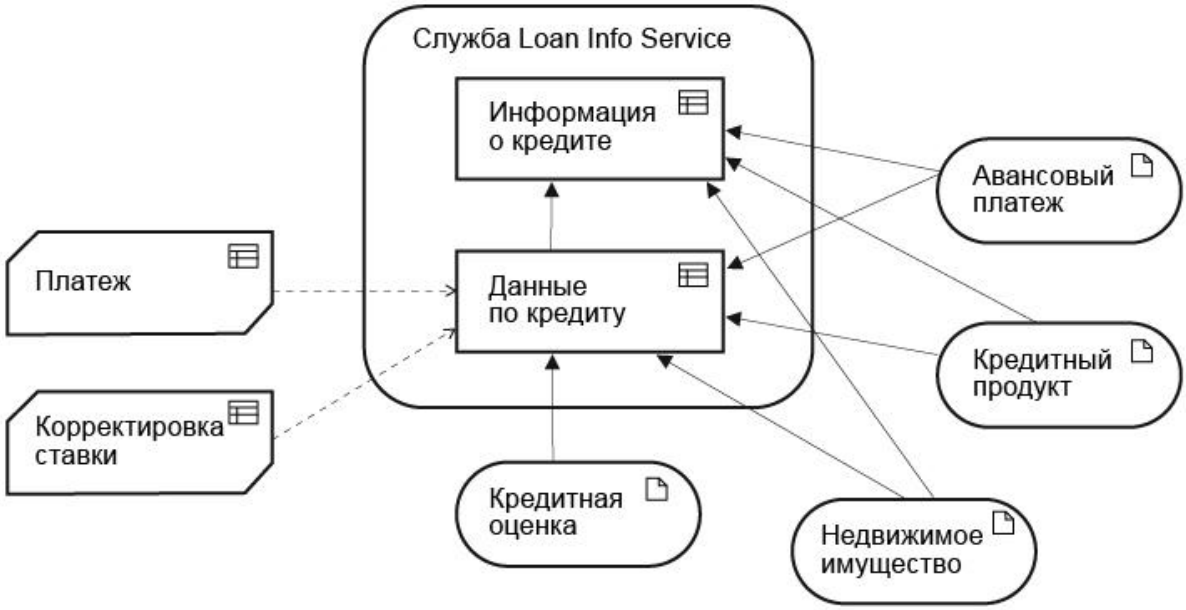


Рисунок 11.41: ДТР импортированной службы Loan Info Service

tLoanInfoTable ||| *tLoanInfoRow*	1	Product Name (Наименование продукта)	TProductName *“Fixed30-NoPoints”, “Fixed30-Standard”, “Fixed15-NoPoints”, “Fixed15-Standard”, “ARM5/1-NoPoints”, “ARM5/1- Standard”*
	2	Amortization Type (Тип амортизации)	tAmortizationtype, *“Fixed rate”, “Variable rate”*
	3	LTV (Отношение размера кредита к стоимости залогового имущества)	tPercent *Number*
	4	Note Amount (Размер кредитного обязательства)	Number
	5	Initial Rate Pct (Начальная ставка, %)	tPercent *Number*
	6	Qualifying Rate Pct (Квалификационная ставка, %)	tPercent *Number*
	7	Initial Monthly Payment (Первоначальный ежемесячный платеж)	Number
	8	Qualifying Monthly Payment (Квалификационный ежемесячный платеж)	Number
	9	Points Amount (Затраты на выдачу кредита)	Number
	10	Fees Amount (Комиссионные)	Number
	11	Funds Toward Purchase (Средства на покупку)	Number
	12	Down Payment (Авансовый платеж)	Number
	13	Closing Costs (Стоимость закрытия)	Number
	14	Cash to Close (Наличные для закрытия)	Number

Рисунок 11.42: Тип tLoanInfoTable, коллекция tLoanInfoRow

Данные по кредиту
tLoanData

1	Points Amount (Затраты на выдачу кредита) *Number*	decimal ((Property.Purchase Price – Down Payment)* Loan Product.Points Ptc/100,2
2	Note Amount (Размер кредитного обязательства) *Number*	Property.Purchase Price – Down Payment + Loan Product.Fees Amount + Points Amount
3	LTV (Отношение размера кредита к стоимости залогового имущества) *tPercent*	decimal (100*Note Amount/Property.Purchase Price, 2)
4	Closing Costs (Стоимость закрытия) *Number*	Decimal(0.02*Note Amount,2)
5	Funds Toward Purchase (Средства на покупку) *Number*	Note Amount-Loan Product.Fees Amount – Closing Costs
6	Interest rate Percent (Процентная ставка) *tPercent*	Loan Product.Best Rate Ptc + Rate Ajustments (Credit Score, LTV)
7	Qualifying Rate Pct (Квалификационная ставка, %) *tPercent*	If Loan Product.Type=”Variable rate” then Interest Rate Percent+2 else Interest Rate Percent
8	Monthly Payment (Ежемесячный платеж) *Number*	payment
		1 — P, *Number* — Note Amount
		2 — r, *Number* — Interest Rate Percent/100
		3 — n, *Number* — Loan Product.Term
9	Qualifying Monthly Payment (Квалификационный ежемесячный платеж) *Number*	payment
		1 — P, *Number* — Note Amount
		2 — r, *Number* — Qualifying Rate Percent/100
		3 — n, *Number* — Loan Product.Term
Результат		

Рисунок 11.43: Данные по кредиту

Внутри службы **«Данные по кредиту»** выполняют вычисления, используемые для представления решения **«Информация о кредите»**. Решение **«Данные по кредиту»** моделируется как контекст без поля окончательного результата. То есть каждая запись контекста создает компонент результата. (Текст «**Результат**» в окне окончательного результата является артефактом инструмента, которого нет в спецификации, и перезаписывается литеральным выражением, если у контекста есть значение в окне окончательного результата.) Несколько замечаний по поводу логики, показанной на рисунке 11.43:

- Арифметика FEEL может прозводить значения со множеством знаков после десятичной точки. Функция decimal(x, 2) округляет значение x до 2 десятичных знаков.
- Запись контекста «Процентная ставка» вызывает модель бизнес-знаний **«Корректировка ставки»** (рис. 11.44), функцию **Кредитной оценки** заемщика и **Отношение размера кредита к стоимости залогового имущества**. Это увеличивает процентную ставку по Кредитному продукту на небольшую сумму в зависимости от

кредитного риска.

- Если **Кредитная оценка** заемщика менее 620, то он не соответствует критериям и не может получить кредит. В этом случае **Корректировка ставки** может возвращать null, но тогда все выражения, использующие **Корректировку ставки**, также будут null, что усложнит логику. Для упрощения дальнейшей логики вычислений предпочтительней вернуть число, поскольку в конечном итоге кредит не будет одобрен, т.к. **Кредитная оценка** меньше 620.
- Для кредитов с переменной процентной ставкой в соотношении долга к доходу используется сумма *Квалификационного платежа*, основанная на процентной ставке, которая на 2 процента выше, чем ставка, использованная в первоначальном *Ежемесячном платеже*.
- *Ежемесячный платеж* и *Квалификационный платеж* смоделированы как табличные вызовы модели бизнес-знаний **«Платеж»,** формула амортизации (рисунок 11.45). Параметрами платежа являются сумма кредита *p*, процентная ставка *r* и срок в месяцах, *n*.

Решение **Информация о кредите** (рисунок 11.46), результат **Services.Loan Info**, возвращает строку решения **«Таблица с информацией о продукте»**. Оно также смоделировано как контекст без поля окончательного результата, что означает, что каждая запись контекста представляет столбец **Таблицы с информацией о продукте**.

Корректировка ставки
tPercent

	Входные данные		Выходные данные
U	**Кредитная оценка**	**Отношение размера кредита к стоимости залогового имущества**	**Корректировка ставки**
	tCreditScore *[300...850]*	*tPercent*	*tPercent*
1	>=660	<=60	0
2	[620..660)	<=60	0.125
3	>=700	<=60	0.125
4	[660..700)	>60	0.125
5	[620..660)	(60..70]	0.25
6	[680..700]	(60..70]	0.25
7	[640..680]	>70	0.375
8	[620..640)	(70..80]	0.375
9	[620..640)	>80	0.5
10	<620	-	0.5

Рисунок 11.44: Корректировка процентной ставки

Платеж *Number*	
F	P r n *Number Number Number*
decimal(p*r/12(1+r/12)**-n),2)	

Рисунок 11.45: Модель бизнес-знаний «Платеж»

Информация о кредите *tLoanInfo*		
1	Product (Продукт) *TProductName* *“Fixed30-NoPoints”, “Fixed30-Standard”, “Fixed15-NoPoints”, “Fixed15-Standard”, “ARM5/1-NoPoints”, “ARM5/1- Standard”*	Loan Product.Name
2	Amortization Type (Тип амортизации) *tAmortizationtype* *“Fixed rate”, “Variable rate”*	Loan Product. Type
3	LTV (Отношение размера кредита к стоимости залогового имущества) tPercent *Number*	Loan Data.LTV
4	Note Amount (Размер кредитного обязательства) *Number*	Loan Data.Note Amount
5	Initial Rate Pct (Начальная ставка, %) *tPercent* *Number*	Loan Data.Interest Rate Percent
6	Qualifying Rate Pct (Квалификационная ставка, %) tPercent *Number*	Loan Data.Qualifying Rate Percent
7	Initial Monthly Payment (Первоначальный ежемесячный платеж) Number	Loan Data.Monthly Payment
8	Qualifying Monthly Payment (Квалификационный ежемесячный платеж) *Number*	Loan Data. Qualifying Payment
9	Points Amount (Затраты на выдачу кредита) *Number*	Loan Data.Points Amount
10	Fees Amount (Комиссионные) *Number*	Loan Product.Fees Amount
11	Funds Toward Purchase (Средства на покупку) *Number*	Loan Data.Funds Toward Purchase
12	Down Payment (Авансовый платеж) *Number*	Down Payment
13	Closing Costs (Стоимость закрытия) *Number*	Loan Data.Closing Costs
14	Cash to Close (Наличные для закрытия) *Number*	Property.Purchase Price - Funds Toward Purchase
Результат		

Рисунок 11.46: Информация о кредите

В импортируемой модели решение «**Таблица с информацией о продукте»** (рисунок 11.47) итерирует вызов «**Информация о кредите»** по строкам решения «**Кредитные продукты**». Оно моделируется как литеральное выражение с использованием оператора FEEL *for..in..return*. Здесь *x* – переменная диапазона, означающая один элемент в списке (один **Кредитный продукт** в **Кредитных продуктах),** производящий аргумент вызова функции.

Таблица с информацией о продукте
tLoanInfoTable

```
for x in Loan Products return Services.Loan Info(x,Down Payment, Property, Credit Score)
```

Рисунок 11.47: Таблица с информацией о продукте

Решение **«Таблица с информацией о продукте»** выводит значения для каждого **Кредитного продукта**. Эти значения используются для проверки соответствия заемщика таким критериям, как доходы, активы, обязательства и кредитный рейтинг.

В основе логики проверки соответствия критериям для каждого конкретного кредита лежит моедль бизнес-знаний **«Минимальная кредитная оценка»** (рис. 11.48). Она представляет собой таблицу решений, в которой на основе трех параметров рассчитывается минимальная оценка, необходимая для одобрения кредита.

Параметры:

1) *Кредитная нагрузка* – отношение долга к доходу заемщика);

2) *Отношение размера кредита к стоимости залогового имущества*;

3) *Резервы* – показатель ликвидных активов заемщика после закрытия кредита в единицах ежемесячных *Расходов на жилье*.

Для таблицы реализована политика выбора *Collect* с агрегацией *Minimum*. Это значит, что результат возвращается тогда, когда несколько правил соответствуют наименьшему значению. Когда *Кредитная нагрузка* превышает 95%, заемщик автоматически квалифицируется, как не соответствующий требованиям на получение кредита. В таком случае ни одно правило не будет применяться к набору входных значений, а **Минимальная кредитная оценка** вернет значение null. Последующая логика, ссылающаяся на эту переменную, должна предусматривать возможность нулевого значения.

Минимальная кредитная оценка *tCreditScore*
[300..850]

	Входные данные			Выходные данные
C<	**Кредитная нагрузка**	**Отношение размера кредита к стоимости залогового имущества**	**Резервы**	**Минимальная кредитная оценка**
	tPercent	*tPercent*	*Number*	*tCreditScore [300…850]*
1	>=36	<=75	>2	620
2	>=36	<=75	>0	640
3	>=36	(75..95]	>6	660
4	>=36	(75..95]	>0	680
5	(36..45]	<=75	>6	660
6	(36..45]	<=75	>0	680
7	(36..45]	(75..95]	>6	700
8	(36..45]	(75..95]	>0	720

Рисунок 11.48: Минимальная кредитная оценка

«Минимальная кредитная оценка» вызывается моделью бизнес-знаний **«Соответствие критериям»**, которая, в свою очередь, вызывает модель бизнес-знаний **«Параметры соответствия критериям»** (рисунок 11.49). С помощью модели **«Параметры соответствия критериям»** рассчитывается два ключевых параметра для вычисления **«Минимальной кредитной оценки»**:

- *Отношение размера кредита к стоимости залогового имущества в %*t, и
- Ликвидные активы после закрытия, называемые *Резервами*.

Обратите внимание, что для записи контекста *Расходы на жилье*, суммирующей платеж по кредиту, налоговые и страховые платежи, а также плату в ТСЖ, необходимо предусмотреть сценарий, при котором последний параметр останется пустым (null) во входных данных **«Недвижимое имущество».** Так происходит потому, что сложение null с числом в результате возвращает null. Чтобы избежать такого сценария, вместо оператора "+" нужно использовать функцию *sum()* для списка, отфильтрованного по условию *item! = null*. Подобный прием может также применяться к записи контекста *Доход*.

Параметры соответствия критериям *tEligibilityParameters*						
F		Loan Product *tLoanProduct*	Borrower *tBorrower*	Loan Info *tLoanInfoRow*	Property *tProperty*	Credit Score *tCreditScore [300..850]*
1	Housing Expenses (Расходы на жилье) *Number*	sum([Loan Info.Qualifying Monthly Payment, Poperty.Monthly Tax Payment, Property.Monthly Insurance Payment, Poperty.Monthly HOA Condo Fee][item != null])				
2	Non-Housing Debt Payments (Долговые платежи, несвязанные с оплатой за жилье) *Number*	sum(Borrower. Liabilities[Type !="Real estate loan" and To be paid off=false].Monthly payment)				
3	Income (Доход) *Number*	sum([Borrower.Employemnt Income, Borrower.Other Income][item ! = null])				
4	DTI Pct (Отношение размера кредита к стоимости залогового имущества) *tPercent*	decimal((Housing Expense+Non-Housing Debpt Payments)/Income*100,2				
5	Liquid Assets Before Closing (Ликвидные активы до закрытия) *Number*	sum(Borrower.Assets[Type= "Checking Savings Brokerage account" or Type= "Other Liquid"].Value)				
6	Debts Paid Off by Closing (Долги, погашенные при закрытии) *Number*	sum(Borrower. Liabilities[Type !="Real estate loan" and To be paid off=false].Balance[item! =null])				
7	Liquid Assets After Closing (Ликвидные активы после закрытия) Number	Liquid Assets Before Closing – Debpts Paid Off by Closing – Loan Info.Cash to Close				
8	Reserves (Резервы) *Number*	decimal(Liquid Assets After Closing/Housing Expense,2)				
Результат						

Рисунок 11.49: Параметры соответствия критериям

Для облегчения восприятия, модель бизнес-знаний «**Соответствие критериям**» изображается в двух частях (Рисунок 11.50 и Рисунок 11.51). Эта модель создает строку типа *tTableRow* для решения «**Таблица соответствия критериям**». Она моделируется как контекст, где первые четыре записи (рис. 11.50) вызывают модель бизнес-знаний для вычисления значений, которые заполнят компоненты *Table Row*.

- *Params* вызывает модель бизнес-знаний «**Соответствие критериям**» для данного **Кредитного продукта**.
- *Required Credit Score* использует *Params* для вызова модели бизнес-знаний «**Минимальная кредитная оценка**», возвращая минимальную кредитную оценку, необходимую для того, чтобы данному заемщику был одобрен этот Кредитный продукт.
- *Eligible* является булевым значением, сравнивающим кредитную оценку заемщика с **Минимальной кредитной оценкой.**
- *Recommendation* использует входные данные «**Рейтинг кредиторов**» в сочетании с «**Соответствие критериям**» для возврата рекомендованного значения для **Кредитного продукта**. *Recommendation* иллюстрирует альтернативный синтаксис таблицы решений, введенный в DMN 1.2, называемый обобщенным унарным тестом. В обобщенных унарных тестах входная запись таблицы решений может быть любым выражением FEEL, заменяя "?" для входного выражения. Например, в первом столбце этой таблицы решений правила фильтруют таблицу **Рейтингов кредиторов** для элемента с *Lender Name*, совпадающим с наименованием **Кредитного продукта** и *Customer Rating* в указанном диапазоне, возвращая значение true, если этот фильтр возвращает какие-либо значения.

Соответствие критериям
tEligibilityParameters

F	Loan Product *tLoanProduct*	Borrower *tBorrower*	Loan Info *tLoanInfoRow*	Property *tProperty*	Credit Score *tCreditScore [300..850]*	Ratings *tLenderRatings*

1	Params *tEligibilityParameters*	Eligibility Parameters(Loan Product, Borrower, Loan Info, Property, Credit Score)
2	Required Credit Score *tCreditScore* *[300..850]*	Min Credit Score(Params.DIT Pct, Loan Info.LTV, Params.Reserves)
3	Eligible *Boolean*	If Required Credit Score != null then Credit Score >= Required Credit Score else false
4	Eligible *tRecommendation* *“Best”, “Good”, “Not Recommended”, “Ineligible”*	

	Входные данные		Выходные данные
P	**Кредитный продукт**	**Соответствие критериям**	**Рекомендация**
	tLoanProduct	Boolean	tRecommendation “Best”, “Good”, “Not Recommended”, “Ineligible”
1	count(Ratings[Lender Name=?.Lender Name and Customer Rating>4])>0	true	“Best”
2	count(Ratings[Lender Name=?.Lender Name and Customer Rating in[3..4)])>0	true	
3	count(Ratings[Lender Name=?.Lender Name and Customer Rating<3])>0	true	
4	-	-	

Результат

Рисунок 11.50: Соответствие критериям (часть 1)

Остальная часть модели «**Соответствия критериям**» показана на рисунке 11.51.

- *Table Row* - это вложенный контекст с пустым значением для поля окончательного результата. Каждая контекстная запись представляет столбец в строке.
- Согласно спецификации DMN, поле окончательного результата может быть контекстом, но в этом примере мы используем запись контекста, чтобы создать значение результата и вернуть его в поле результата. Здесь контекстная запись *Table Row* создает структуру строки, и в окне окончательного результата просто выбирается эта контекстная запись.

5	Table row (Строка таблицы) *tTableRow*	1	Product (Продукт) *text*	Loan Product.Lender Name + “-” + Loan Product.Product Name
		2	Note Amount (Размер кредитного обязательства) *Number*	Loan Info.Note Amount
		3	Interest Rate Pct (Процентная ставка, %) *tPercent*	Loan Info.Initial Rate Pct
		4	Monthly Payment (Ежемесячный платеж) *Number*	Loan Info. Initial Monthly Payment
		5	LTV (Отношение размера кредита к стоимости залогового имущества) tPercent *Number*	Loan Info.LTV
		6	DTI Pct (Отношение размера кредита к стоимости залогового имущества) *tPercent*	Params.DTI Pct
		7	Cash to Close (Наличные для закрытия) *Number*	Loan Info.Cash to Close
		8	Liquid Assets After Closing (Ликвидные активы после закрытия) Number	Params.Liquid Assets After Closing
		8	Reserves (Резервы) *Number*	Params.Reserves
		10	Required credit score (Требуемая кредитная оценка) tCreditScore [300..850]	Required Credit Score
		11	Recommendation (Рекомендация) *tRecommendation* *“Best”, “Good”, “Not Recommended”, “Ineligible”*	Recommendation
			Результат	
Строка таблицы				

Рисунок 11.51: Соответствие критериям (часть 2)

Решение «**Таблица соответствия критериям**» (рисунок 11.52) использует альтернативную форму оператора *for..in..return* для итерации по индексу, вместо итерации по значениям элементов списка. Благодаря этому, возвращаемое выражение включает соответствующие элементы в несколько списков, в данном случае, это «**Кредитные продукты**» и «**Таблица с информацией о продукте**».

Таблица соответствия критериям *tEligibilityTable*
`for i in 1.. count(Loan Products) return Eligibility(Lan Products[i]), Borrower, Loan Info Table[i], Property, Credit Score)`

Рисунок 11.52: Таблица соответствия критериям

Решение верхнего уровня **Рекомендуемые кредитные продукты** (рис. 11.53) сначала сортирует **Таблицу соответствия критериям** на основе *Recommendation* и *Monthly payment*, а затем вызывает метод Java для форматирования числовых значений в виде строк для окончательного представления.

Рекомендуемые кредитные продукты *tRecommendedTable*			
1	precedes *boolean*	**F**	(X tTableRow , Y tTableRow) if x.Recommendation != “Ineligible” and y.Recommendation != “Ineligible” then x.Monthly Payment<y.Monthly Payment else if x.Recommendation != “Ineligible” and y.Recommendation = “Ineligible” then true else false
2	Sorted Table *tEligibilityTable*		sort(Eligibility Table, precedes)
Для строки в Sorted Table возвращать Format Row (row)			

Рисунок 11.53: Рекомендуемые кредитные продукты

- Первая запись контекста *precedes* является определением для функции FEEL *sort()*. Второй параметр *sort()*, называемый *precedes function*, является булевой функцией с двумя аргументами, представляющими элементы списка. Он возвращает true, если первый аргумент предшествует второму в отсортированном списке.

- Запись контекста *Sorted Table* выполняет сортировку. При использовании простых критериев сортировки функция *precedes function* обычно определяется как встроенная анонимная функция с использованием ключевого слова *function*, например,

  `sort(myTable, function(x,y) x.Amount < y.Amount),`

  которая сортирует строки *myTable* в порядке возрастания столбца *Amount (Сумма)*. Однако в решении «**Рекомендуемые кредитные продукты**» вместо этого используется именованная функция *precedes function*, в записи контекста *precedes*. В этом случае имя функции передает второй аргумент *sort()*.

- В поле окончательного результата выполняется итерация запроса модели бизнес-знаний **Format Row**, которая выполняет статический метод Java для форматирования числовых значений в отсортированной таблице в виде строк с символом валюты и двумя цифрами после десятичной точки. **Format Row** (рисунок 11.54) работает с одной строкой таблицы *Sorted Table*. Она моделируется как контекст.

- Первая запись контекста *string format* представляет собой определение функции Java, обозначенное кодом J. Согласно DMN такое определение функции является контекстом с двумя записями *class* и *method signature*. В этом примере на число накладывается маска строки, в результате чего возвращается отформатированное число в виде строки.

- Вторая запись контекста *formatted row* генерирует строку **«Рекомендуемые кредитные продукты»** в окончательном формате представления, вызывая *string format* для форматирования значений суммы и процентов.

- Поле окончательного результата возвращает отформатированную строку *formatted row*.

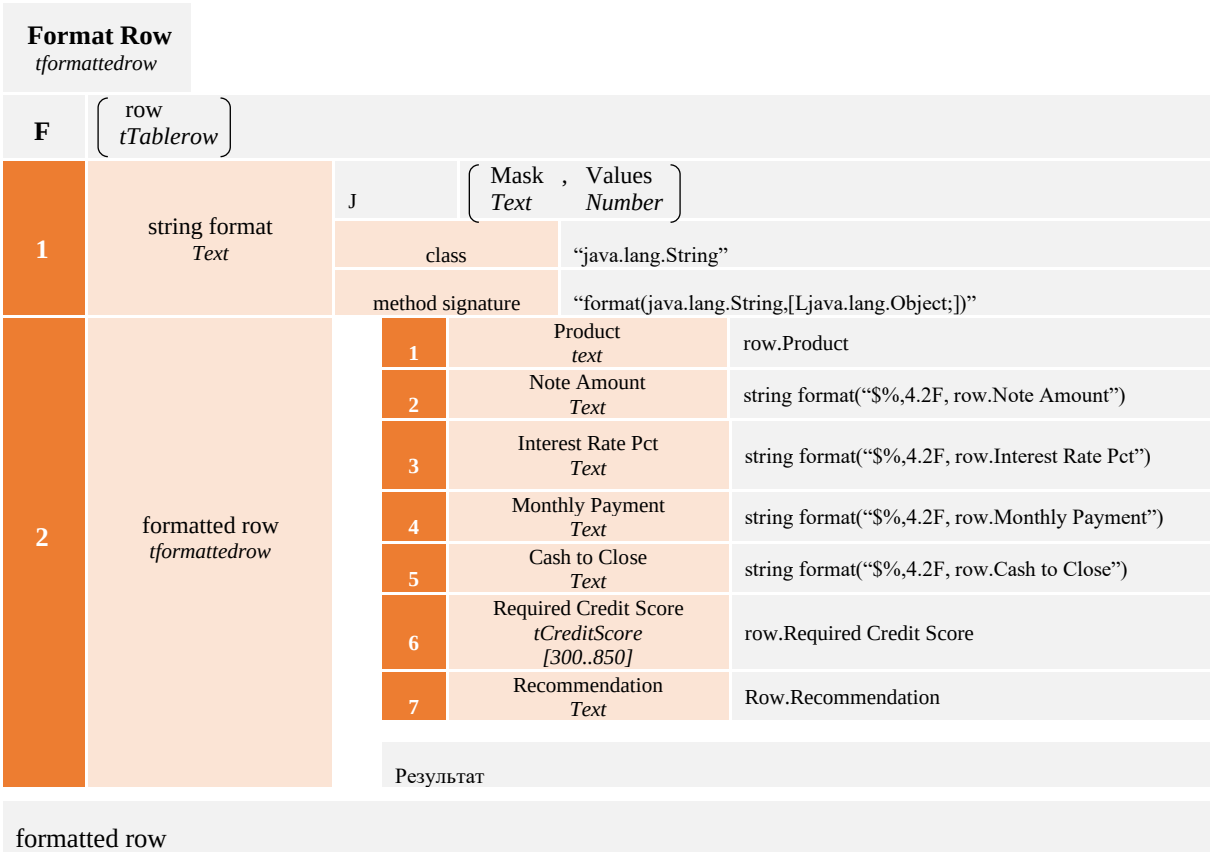


Рисунок 11.54: Format Row

На рисунке 11.55 показан результат решения «**Рекомендуемые кредитные продукты**» на основе входных тестовых данных, приведенных на рисунке 11.56.

Продукт	Размер кредитного обязательства	Процентная ставка	Размер ежемесячного платежа	Наличные на закрытие	Требуемая кредитная оценка	Рекомендация
Кредитор Б - ARM5/1-Standard	$273,775.90	3.75	$1,267.90	$75,475.52	720	Good
Кредитор В - Fixed30Standard	$274,599.40	3.88	$1,291.27	$75,491.99	680	Best
Кредитор Б - ARM5/1-NoPoints	$271,776.00	4.00	$1,297.50	$75,435.52	720	Good
Кредитор А - Fixed30-NoPoints	$271,925.00	4.08	$1,310.00	$75,438.50	680	Best
Кредитор В - Fixed15-Standard	$274,045.90	3.38	$1,942.33	$75,480.92	720	Best
Кредитор А - Fixed15-NoPoints	$270,816.00	3.75	$1,969.43	$75,416.32	720	Best

Рисунок 11.55: Результат решения «Рекомендуемые кредитные продукты», полученный на основе тестовых данных

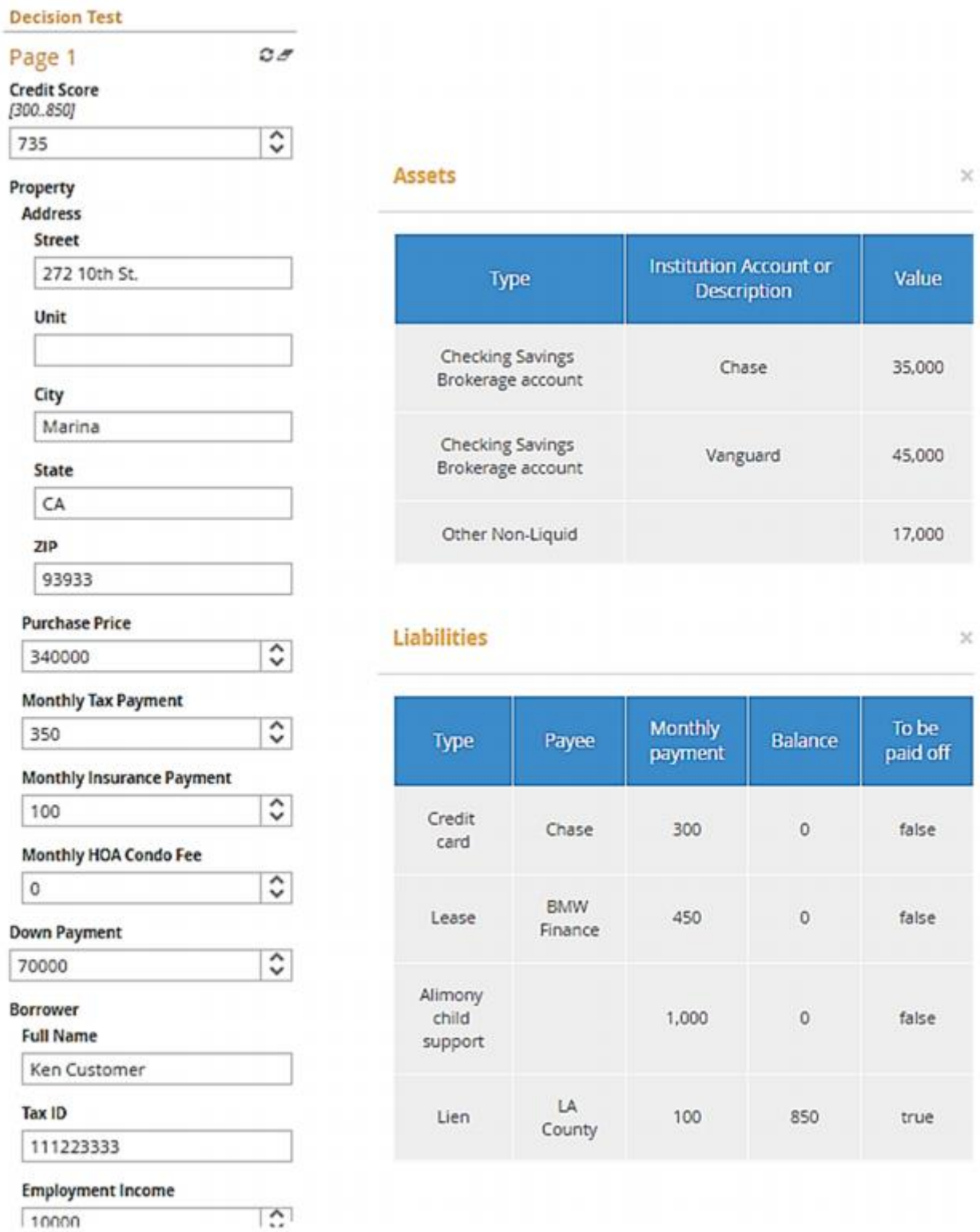


Рисунок 11.56: Тестовые входные данные (частично)

12 Форматы обмена

12.1 Обмен неполными моделями

Обмен неполными моделями **DMN** является обычной практикой. Такой обмен часто происходит при выполнении итеративного моделирования, когда один пользователь (например, эксперт по источникам знаний или бизнес-пользователь) сначала определяет модель высокого уровня, а затем передает ее другому человеку для завершения или доработки.

«Неполными» считаются модели, в которых еще не заполнены все обязательные атрибуты модели или нижняя граница множества атрибутов и ассоциаций не удовлетворена.

XMI позволяет осуществлять обмен такими неполными моделями. В **DMN** мы расширяем эту возможность для обмена XML-файлами на основе XML-схемы **DMN**. Предполагается, что для таких файлов XML разработчики обеспечат возможность обмена:

- без учета отсутствующих атрибутов, помеченных как «обязательные» в XML-схеме **DMN**;
- посредством уменьшения нижней границы элементов, у которых значение «minOccurs» больше 0.

12.2 Машиночитаемые файлы

Все машиночитаемые файлы, включая XSD, XMI и XML-файлы, можно найти в документе OMG Document от 15-11-12, который является zip-архивом;

- для модели **DMN** XMI основным файлом является DMN.xmi;
- для **DMN** XSD Interchange (поддерживаемые уровни соответствия 1, 2 и 3) основным файлом является DMN.xsd;
- социализация примера в Разделе 11 рассматривается в примере ch11.

12.3 XSD

12.3.1 Структура документа

Специфичный для данного домена набор элементов модели взаимозаменяем в одном или нескольких файлах **DMN**. Корневой элемент каждого файла ДОЛЖЕН быть `<DMN:Definitions>`. Набор файлов ДОЛЖЕН быть автономным, т. е. все определения, которые используются в файле, ДОЛЖНЫ быть импортированы прямо или косвенно с помощью элемента `<DMN:Import>`.

Каждый файл ДОЛЖЕН объявить пространство имен “`namespace`”, которое МОЖЕТ быть различным для различных файлов одной модели.

Файлы **DMN** МОГУТ импортировать файлы, отличные от **DMN** (такие как XSD и PMML), если содержащиеся в них элементы используют внешние определения.

12.3.2 Ссылки в DMN XSD

Многие элементы **DMN**, на которые могут ссылаться другие элементы, содержат идентификаторы ID. Внутри **BPMN** XSD ссылки на элементы выражаются через эти идентификаторы ID. Тип XSD IDREF является традиционным механизмом для ссылки по идентификаторам ID, однако он может ссылаться только на элемент внутри того же файла. Элементы **DMN** типа `DMNElementReference` поддерживают ссылки по ID между файлами при помощи атрибута `href`, значение которого должно быть допустимой URI ссылкой [RFC 3986], где компоненты пути могут быть абсолютными или относительными, ссылка не имеет компонента запроса, а фрагмент состоит из значения `id` адресуемого элемента **DMN**.

Например, рассмотрим следующий элемент `Decision`:

```
<decision name="Pre-Bureau Risk Category" id="prebureauriskDec01">…</decision>.
```

Когда на это решение `Decision` ссылаются, например, при помощи элемента `InformationRequirement`, которое находится в определенном в другом файле

Decision, ссылка может иметь следующий вид:

```
<requiredDecision
href="http://www.example.org/Definitions01.xml#prebureauriskDec01"/>,
```

где "http://www.example.org/Definitions01.xml" является ссылкой URI на XML-документ, в котором определено решение "Pre-Bureau Risk Category" (например, значение атрибута locationURI в соответствующем элементе импорта), а «prebureauriskDec01» – значение атрибута id для Решения.

Если компонент пути в ссылке URI является относительным, базовый URI, для которого применяется относительная ссылка, определяется, как указано в [RFC 3986]. Согласно этой спецификации, «если базовый URI не является встроенным, а представление не инкапсулировано в какой-либо другой объект, тогда, если для получения представления использовался URI, этот URI должен считаться базовым URI» ([RFC 3986], раздел 5.1.3). То есть, если ссылка не входит в область атрибута xml:base XBASE, значение атрибута href, содержащее только фрагмент и не включающее в себя компонентов пути, ссылается на элемент **DMN**, который определен в том же экземпляре XML-файла в виде адресуемого элемента. В приведенном ниже примере, с учетом того, что элемент requiredDecision не входит в область атрибута xml:base, элемент **DMN**, id которого является "prebureauriskDec01", должен быть определен в том же XML-документе:

```
<requiredDecision href="#prebureauriskDec01" />.
```

Обратите внимание, что ссылка на процессы **BPMN** и задачи, использующие решение, осуществляется также при помощи атрибута href: он совместим с системой, что позволяет ссылаться на внешние экземпляры Process и Task в определениях **BPMN 2.0**, которые также основаны на идентификаторах.

Атрибут typeRef ссылается на ItemDefinitions и встроенные типы по имени, а не по идентификатору. Для поддержки импортируемых типов, typeRef использует синтаксис квалифицированного имени с указанием пространства имен [qualifer].[Local-name], где спецификатор для импортируемого типа задается атрибутом имени элемента Import. Префикс ДОЛЖЕН быть опущен в случае, если атрибут ссылается на не импортируемый тип.

13 Обмен диаграммами DMN (DMN DI)

13.1 Область применения

В этой главе описывается метамодель и схема обмена диаграммами DMN 1.3 (DMN DI). Основное предназначение DMN DI – это обеспечение обмена диаграммами DMN между инструментами моделирования, но не воспроизведение этих диаграмм. DMN DI реализует самый простой подход к обмену, гарантирующий однозначный рендеринг диаграмм. При этом, сам по себе формат DMN DI не подразумевает, что при обмене будут сохранены или переданы какие-либо функциональные особенности, присущие конкретным инструментам моделирования, например, стилевое оформление, особенности компановки, и т. д..

DMN DI не проверяет наличие синтаксических или семантических ошибок в диаграмме DMN.

Эта версия DMN DI ориентирована на обмен диаграммами требований к решению (DTP). Функция обмена диаграммами для табличных выражений и таблиц принятия решений возможно будет реализована в следующих версиях.

13.2 Определение и обмен диаграммами

Метамодель DMN DI, аналогичная метамодели абстрактного синтаксиса DMN, определяется как метамодель на основе MOF. Как таковые, её экземпляры могут быть сериализованы и переданы с использованием XMI. Также DMN DI определяется схемой XML, поэтому её экземпляры могут быть сериализованы и переданы с использованием XML.

И метамодель, и схема DMN DI согласованы со Стандартом определения диаграмм OMG (версия 1.1). Стандарт состоит из двух частей: «Общие сведения о диаграммах» и «Обмен диаграммами». В первой части определяются общие типы, такие как границы и точки; во второй – приводятся общие принципы для определения предметно-ориентированных моделей диаграмм. DMN DI, будучи предметно-ориентированной схемой обмена диаграммами, определяет несколько новых классов метамодели, которые являются производными от абстрактных классов, упомянутых в второй части стандарта «Обмен диаграммами».

Назначение DMN DI – это обмен упорядоченными фигурами и гранями, из которых состоит диаграмма **DMN**. Каждая фигура и грань ссылается на определенный элемент модели **DMN**, который в свою очередь, является частью конкретной модели **DMN**. По сути в DMN DI содержится исключительно та информация, которая не отображается и не может быть выведена из модели DMN. Иными словами, для рендеринга диаграммы **DMN** НЕОБХОДИМЫ как экземпляр(ы) DMN DI, так и модель **DMN**.

В разрезе DMN DI, диаграмма DMN – это снепшот модели **DMN** в определенный момент времени. Можно осуществить обмен несколькими диаграммами **DMN**, ссылаясь на элементы одной и той же модели **DMN**. Каждая диаграмма может обеспечивать неполное или частичное отображение содержимого модели **DMN**. Согласно разделу 12, пакет модели **DMN** состоит из одного или нескольких файлов. Каждый файл может содержать любое количество диаграмм **DMN**. Инструмент экспорта может решать, сколько диаграмм экспортировать, а инструмент импорта может решать, надо ли показывать эти диаграммы пользователю и каким образом это сделать.

13.3 Как читать эту главу

В разделе 13.4 подробно описана метамодель, используемая для сохранения компановки и внешнего вида диаграмм **DMN**. В разделе 13.5 в таблицах приводится библиотека изображений элементов **DMN** и указывается однозначное разрешение элемента модели **DMN** в его изображение.

13.4 Мета-модель обмена диаграммами DMN

13.4.1 Обзор

DMN DI является экземпляром метамодели OMG DI. Основная концепция DMN DI (как и DI в целом), заключается в том, что сериализация диаграммы [DMNDiagram] для обмена происходит при помощи спецификации коллекции фигур [DMNShape] и граней [DMNEdge].

Классы DMN DI определяют только визуальные свойства, используемые для отображения. Все остальные свойства, которые НЕОБХОДИМЫ для однозначного воспроизведения элемента **DMN**, являются производными от элемента **DMN** [dmnElementRef].

Диаграммы **DMN** могут отображать неполную модель **DMN** или её части. Некоторые элементы модели **DMN** могут быть исключены из всех экземпляров диаграмм, которые участвуют в обмене.

DMN DI не реализует как таковую концепцию вложенности. DMNDiagram – это упорядоченная коллекция смешанных DMNShape(s) и DMNEdge(s). Положение DMNShape(s) и DMNEdge(s) внутри DMNDiagram определяется Z-порядком. DMNShape(s) и DMNEdge(s), которые нужно изобразить «поверх» других DMNShape(s) и DMNEdge(s), ДОЛЖНЫ появляться после них в DMNDiagram. Таким образом, инструмент экспорта ДОЛЖЕН упорядочивать все DMNShape(s) и DMNEdge(s) так, чтобы желаемое изображение могло быть срендерено.

13.4.2 Единица измерения

Согласно OMG DD, все координаты и длины, определенные DMN DI, считаются заданными в пользовательских единицах, если не указано иное. Пользовательская единица – это значение в пользовательской системе координат, которое изначально (до применения какого-либо преобразования) согласуется с системой координат устройства (например, пиксельной сеткой дисплея). Пользовательская единица, следовательно, представляет собой логическую, а не физическую единицу измерения. Некоторые приложения могут задавать физические размеры для диаграммы (в основном для печати), поэтому в качестве разрешения диаграммы можно указать результат преобразования пользовательской единицы в физическую. В данной спецификации для единообразия используются дюймы, но инструменты моделирования легко могут конвертировать дюймы в другие физические единицы измерения. Разрешение указывает, сколько пользовательских единиц умещается в одной физической единице (например, разрешение 300 указывает, что 300 пользовательских единиц умещаются в 1 дюйм на устройстве).

13.4.3 DMNDI [Class]

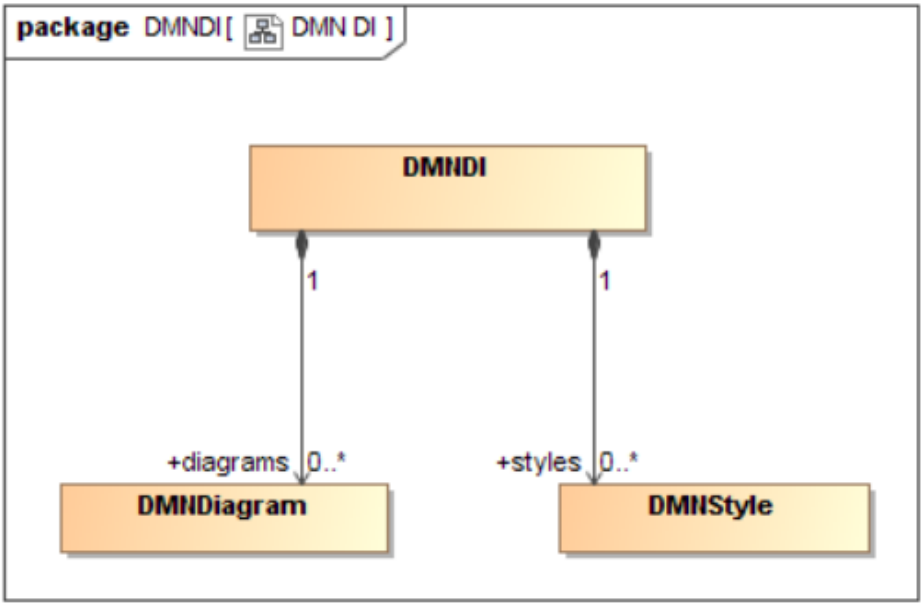


Рисунок 13.1: DMNDI

Класс DMNDI является контейнером для общих стилей DMNStyle и всех диаграмм DMNDiagram, определенных в Definitions.

Таблица 87: Атрибуты DMNDI

Атрибуты	Описание
styles: DMNStyle [0..*]	Список общих DMNStyle, на которые могут ссылаться все DMNDiagram и DMNDiagramElement.
diagrams: DMNDiagram [0..*]	Список DMNDiagram

13.4.4 DMNDiagram [Class]

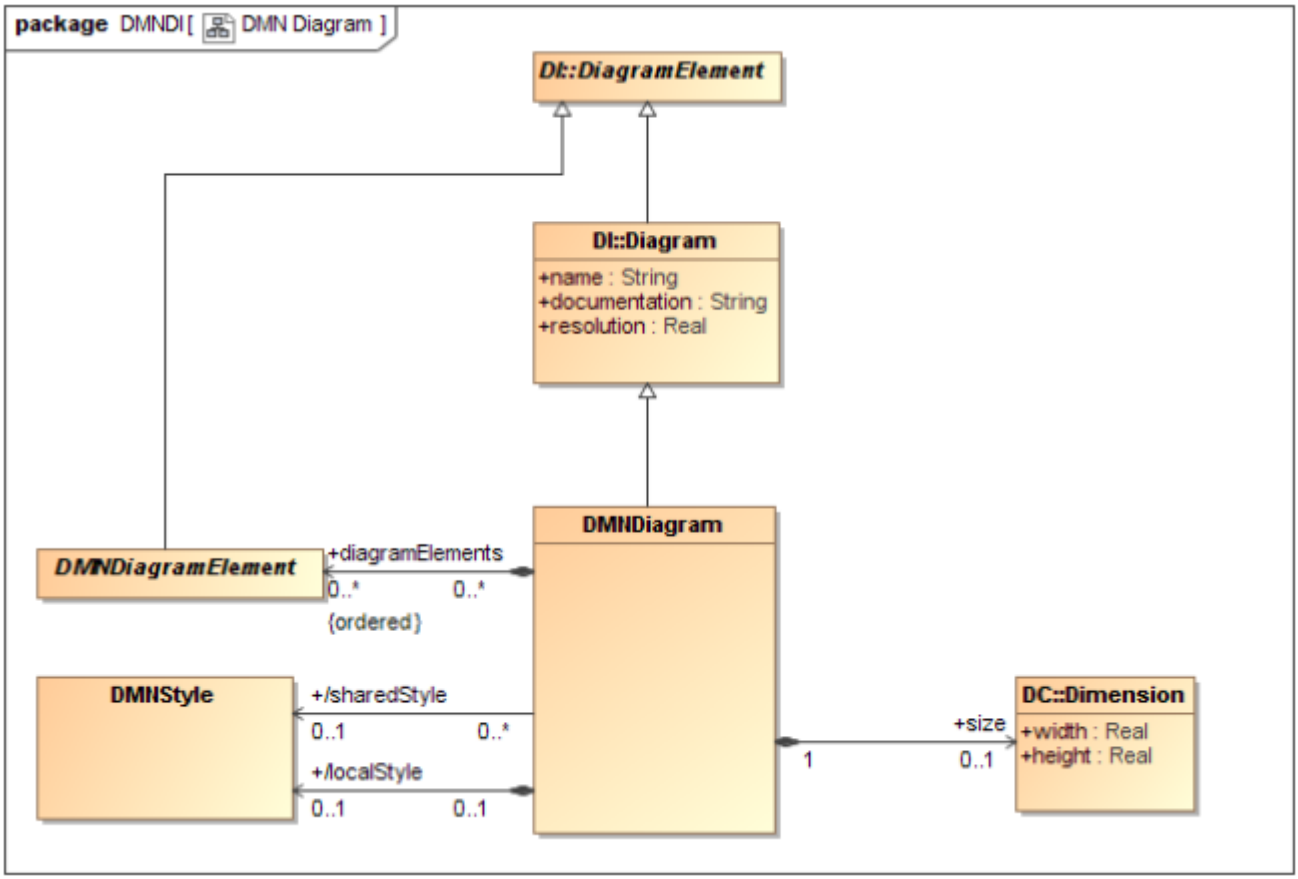


Рисунок 13.2: DMNDiagram

Класс DMNDiagram определяет DI::Diagram. Это вид диаграмм, представляющий описание всей модели **DMN** или её части.

DMNDiagram является контейнером для DMNDiagramElement (DMNShape (s) и DMNEdge (s)). DMNDiagram не может включать другие DMNDiagram.

DMNDiagram может определять DMNStyle локально и/или может ссылаться на общий DMNStyle, определенный в DMNDI. Свойства, определенные в локальном стиле, переопределяют свойства в общем стиле. Эта комбинация стилей (общий и локальный) является стилем по умолчанию для всех элементов DMNDiagramElement, содержащихся в DMNDiagram.

Класс DMNDiagram представляет двумерную поверхность с началом координат (0, 0) в верхнем левом углу. Это означает, что оси X и Y имеют увеличивающиеся координаты справа и снизу. Для элементов диаграммы, которые вложены в DMNDiagram, допускаются только положительные координаты.

Атрибуты DMNDiagram:

Таблица 88: Атрибуты DMNDiagram

Атрибуты	Описание
name: Sring	Наименование диаграммы. По умолчанию пустая строка String.
documentation: String	Документация по диаграмме. По умолчанию пустая строка String.
resolution: Real	Разрешение диаграммы, выраженное в пользовательских единицах на дюйм. По умолчанию 300.
diagramElements: DMNDiagramElement [0..*]	Список DMNDiagramElement (DMNShape и DMNEdge), которые изображены на диаграмме.
sharedStyle: DMNStyle [0..1]	Ссылка на DMNStyle, определенный в DMNDI, который служит стилем по умолчанию для DMNDiagramElement в данной DMNDiagram.
localStyle: DMNStyle [0..1]	DMNStyle, который определяет стиль по умолчанию для этой диаграммы. Свойства, определенные в этом стиле, переопределяют свойства в sharedStyle.
size: DC::Dimension [0..1]	Размер диаграммы. Если он не задан, размер DMNDiagram не ограничен.

13.4.5 DMNDiagramElement [Class]

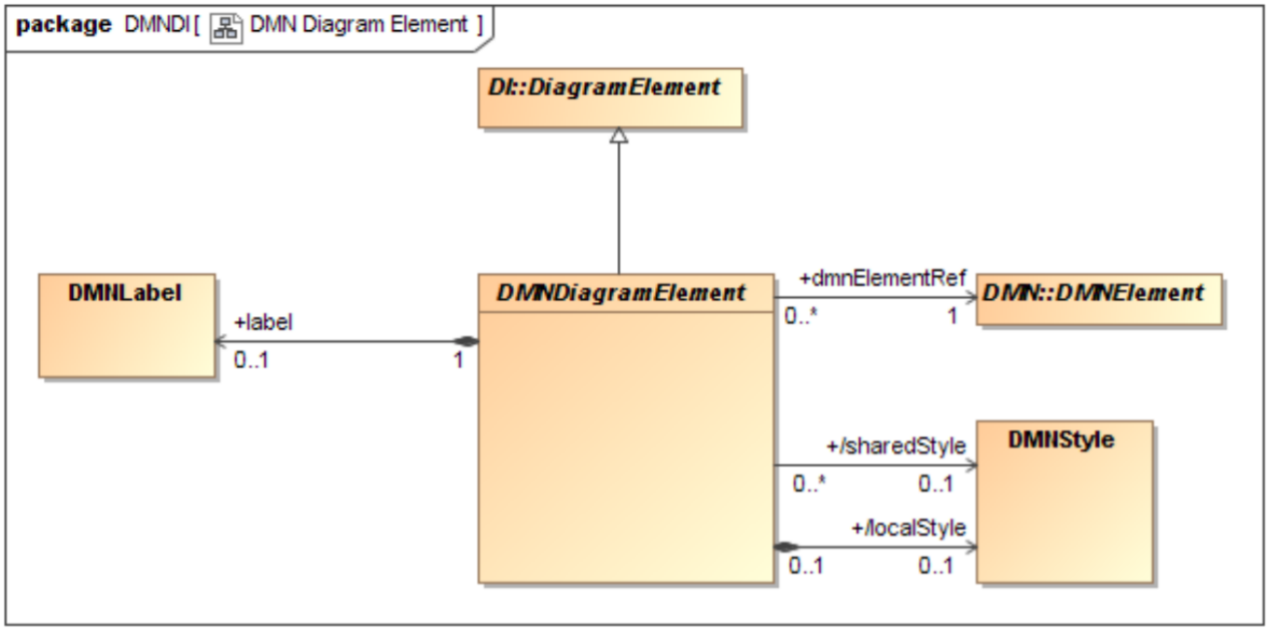


Рисунок. 13.3: DMNDiagramElement

Класс DMNDiagramElement содержится в DMNDiagram и является базовым классом для DMNShape и DMNEdge.

DMNDiagramElement наследует стиль от родительской DMNDiagram. Кроме того, он может ссылаться на один из общих стилей DMNStyle, определенных в DMNDI, и/или он может определять локальный стиль. В разделе 13.4.9 приводится более подробная информация о стилевом оформлении.

Также DMNDiagramElement МОЖЕТ содержать DMNLabel, если у него есть видимая текстовая подпись (text lable). Если подпись DMNLabel не задана, элемент DMNDiagramElement должен отображаться без подписи.

Атрибуты DMNDiagramElement:

Таблица 89: Атрибуты `DMNDiagramElement`

Атрибуты	Описание
dmnElementRef: `DMNElement` **`[1]`**	Ссылка на отображаемый `DMNElement`.
sharedStyle: `DMNStyle` **`[0..1]`**	Ссылка на `DMNStyle`, определенный в DMNDI.
localStyle: `DMNStyle` **`[0..1]`**	`DMNStyle`, который определяет стиль для данного элемента.
label: `DMNLabel` **`[0..1]`**	Опциональная подпись, если у `DMNElement` есть видимая текстовая подпись.

13.4.6 DMNShape [Class]

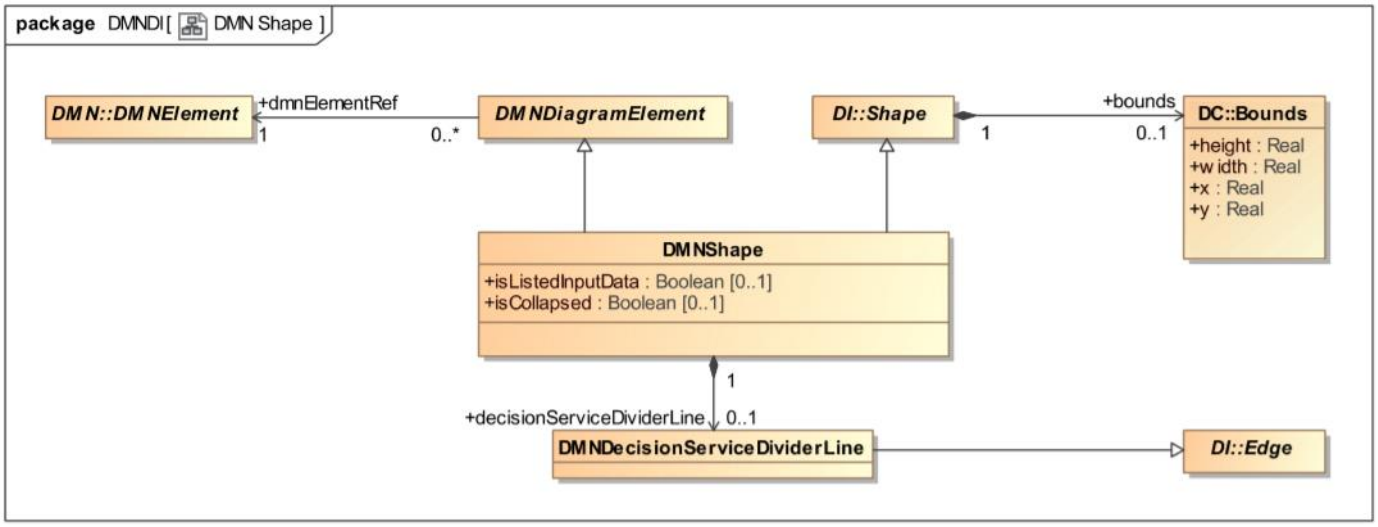


Рисунок 13.4: DMNShape

Класс `DMNShape` определяет DI :: Shape и `DMNDiagramElement`. Это своего рода фигура, которая изображает элемент `DMNElement` модели DMN.

`DMNShape` представляет решение, модель бизнес-знаний, элемент входных данных, источник знаний, службу принятия решений или текстовую аннотацию, изображенную на диаграмме.

У DMNShape есть три дополнительных свойства (isListedInputData, isCollapsed и SolutionServiceDividerLine), которые используются для более полного описания тех фигур, чей внешний вид не возможно установить на основании модели DMN.

`DMNShape` расширяет `DI :: Shape` и `DMNDiagramElement` и обладает следующими атрибутами:

Таблица 90: Атрибуты DMNShape

Атрибуты	Описание
bounds: DC:: Bounds [1]	Границы фигуры относительно границ родительской DMNDiagram. Границы ДОЛЖНЫ быть указаны.
dmnElementRef: DMNElement [1]	Ссылка на решение, модель бизнес-знаний, элемент входных данных, источник знаний, службу принятия решений, группу или текстовую аннотацию ДОЛЖНА быть указана.
isListedInputData: Boolean [0..1]	Если DMNShape представляет элемент Входные данные, то данный атрибут определяет, каким образом они отображаются. Входные данные могут быть перечислены в элементе «Решение» (true) или изображены как отдельные элементы нотации на ДТР (false).
decisionServiceDividerLine: DMNDecisionServiceDividerLine [0..1]	Если DMNShape отображает Службу принятия решений, то данный атрибут ссылается на линию DMNDecisionServiceDividerLine, которая представляет собой грань DI :: Edge. Она определяет s, где DMNShape делится на две части прямой сплошной линией. Такое может случиться, например, когда DMNShape изображает Службу решений, у которой набор выходных решений меньше, чем набор инкапсулированных решений. Начальная и конечная точки SolutionServiceDividerLine ДОЛЖНЫ располагаться на границе DMNShape.
isCollapsed Boolean [0..1] = false	Если DMNShape отображает DecisionService, данный атрибут определяет, как показывать службу: в развернутом виде– false, в свернутом – true. По умолчанию false.

13.4.7 DMNEdge [Class]

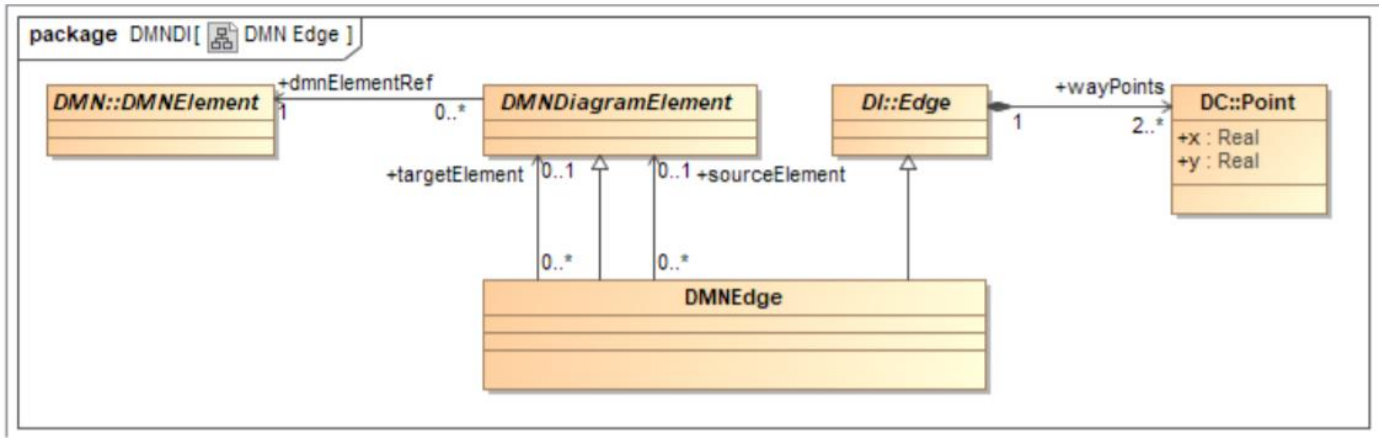


Рисунок 13.5: DMNEdge

Класс DMNEdge определяет DI :: Edge и DMNDiagramElement. Это своего рода грань, которая позволяет изобразить отношения между двумя элементами модели **DMN**.

DMNEdge используется для отображения требований или ассоциаций на модели **DMN**. Поскольку DMNDiagramElement может отображаться более одного раза, атрибуты sourceElement и targetElement позволяют определить, с каким изображением связана DMNEdge.

Если у DMNEdge есть источник, его sourceModelElement ДОЛЖЕН ссылаться на DMNDiagramElement, с которого он начинается. Упомянутый DMNDiagramElement ДОЛЖЕН разрешаться в DMNElement, который является фактическим источником требования или ассоциации. Для требования это обязательный элемент DMNElement.

Если DMNEdge направлен к какому-то элементу, его targetModelElement ДОЛЖЕН ссылаться на DMNDiagramElement, на котором он заканчивается. DMNDiagramElement ДОЛЖЕН разрешаться в DMNElement, который фактически является элементом, к которому направлено требование или ассоциация. В случае с требованиями, это DMNElement, который содержит требовние.

DMNEdge расширяет DI :: Edge и имеет следующие атрибуты:

Таблица 91: Атрибуты DMNEdge

Атрибуты	Описание
wayPoints: DC:: Point **[2..*]**	Список точек относительно родительской DMNDiagram, который определяет соединенные отрезки грани. НЕОБХОДИМО указать не менее двух (2) точек.
dmnElementRef: DMNElement **[1]**	Ссылка на InformationRequirement, KnowledgeRequirement, AuthorityRequirement или Association.
SourceElement: DMNDiagramElement **[0..1]**	Определяет элемент DMNDiagramElement, из которого исходит DMNEdge. Атрибут НЕОБХОДИМО указывать, если у DMNEdge есть источник source.
TargetElement: DMNDiagramElement **[0..1]**	Определяет элемент DMNDiagramElement, к которому направлен DMNEdge. Атрибут НЕОБХОДИМО указывать, если у DMNEdge есть target.

13.4.8 DMNLabel [Class]

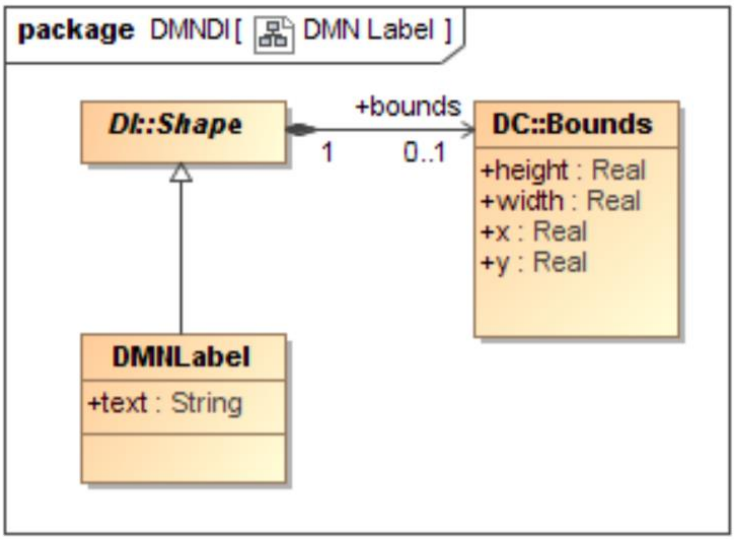


Рисунок 13.6: DMNLabel

DMNLabel представляет собой текстовую информацию об элементе DMN.

Подпись DMN не является элементом верхнего уровня и всегда вложена либо в DMNShape, либо в DMNEdge. Она не ссылается напрямую на элемент **DMN**, а скорее наследует ссылку от родительских DMNShape или DMNEdge. Текстовая

информация, отображаемая в подписи, получена из атрибута name соответствующего DMNElement.

DMNLabel расширяет DI :: Shape и обладает следующими атрибутами:

Таблица 92: Атрибуты DMNLabel

Атрибуты	Описание
bounds: Bounds [0..1]	Границы DMNLabel. Когда границы не заданы, подпись располагается в позиции по умолчанию согласно разделу 13.5.
dmnElementRef: DMNElement [1]	Необязательный печатный текст, который ДОЛЖЕН отображаться вместо имени элемента DMNElement, если он присутствует.

13.4.9 DMNStyle [Class]

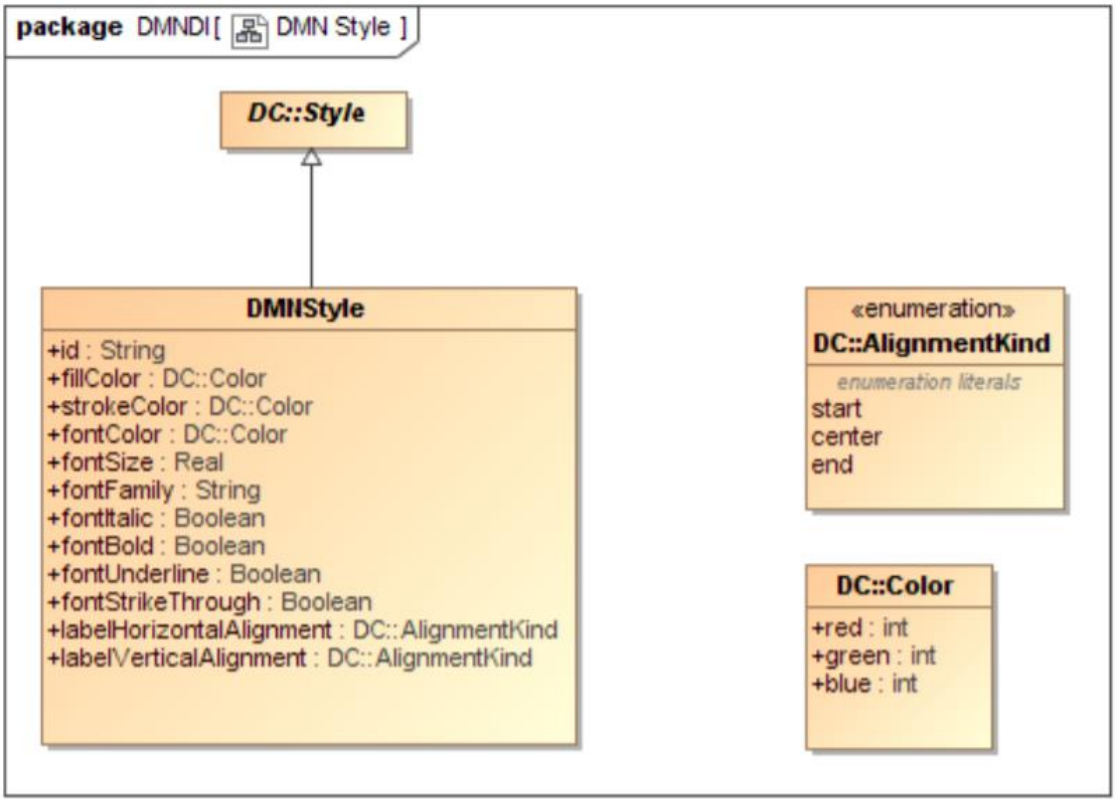


Рисунок 13.7: DMNStyle

DMNStyle определяет DC :: Style. Это стиль, который передает параметры внешнего вида для DMNDiagramElement.

DMNStyle используется для хранения некоторых ненормативных визуальных атрибутов, таких как цвета и шрифт. DMN не регламентирует семантику цвета и стиля шрифта, но инструменты моделирования могут использовать такое форматирование и обмениваться соответсвующими атрибутами.

Стиль DMNDiagramElement рассчитывается путем фильтрации атрибутов DMNStyle, определенных на другом уровне иерархии. Каждый атрибут рассматривается независимо (это означает, что каждый атрибут DMNStyle может перекрываться по отдельности). Существуют следующие правила приоритета:

- DMNStyle, определенный атрибутом localStyle элемента DMNDiagramElement
- DMNStyle, на который ссылается атрибут sharedStyle элемента DMNDiagramElement
- DMNStyle, определенный атрибутом localStyle родительской DMNDiagram
- DMNStyle, на который ссылается атрибут sharedStyle родительской DMNDiagram

Значение атрибута по умолчанию определено в таблице 93 (атрибуты DMNStyle).

Например, предположим, что:

- У `DMNDiagramElement` есть локальный `DMNStyle`, который определяет fillColor и strokeColor.
- Его родительская `DMNDiagram` определяет локальный `DMNStyle`, который задаёт fillColor и fontColor.

Тогда полученный `DMNDiagramElement` должен использовать:

- fillColor и strokeColor, определенные на уровне `DMNDiagramElement` (так как они определены локально).
- fontColor, определенный на уровне `DMNDiagram` (так как fillColor был перекрыт локально).
- Все остальные атрибуты `DMNStyle` сохраняют значения по умолчанию.

`DMNStyle` расширяет `DC :: Style` и обладает следующими свойствами:

Таблица 93: Атрибуты `DMNStyle`

Атрибуты	**Описание**
id: `String` **`[0..1]`**	Уникальный идентификатор стиля, позволяющий ссылаться на него. `DMNDiagramElement` и `DMNDiagram` могут ссылаться только на стили, определенные в DMNDI.
fillColor: DC:: Color **`[1]`**	Цвет, используемый для заполнения фигуры. Не относится к `DMNEdge`. По умолчанию белый.
strokeColor: DC::Color [0..1]	Цвет, используемый для рисования границ фигуры. По умолчанию черный.
fontColor: DC::Color [0..1]	Цвет, используемый для подписи. По умолчанию черный.
fontFamily: String [0..1]	Разделенный запятыми список названий шрифтов, которые можно использовать для отображения текста. По умолчанию Arial.
fontSize: Real [0..1]	Размер шрифта, используемого для отображения текста, указанный в типографских пунктах. По умолчанию 8.
fontItalic: Boolean [0..1]	Определяет, должен ли текст выделяться курсивом. По умолчанию fasle.
fontBold: Boolean [0..1]	Определяет, должен ли текст выделяться полужирным шрифтом. По умолчанию fasle.
fontUnderline: Boolean [0..1]	Определяет, должен ли текст подчеркиваться. По умолчанию fasle.
fontStrikeThrough: Boolean [0..1]	Определяет, должен ли текст отображаться зачеркнутым. По умолчанию fasle.
labelHorizontalAlignement: `AlignmentKind [0..1]`	Определяет, горизонтальное положение текста в пределах границ подписи. Значение по умолчанию зависит от элемента `DMNDiagramElement`, к которому прикреплена подпись (см. 13.5).
labelVerticalAlignment: `AlignmentKind [0..1]`	Определяет, вертикальное положение текста в пределах границ подписи. Значение по умолчанию зависит от элемента `DMNDiagramElement`, к которому прикреплена подпись (см. 13.5). Начало означает «верх», а конец - «низ».

13.5 Библиотека изображений и разрешений абстрактных элементов нотации

DMN нотация определяет изображение для каждого элемента **DMN**.

Для того, чтобы произвести обмен, необходимо сериализовать **DMN**-диаграмму, основываясь на спецификации коллекции `DMNShape(s)` (см. 13.4.6) и `DMNEdge(s)` (см. 13.4.7) в `DMNDiagram` (см. 13.4.4). Атрибуты `DMNShape`(s) и `DMNEdge`(s) должны быть заполнены таким образом, чтобы принимающая сторона могла однозначно отобразить диаграмму DMN. Если точнее, `DMNShape(s)` и `DMNEdge(s)` ДОЛЖНЫ ссылаться на элементы модели **DMN**. Если ссылка на `DMNElement` отсутствует или недействительна, предполагается, что фигура или грань не будут отображены.

При рендере диаграммы **DMN** правильное отображение `DMNShape` или `DMNEdge` зависит главным образом от элемента модели DMN, на который они ссылаются, и его конкретных атрибутов и/или ссылок. Информация в данном разделе ознакомит вас с библиотекой изображений элемента DMN и поможет обеспечить однозначное разрешение между элементом модели DMN [`DMNElement`] и его изображением. Таблицы разрешения изображения приведены ниже как для `DMNShape` (см. 13.5.2), так и для `DMNEdge` (см. 13.5.3).

13.5.1 Подписи

Подписи (атрибуты имени) могут присутствовать как на `DMNShape`, так и на `DMNEdge`. Они размещаются на фигуре/грани, под или над фигурой/гранью, в любом направлении или положении, в зависимости от предпочтений разработчика моделей или особенностей инструментов моделирования.

Подписи считаются необязательными для `DMNShape` и `DMNEdge`. Если подпись все же используется, то ее положение определяется границами `DMNLabel` соответствующей `DMNShape` или `DMNEdge`. Проще говоря, видимость подписи определяется наличием элемента `DMNLabel`.

Границы `DMNLabel` являются необязательными и всегда относятся к исходной точке `DMNDiagram`, содержащей `DMNLabel`. Таблицы разрешения изображений, приведенные ниже, иллюстрируют положения подписей по умолчанию, если для `DMNLabel` не заданы границы (для видов `DMNShape` (см. 13.5.2) и видов `DMNEdge` (см. 13.5.3)).

Если `DMNLabel` содержится в `DMNShape`, отображаемый текст - это имя элемента `DMNElement`.

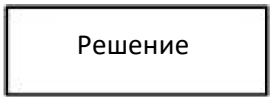
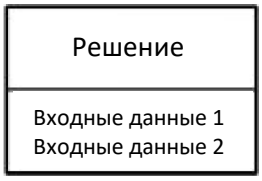
13.5.2 Разрешение DMNShape

DMNShape может быть использован для того, чтобы изобразить Решение, Модель бизнес-знаний, Входные данные, Источник знания, Группу и Службу решений.

13.5.2.1 Решение

Решение изображается на ДТР в виде прямоугольника, как правило, нарисованного сплошной линией. Если используется опция «перечисление входных данных», то все требования данного решения для входных данных должны быть перечислены под подписью решения, при этом они должны быть отделены от нее горизонтальной линией. Названия перечисленных входных данных должны находиться точно внутри фигуры элемента ДТР.

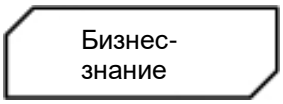
Таблица 94. Разрешение изображения Решения

DMNElement	Атрибуты DMNShape	Изображение
Решение	Отсутствуют	
Решение с двумя элементами входные данные	У фигур входных данных есть атрибут isListedInputData=true	

13.5.2.2 Модель бизнес-знаний

Модель бизнес-знаний изображается на ДТР в виде прямоугольника с двумя срезанными краями. Контур фигуры обычно выполняется сплошной линией.

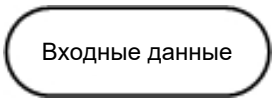
Таблица 95. Разрешение изображения Модели бизнес-знаний

DMNElement	Атрибуты DMNShape	Изображение
Модель бизнес-знаний	Отсутствуют	

15.5.2.3 Элемент Входные данные

Элемент Входные данные изображается на ДТР в виде фигуры с двумя параллельными прямыми сторонами и двумя полукруглыми краями. Контур фигуры, как правило, нарисован сплошной линией.

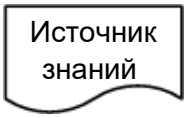
Таблица 96. Разрешение изображения Входных данных

DMNElement	Атрибуты DMNShape	Изображение
Входные данные	Отсутствуют	

13.5.2.4 Источник знаний

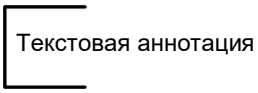
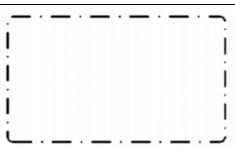
Источник знаний изображается в виде фигуры с тремя прямыми сторонами и одной волнообразной, как правило, нарисованными сплошной линией.

Таблица 97. Разрешение изображения Источника знаний

DMNElement	Атрибуты DMNShape	Изображение
Источник знаний	Отсутствуют	

13.5.2.5 Артефакты

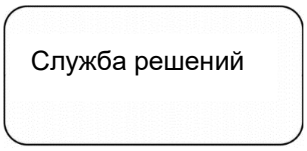
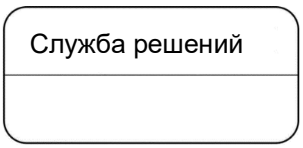
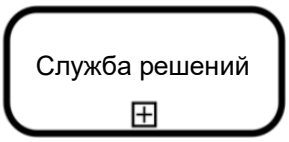
Таблица 98. Разрешение изображения Артефактов

DMNElement	Атрибуты DMNShape	Изображение
Текстовая аннотация	Отсутствуют	
Группа	Отсутствуют	

13.5.2.6 Служба решений

Если набор выходных решений меньше, чем набор инкапсулированных решений, то Службу решений следует разделить на две части сплошной прямой линией.

Таблица 99. Разрешение изображения Службы решений

DMNElement	Атрибуты DMNShape	Изображение
Служба решений	Отсутствуют либо isCollapsed=false	
Служба решений	DecisionServiceDividerLine isCollapsed=false	
Служба решений	isCollapsed=true	

13.5.3 Разрешение DMNEdge

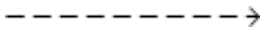
13.5.3.1 Требование к информации

Таблица 100. Разрешение изображения Требования к информации

DMNElement	Изображение
Требование к информации	

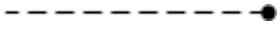
13.5.3.2 Требование к знаниям

Таблица 101. Разрешение изображения Требования к знаниям

DMNElement	Изображение
Требование к знаниям	

13.5.3.3 Требования к полномочиям

Таблица 102. Разрешение изображения Требования к полномочиям

DMNElement	Изображение
Требование к информации	

13.5.3.4 Ассоциации

Когда `DMNEdge` изображает Ассоциацию, НЕОБХОДИМО указывать связанный с ней `DMNElement`.

Таблица 103. Разрешение изображения Ассоциации

DMNElement	Изображение
Ассоциация без associationDirection	
Ассоциация с associationDirection с одного конца	
Ассоциация с associationDirection с обоих концов	

ПРИЛОЖЕНИЯ

Все Приложения приводятся здесь в ознакомительных целях.

В приложении А обсуждаются вопросы, связанные с применением **DMN** совместно с **BPMN**. Целью этого раздела является задать общее направление для практикующих пользователей, однако сам раздел не является нормативным.

В приложении Б содержится ненормативный глоссарий, призванный помочь читателям лучше понять данную спецификацию.

Эта страница намеренно оставлена пустой.

Приложение А: Взаимодействие с BPMN

(Приводится в ознакомительных целях)

1. Цели BPMN и DMN

Стандарт OMG «Нотация и модель бизнес-процессов» (Business Process Model and Notation) предоставляет стандартную систему условных обозначений (нотацию) для описания бизнес-процессов в виде оркестровки задач. Успех **BPMN** стал основным мотивирующим фактором в создании **DMN.** Ожидается, что бизнес-решения, описанные с использованием **DMN**, будут широко использоваться в бизнес-процессах, описанных с помощью **BPMN**.

Все утверждения, приводимые ниже и относящиеся к **BPMN**, являются выдержками из документа OMG (formal/11-01-03), если не указано иное.

Цели **BPMN** приводятся в спецификации и их легко можно сравнить с целями **DMN**:

- Цель 1: «*Основная цель BPMN заключается в предоставлении единой нотации, понятной всем бизнес-пользователям, начиная с бизнес-аналитиков, создающих первые варианты процессов, разработчиков, ответственных за внедрение технологий выполнения бизнес-процессов, и заканчивая представителями деловых кругов, которые будут управлять этими бизнес-процессами и контролировать их выполнение. Таким образом, нотация BPMN позволяет преодолеть разрыв между моделированием бизнес-процессов и их реализацией*»**.** Пользователями **DMN** также будут бизнес-аналитики (разработка решения), а затем и представители деловых кругов (заполнение данными таких моделей решений, как таблицы решений). Технические разработчики могут привлекаться на этапе сопоставления условий бизнеса с соответствующими технологиями сбора, передачи и обработки данных. Таким образом, про **DMN** также можно сказать, чтобы данная нотация позволяет ликвидировать разрыв между моделированием решения (бизнес-аналитик) и его реализацией, как правило, с использованием каких-либо технологий исполнения решения;
- Цель 2: «*Обеспечить возможность визуализации языков XML, предназначенных для исполнения бизнес-процессов, таких как WSBPEL (Web Services Business Process Execution Language), с помощью бизнес-ориентированной нотации*». Визуализация других языков XML (таких как W3C RIF или OMG PRR) не является заявленной целью **DMN**; ожидается, что **DMN** предоставит уровень спецификации MDA для таких языков. Однако это не исключает использования **DMN** (например, таблиц решений) для представления исполняемых форм (таких как правила производства);
- Цель 3: «*Нотация **BPMN** была задумана как средство стандартизации модели и нотации бизнес-процессов в условиях, когда существует множество различных нотаций и точек зрения на моделирование. Таким образом, **BPMN** предоставляет простой способ передачи информации о процессе другим бизнес-пользователям, разработчикам процессов, заказчикам и исполнителям*». Аналогичным образом целью **DMN** является стандартизация модели и нотации принятия решений для различных инструментов моделирования со схожей семантической моделью. Тем самым **DMN,** подобно BPMN, облегчит обмен информацией о решениях между бизнес-сообществами и инструментами.

2. Задачи в BPMN и Решения в DMN

Большинство диаграмм **BPMN** содержат задачи, которые включают в себя принятие решений. Такие задачи можно смоделировать в **DMN**. Входные данные, полученные или сгенерированные ранее в процессе, обрабатываются в таких задачах, а полученные в результате обработки решения, используются в дальнейшем в этом процессе. Результаты принятия решений могут использоваться двумя основными способами:

- в другой процессной задаче;

- результаты могут влиять на выбор последовательности потоков из шлюза.

В последнем случае решения используются для определения подпроцессов или задач, которые должны выполняться далее (определяется ход процесса). Таким образом, **DMN** дополняет **BPMN** подобно тому, как моделирование решений дополняет процесс моделирования бизнес-процессов (определение оркестровок или рабочих задач).

Например, на рисунке А.1 приводится пример [1] процесса, описанного с помощью нотации BPMN.

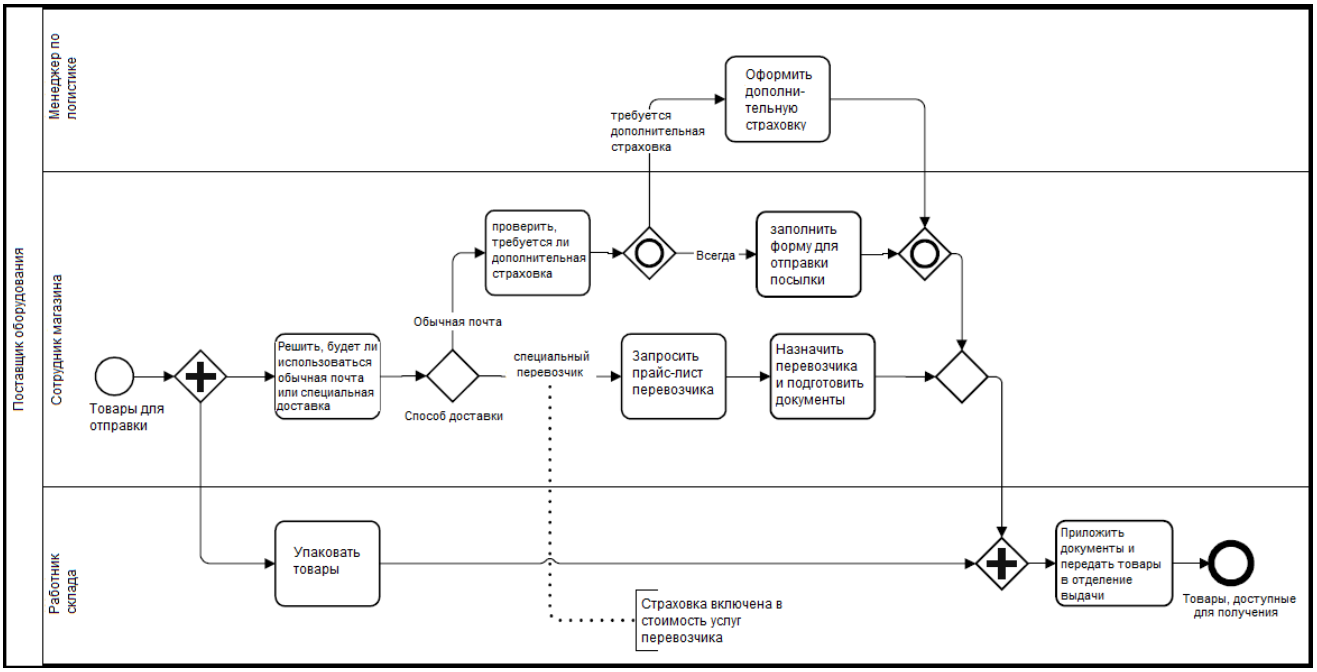


Рисунок А.1: Процесс принятия решения в BPMN

Проанализировав данный пример, мы видим, что процесс включает в себя:

- задачу, название которой начинается с «Решить ...», в ходе выполнения которой принимается решение об (использовании) обычной или специальной почты. Данная задача предшествует исключающему шлюзу, который использует результат этого решения для маршрутизации процесса;
- задачу, название которой начинается с «Проверить ...», в ходе выполнения которой принимается решение о необходимости дополнительного страхования. Данная задача предшествует включающему шлюзу, который на основе результата решения может отправить процесс по дополнительной ветке;
- задачу, название которой начинается с «Назначить ...», которая подразумевает выбор перевозчика на основе некоторых критериев. В ходе предыдущей задачи происходит сбор данных для принятия этого решения. В автоматизированной системе это, вероятно, будет подпроцесс, включающий решение и некоторые другие действия (например, «подготовка документов»).

Из этого примера видно, что даже простой бизнес-процесс в BPMN может иметь несколько задач принятия решений.

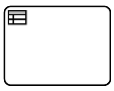
---

[1] Shipment Process in a Hardware Retailer example, Ch5.1, BPMN 2.0 By Example, June 2010, OMG reference 10-06-02

3. Типы задач BPMN, релевантных DMN

BPMN определяет [1] различные типы задач, которые могут быть использованы для принятия решений. Соответствующие задачи приведены в таблице 104:

Таблица 104. Задачи BPMN, которые могут быть использованы в DMN

	Тип задачи	Роль решения
1	Цикл Множественное выполнение Компенсация 	Роль явно не обозначена. Хотя процесс принятия решения может выполняться итерационно или зацикливаться (например, производственные правила, выполняющие циклы «Выполнение до завершения» на основе алгоритма Rete), такие типы задач не считаются релевантными на уровне бизнес-моделирования.
2	 Сервисная задача	Решение будет выполняться (если оно автоматизировано) службой принятия решений. Тем не менее, автоматическое исполнение модели принятия решений в бизнес-процессе не гарантируется.
3	 Пользовательская задача	Задачи принятия решения, выполняемые вручную как часть бизнес-процесса, ориентированного на рабочий процесс, могут быть указаны как пользовательская задача.
4	 Бизнес-правило	Бизнес-правило определяется в **BPMN 2** как местозаполнитель для решений (на основании бизнес-правил) и является естественным заполнителем для задачи принятия решения. *Обратите внимание, что бизнес-правила (как определено в OMG SBVR) могут ограничивать любые виды операций в процессе, а не только бизнес-решения.*
5	 Сценарий	Сегодня задачи решения могут быть закодированы с использованием языков сценариев бизнес-процессов.

Возможно, в следующей версии **BPMN** определение задачи будет уточнено и расширено для обеспечения наиболее полного соответствия требованиям моделирования и **DMN**. Иными словами, задача решения **BPMN** будет определяться как некая задача, используемая для принятия решения, смоделированного при помощи **DMN.** Пока этого не произошло, бизнес-правило является наиболее естественным способом для реализации данного функционала. Однако, как отмечено в разделах 5.2.2 Моделирование требований к автоматизированному принятию решений и 6.3.6 Метамодель Artifact, решение в DMN может быть связано с любой Задачей, что обеспечивает гибкость в реализации.

[1] См. главу 10.2.3 в Спецификации **BPMN**.

4. Шлюзы в процессе и решения

Шлюзы процесса можно условно разделить на два типа:

1. Шлюз, который определяет маршрут(ы) процесса на основе существующих данных.
2. Шлюз, который определяет маршрут (ы) процесса на основе результата одного или нескольких решений, полученных в ходе выполнения предыдущей задачи в процессе.

В последнем случае задача принятия решения (задача принятия решения с использованием **DMN**) может потребовать расширенной нотации для более полного определения взаимосвязи задачи решения со шлюзом (ами), которые его используют.

5. Связь моделей BPMN и DMN

DMN предлагает нормативный и ненормативный подход к связыванию моделей бизнес-процессов BPMN с моделями принятия решений.

1. Создание связей между решениями, задачами и процессами.

Согласно разделу и 6.3.6 Метамодель Artifact в **DMN 1.3**, контекст процесса для экземпляра решения `Decision` определяется его связью с любым количеством `usingProcesses`, которые являются экземплярами `Process`, в соответствии OMG **BPMN 2,** и с любым количеством `usingTasks`, которые являются экземплярами `Task`, в соответствии OMG **BPMN 2**. Таким образом, каждое решение может быть связано с одним или несколькими бизнес-процессами (чтобы указать, что решение принято во время исполнения этих процессов) и/или с одной или несколькими конкретными задачами (чтобы указать, что эти задачи включают принятие решения). ПО для моделирования ДОЛЖНО предусматривать возможность определить эти связи для каждого решения.

ПО для моделирования МОЖЕТ предусматривать валидацию обеих моделей (BPMN и DMN), например, с целью проверки того, что:

- решение не связано с задачами, которые являются частью Процессов, не связанных с Решением;
- решение не связано с задачами, которые не являются частью какого-либо Процесса, связанного с Решением.

Во время разработки может оказаться целесообразным связать решение только с процессом, но при этом не допускается несогласованность между ассоциациями «Задачи» и «Процесса».

Стоит отметить, что этот подход позволяет определять и проверять отношения между моделями бизнес-процессов и моделями решений, но сам по себе не позволяет автоматически исполнять решения, смоделированные в **DMN**, в процессах, смоделированных в **BPMN**.

2. Службы решений.

Один из ненормативных подходов к автоматизации принятия решений описывается в Приложении A: инкапсуляция решений **DMN** в «службе решения», вызванной из задачи **BPMN** (например, из служебной задачи или бизнес-правила, как описано в Приложении A.3 выше). Свойства `usingProcesses` и `usingTasks` позволяют определять и проверять ассоциации между **BPMN** и **DMN**. Определение служб решений затем предоставляет подробную спецификацию требуемого интерфейса.

Приложение Б: Глоссарий

(Приводится в ознакомительных целях)

А	
Агрегация	Вычисление единого результата на основании **нескольких результатов таблицы принятия решений**. **DMN** определяет четыре оператора агрегации в политике выбора Collect, а именно: + (sum), < (min), > (max), # (count). Если оператор не указан, результаты политики выбора Collect возвращаются без агрегирования.
В	
Вертикальная ориентация	Ориентация таблиц решений, при которой правила принятия решений представлены в виде столбцов, а условия – в виде строк.
Входная запись	**Выражение,** определяющее ячейку условия в **таблице решений** (т. е. пересечение **правила принятия решения** и **раздела** "вход").
Входное выражение	**Выражение,** определяющее элемент, который нужно сравнить с **входными записями** во входном **условии** в **таблице решений**.
Входное значение	**Выражение,** определяющее ограниченный диапазон ожидаемых значений для входного **условия** в **таблице решений**.
Входные данные	Информация, используемая в качестве входа одним или несколькими решениями, которая определяется вне **модели принятия решения**.
Вызов	Механизм, позволяющий оценивать одно выражение при помощи другого такого выражения, используя ряд **привязок**.
Выражение	**Литеральное выражение**, **таблица решений**, **вызов,** список, **контекст**, определение функции или **отношение**, используемые для определения части **логики решения** для **модели принятия решений** в **DMN.** Возвращает одно значение при интерпретации.
Выходная запись	**Выражение,** определяющее ячейку заключения в **таблице решений** (т. е. пересечение правила принятия решения и выходного **условия).**
Выходное значение	**Выражение,** определяющее ограниченный диапазон значений домена для выходного **условия** в **таблице принятия решений**.
Г	
Горизонтальная ориентация	Ориентация **таблиц решений**, в которых **правила принятия решений** представлены в виде строк, а **условия** – в виде столбцов.
График требований принятия решений	График, содержащий элементы "**Решения**", "**Модели бизнес-знаний**" и "**Входные данные**", связанные **требованиями.**
ГТР	См. **График требований принятия решений**.

Д

Диаграмма требований принятия решений	Диаграмма, являющаяся (отфильтрованным) представлением **ГТР**.
ДТР	См. **Диаграмму требований принятия решений**.

З

Запись контекста	Одна пара "ключ-значение" в **контексте.**

И

Информационный элемент	**Элемент DMN**, используемый для моделирования **переменной** или **параметра** на **уровне логики решения** в **моделях решений DMN**.
Источник знаний	Источник полномочий, определенный для **решений** или **моделей бизнес-знаний.** Например, эксперты в предметной области, ответственные за определение решений и моделей бизнес-знаний или поддержание их в актуальном виде; или первичные документы, на основании которых строятся модели бизнес-знаний; или набор тестовых данных, которым должны соответствовать решения.

К

Контекст	В **FEEL** – карта пар "ключ-значение", называемых **записями контекста**.
Кросс-таблица	**Ориентация таблиц решений**, в которой два **входных выражения** образуют два измерения таблицы, а **выходные записи** образуют двумерную сетку.

Л

Литеральное выражение	Текст, определяющий **логику принятия решения** и описывающий, как выходное значение получается из входных значений. Текст может быть составлен на английском языке или с использованием языка выражений по умолчанию **FEEL.**
Логика принятия решений	Логика, используемая для принятия решений, которые определяются в **DMN** как **выражение значения решений** и **моделей бизнес-знаний** и визуально представляются в виде **табличных выражений**.

М

Модель бизнес-знаний	Некая **логика принятия решения** (например, таблица решений), инкапсулированная как многократно используемая функция, которая может быть вызвана **решениями** или **другими моделями бизнес-знаний**.
Модель принятия решений	Формальная модель области принятия решений, выраженная в **DMN** в виде **решения** и **логики принятия решений.**

Н

Набор элементов	Используется для определения названных групп **элементов ГТР** в **Определениях.**

О

Один результат	Тип **таблицы решений**, которая может возвращать **выходную запись** только одного **правила принятия решений.**
Определение элемента	Используется для моделирования структуры и диапазона значений **входных данных** и результатов **решений** с использованием языка типа, например, **FEEL** или XML Schema.
Определения	Контейнер для всех элементов **модели решения DMN**. Обмен файлами **DMN** всегда происходит через одно или несколько определений.
Организационная единица	**Элемент бизнес-контекста**, представляющий единицу организации, которая принимает **решение** или является его владельцем.
Ориентация	Стиль представления **таблицы решений**: горизонтальная ориентация (правила принятия решений располагаются в строках, условия – в столбцах), вертикальная ориентация (правила принятия решений располагаются в столбцах, условия – в строках) или кросс-таблица (правила составлены из двух входных измерений).
Отношение	Разновидность **табличного выражения**, показывающая вертикальный список однородных горизонтальных **контекстов** (без ячеек результатов) с именами, которые появляются только один раз в верхней части списка как реляционная таблица.

П

Переменная	Представляет собой входное значение **решения**, используемое в описании **логики принятия решения** или значение, которое передается как **параметр** функции.
Подграфик требований	Ориентированный график, полученный в результате транзитивного замыкания требований **элемента ГТР**; то есть подграфик **ГТР,** представляющий все процессы принятия решений, обязательные для конкретного элемента.
Показатель эффективности	**Элемент бизнес-контекста**, представляющий собой показатель эффективности бизнеса, на который влияет **решение.**
Политика выбора	Указывает, как следует интерпретировать перекрывающиеся **правила принятия решений**. **Таблица с одним результатом** возвращает результат применения только одного правила, таблица с множественными результатами может возвращать результаты (или агрегированный результат) применения нескольких правил.
Правило принятия решений	В **таблице решений –** правило принятия решения определяет совокупность выводов или результатов (**выходных записей**) с набором условий (**входных записей**).
Привязка	Привязка в **вызове** – это связь параметров вызываемого выражения с входными переменными вызывающего выражения с использованием формулы привязки.

Р

Результат	В таблице решений – результат успешного сопоставления всех **входных выражений правила принятия решений**.
Решение	Процесс определения **выходного значения** из числа **входных значений** с использованием **логики принятия решения**, определяющей, каким образом это происходит.

С

Синтаксически корректный	При описании э**лемента ГТР** или **Требования** указывает, что такой элемент или требование соответствует ограничениям на целостность ссылочных данных, ацикличность и т. д.
Служба принятия решений	Программный компонент, инкапсулирующий **модель принятия решения** и представляющий ее как службу, которая может быть использована (например) задачей в модели процесса **BPMN.**

Т

Таблица принятия решений	Табличное представление набора связанных входных и выходных выражений, организованных в виде **правил принятия решений**, указывающих, какая **выходная запись** применяется к определенному набору **входных записей**.
Таблица решений с множественными результатами	Тип **таблицы решений**, которая может возвращать **выходные записи** из нескольких **правил принятия решений**.
Табличная функция	Разновидность **табличного выражения**, демонстрирующая вид, параметры и тело функции.
Табличное выражение	Графическое изображение, которое позволяет разбить модель **логики решения** на небольшие составляющие, которые могут быть связаны с артефактами **ДТР.**
Табличное литеральное выражение	Разновидность **табличного выражения**, показывающая **литеральное выражение.**
Табличный вызов	Разновидность **табличного выражения**, показывающая привязки параметров, которые предоставляют контекст для оценки тела **модели бизнес-знаний**.
Табличный контекст	Разновидность **табличного выражения**, демонстрирующая набор пар n («имя – значение») с необязательным значением результата.
Табличный список	Разновидность **табличного выражения**, показывающая список из n элементов.
Точка принятия решения	Точка в бизнес-процессе, в которой происходит принятие решений. В **BPMN 2.0** графически изображается в виде бизнес-правила и, может быть реализована как вызов **службы решения**.
Требование	Зависимость одного **элемента ГТР** от другого. Различают **требование к информации, требование к знаниям**, либо **требование к полномочиям**.
Требование к информации	Зависимость **решения** от элемента **входных данных** или другого **решения**, которая обеспечивает наличие переменной, используемой в логике **принятия решения**.
Требования к знаниям	Зависимость **модели принятия решений** или **бизнес-знаний** от **модели бизнес-знаний**, которая должна использоваться при оценке логики **принятия решений**.
Требования к полномочиям	Зависимость одного элемента графика требований принятия решений от другого элемента, который предоставляет инструкции или служит источником знаний первого элемента.

У

Условие	В **таблице решений** условие определяет объект, который задается входным выражением или областью выходных данных, и конечный

	набор поддоменов предметной области, которые релевантны одной из составляющих **логики решения**, описываемой таблицей решения.
Уровень логики принятия решений	Детальный уровень моделирования в **DMN,** состоящий из **выражений значений**, связанных с **решениями** и **моделями бизнес-знаний**.
Уровень требований к решениям	Более абстрактный уровень моделирования в **DMN**, состоящий из **ГТР,** представленного в виде одной или нескольких **ДТР.**
Ф	
Формальный параметр	Именованное, типизированное значение, используемое при вызове функции для передачи **информационного элемента** для дальнейшего использования в теле функции.
Э	
Элемент бизнес-контекста	Элемент, представляющий бизнес-контекст решения: либо **организационное подразделение**, либо **показатель эффективности.**
Элемент ГТР	Любой компонент ГТР: **решение, модель бизнес-знаний**, **входные данные** или **источник знаний**.
Элемент DMN	Любой элемент модели решения DMN: элемент ГТР, элемент бизнес-контекста, выражение, определения, набор элементов, информационный элемент или определение элемента.
Any	**Политика выбора** для **таблиц решений с одним результатом** с перекрывающимися **правилами принятия решений**: в рамках этой политики результатом может стать любое совпадение
Collect	**Политика выбора** для **таблиц решений с множественными результатами** с перекрывающимися **правилами принятия решений**: в рамках этой политики все совпадения будут возвращаться в виде списка в произвольном порядке. Оператор может быть добавлен для указания функции, которая будет применяться к выходам: см. **Агрегация**.
FEEL	“Friendly Enough Expression Language”(Достаточно дружественный язык выражений), который является языком выражения по умолчанию для DMN.
First	**Политика выбора** для **таблиц решений с одним результатом** с перекрывающимися **правилами принятия решений**: в соответствии с этой политикой, результатом будет являться первое совпадение, согласно порядку применения **правил**.
Output Order	**Политика выбора** для **таблиц решений с множественными результатами** с перекрывающимися **правилами принятия решений**: в соответствии с этой политикой все совпадения будут возвращены в виде списка в порядке убывания приоритета. Приоритеты задаются в упорядоченном списке значений.
Priority	**Политика выбора** для **таблиц решений с одним результатом** с перекрывающимися **правилами принятия решений**: в рамках данной политики, результатом является совпадение с самым высоким приоритетом вывода. Приоритеты вывода задаются в упорядоченном списке значений.
Rule Order	**Политика выбора** для **таблиц принятия решений с множественным выходом** с перекрывающимися **правилами принятия решения**: в рамках этой политики все совпадения будут возвращены в виде списка в порядке определения правил принятия решений.

S-FEEL	Простое подмножество **FEEL** для **моделей решений,** которые используют только простые **выражения:** в частности, **модели принятия решений**, где **логика решения** моделируется в основном или только с использованием **таблиц решений.**
Unique	**Политика выбора** для **таблиц решений с одним результатом** в которых не допускается перекрытие правил, и все **правила принятия решений** являются исключительными. Можно применить только одно правило.