Н. М. Рашевский, Т. В. Ерещенко

Модели информационных процессов и систем

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

На правах рукописи

Н. М. Рашевский, Т. В. Ерещенко

Модели информационных процессов и систем

Учебное пособие

Волгоград
2021

УДК

Рецензенты:

Печатается по решению редакционно-издательского совета
Волгоградского государственного технического университета

Рашевский, Н. М.
Модели информационных процессов и систем : учебно-методическое пособие / Н. М. Рашевский, Т. В. Ерещенко ; ВолгГТУ. – Волгоград, 2021. – 73 с.
ISBN 978-5-9948-0000-0

Учебное пособие предназначено для подробного ознакомления с подходами к моделированию процессов и систем. Представленные материалы могут быть использованы студентами очной и заочной форм обучения по направлениям 09.03.02 «Информационные системы и технологии», 09.04.02 «Информационные системы и технологии» (профиль «Искусственный интеллект в проектировании городской среды»), 08.04.01 «Строительство» (профиль «Организация информационного моделирования в строительстве»), а также специальностей ПГС, СУЗиС, ПСК.

Ил. 9. Табл. 0. Библиогр.: 55 назв.

ISBN 978-5-9948-0000-0

ОГЛАВЛЕНИЕ

Глава 1. О системах

1.1. Свойство эмерджентности

Из определения «системы» (полный, целостный набор элементов (компонентов), взаимосвязанных и взаимодействующих между собой так, чтобы могла реализоваться функция системы [1]) следует, что главным свойством системы является целостность, единство, достигаемое посредством определенных взаимосвязей и взаимодействий элементов системы и проявляющиеся в возникновении новых свойств, которыми элементы системы не обладают. Это свойство эмерджентности (от англ. emerge — возникать, появляться).

Приведем несколько определений.

Эмерджентность — степень несводимости свойств системы к свойствам элементов, из которых она состоит.

Эмерджентность — свойство систем, обусловливающее появление новых свойств и качеств, не присущих элементам, входящих в состав системы. [1]

Эмерджентность — это отношение между свойствами объекта и свойствами его частей. [2]

Различные типы эмерджентности можно условно разделить на четыре типа или класса.

Тип I вообще не содержит обратной связи, только отношения “прямой связи” (Единственной ограниченной генерирующей ролью или процессом «сверху-вниз» является преднамеренное проектирование машины: каждой детали назначается определенная и фиксированная роль, и эта роль не меняется с течением времени. Поведение каждой части всегда одинаково, оно не зависит от состояний других частей и глобального состояния системы[3]).

Основная характеристика типа II простая обратная связь: (а) отрицательная (включает в себя влияние снизу-вверх и обратную связь сверху-вниз

от группы или окружающей среды. Примерами могут служить формы самоорганизации в Интернете, напр2\имер, во Всемирной паутине (WWW) и Википедия, в проектах с открытым исходным кодом, таких как Linux и Mozilla. Также существует баланс между исследованием, разнообразием и случайностью (через влияние снизу-вверх), с одной стороны, и эксплуатацией, единством и порядком (через ограничения сверху-вниз), с другой стороны[3]) или (б) положительная (форма нежелательного, негативного возникновения через положительную обратную связь - имитацию. Например, Экономическая инфляция является (отрицательным) примером положительной обратной связи: высокая заработная плата приводит к высоким ценам на продукцию, высокие цены повышают стоимость жизни, а высокая стоимость жизни увеличивает заработную плату[3]).

Тип III появляется в очень сложных системах со множеством контуров обратной связи или сложных адаптивных системах с интеллектуальными сущностями. Это класс с большим количеством внешнего влияния в процессе возникновения.

Возникновение типа IV характеризуется многоуровневым возникновением и огромным количеством разнообразия в созданной системе, т.е. число возможных состояний возникающей системы астрономическое из-за комбинаторного взрыва. Это форма возникновения, которая отвечает за структуры более высокого уровня сложности, которые даже в принципе не могут быть сведены к прямому воздействию свойств и законов элементарных компонентов. [3]

Эмерджентное свойство может проявляться, когда несколько простых сущностей (агентов) действуют в среде, формируя более сложные модели поведения как коллектив. Процессы, вызывающие эмерджентные свойства, могут происходить в наблюдаемой системе, и обычно их можно идентифицировать по характерным для них схемам накопления изменений, обычно называемым «ростом». Эмерджентное поведение может возникать из-за слож-

ных причинно-следственных связей в разных масштабах и обратной связи, известной как взаимосвязь. Само эмерджентное свойство может быть как очень предсказуемым, так и непредсказуемым и беспрецедентным и представлять собой новый уровень эволюции системы. Сложное поведение или свойства не являются свойством только одной такой сущности, и их нельзя легко предсказать или вывести из поведения сущностей более низкого уровня (Например, форма и поведение стаи птиц или стаи рыб являются хорошими примерами эмерджентных свойств).

Одна из причин, по которой эмерджентное поведение трудно предсказать, заключается в том, что количество взаимодействий между компонентами системы увеличивается экспоненциально с увеличением количества компонентов, что позволяет появиться многим новым и тонким типам поведения. Возникновение часто является продуктом определенных моделей взаимодействия. Отрицательная обратная связь вводит ограничения, которые служат для фиксации структур или поведения. Напротив, положительная обратная связь способствует изменениям, позволяя локальным вариациям перерастать в глобальные модели. Другой способ, которым взаимодействия приводят к эмерджентным свойствам, — это двухфазная эволюция. Это происходит, когда взаимодействия применяются с перерывами, что приводит к двум фазам: одна, в которой шаблоны формируются и растут, а другая, в которой они уточняются или удаляются.

С другой стороны, просто наличия большого количества взаимодействий самого по себе недостаточно, чтобы гарантировать эмерджентное поведение; многие из взаимодействий могут быть незначительными или нерелевантными, или могут компенсировать друг друга. В некоторых случаях большое количество взаимодействий может фактически препятствовать проявлению эмерджентности, создавая много «шума», чтобы заглушить любой возникающий «сигнал». Эмерджентное поведение может потребовать временной изоляции от других взаимодействий, прежде чем оно достигнет кри-

тической массы, достаточной для самоподдержки. Таким образом, не просто количество связей между компонентами способствует возникновению, но и то, как эти связи организованы. Иерархическая организация — один из примеров, который может генерировать эмерджентное поведение (бюрократия может вести себя совершенно иначе, чем отдельные отделы этой бюрократии); но эмерджентное поведение может также возникать из более децентрализованных организационных структур, таких как рынок. [4]

1.2. Связи в системах, положительная и отрицательная обратная связь

Связь - одно из важнейших системных понятий (рис. 1). Атрибутами связи являются: направленность, пространственность, сила, характер. Связь имеет особую роль. Именно она характеризует отношение системы со средой. [1]

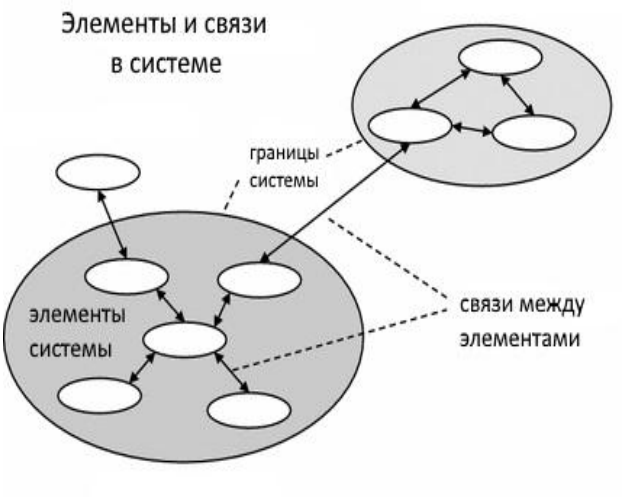


Рис. 1. Элементы и связи в системе

Обратная связь (рис. 2) - это отражение данным объектом оказанного на него прямого воздействия, направленное на источник данного воздействия.

В саморегулируемых системах обратная связь представляет собой реагирование на осуществляемые прямые воздействия. Обратную связь можно,

исходя из изложенного, определить как сигнал о реакции среды на результаты функционирования системы. Обратная связь есть функция процесса, которая предназначена для сравнения выхода с его критерием (моделью). Цель обратной связи - управление. Таким образом, обратная связь обусловливает наличие подсистемы, предназначенной для восприятия выхода с целью установления или сохранения управления. [1]

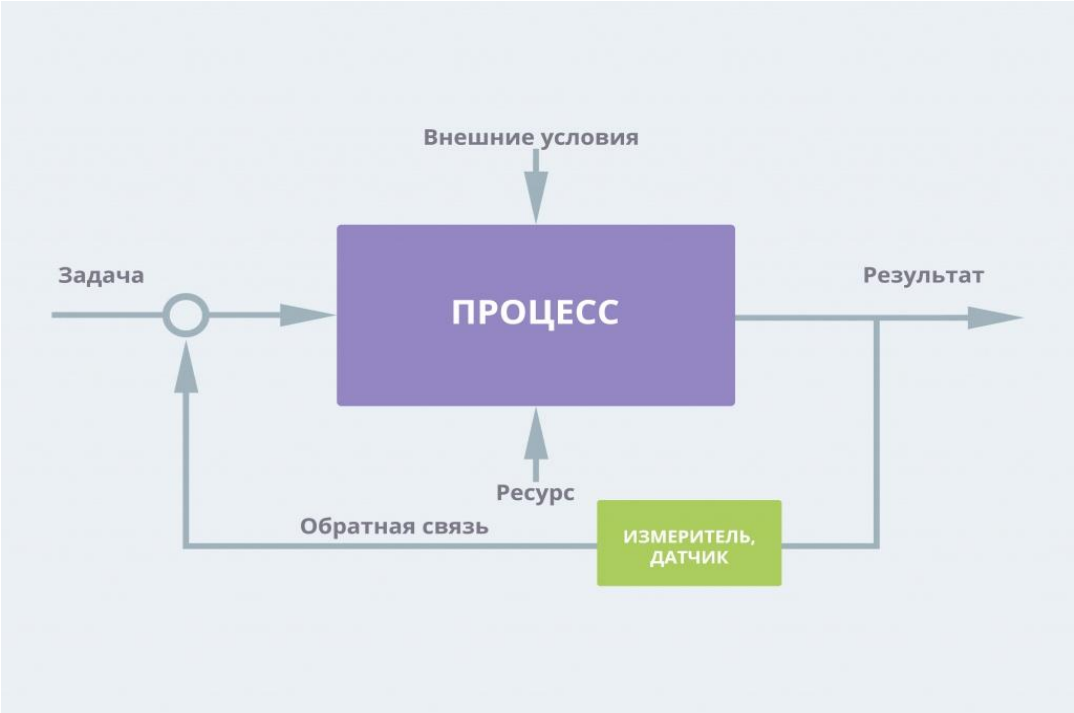


Рис. 2. Обратная связь в системе

По характеру воздействия выделяют положительные, отрицательные и нейтральные связи. Характер воздействия проявляется в изменении уровня организованности объекта воздействия, соответствующего элемента системы, а также значимости его в данной системе. Положительные связи элементов внутри системы повышают организованность и значимость элементов внутри системы, отрицательные -понижают. [1]

Положительные и отрицательные обратные связи можно проиллюстрировать, рассмотрев устройство, разработанное людьми: паровой двигатель. Паровые двигатели обычно обладают "регулятором". Это устройство, в кото-

ром клапан управляется вращающимися весовыми рычагами, вращение которых приводится в действие паровым двигателем. Поскольку рычаги вращаются быстрее, увеличенная центробежная сила заставляет их несколько закрывать клапан и таким образом замедлять двигатель. Когда двигатель замедляется, рычаги регулятора вращаются медленнее и поэтому несколько открывают клапан; таким образом, двигатель ускоряется. Это динамический баланс между открытием клапана (‘положительная обратная связь") и закрытие клапана ("отрицательная обратная связь"), что позволяет паровому двигателю поддерживать постоянную скорость. Это взаимодействие между положительными и отрицательными обратными связями позволяет паровому двигателю обрабатывать данные о своем собственном состоянии. [2]

Обобщение понятий.

Обратная связь, уменьшающая влияние входного воздействия на выходную величину, называется отрицательной, а увеличивающая это влияние — положительной. То есть положительная (усиливающая) обратная связь усиливает тенденцию изменения выхода системы, а отрицательная (уравновешивающая) — ее уменьшает. [3]

1.3. Система и модель системы

В Большой Советской Энциклопедии даётся следующее определение: «Система – объективное единство закономерно связанных друг с другом предметов, явлений, а также знаний о природе и обществе». В понятии система объективное и субъективное составляют диалектическое единство. Следовательно, систему надо рассматривать как единое целое, и исследовать ее функции также необходимо в совокупности, во взаимовлиянии. Любая система не может быть изолирована, она всегда является составной частью другой более крупной системы.

Признаки системы:

– целостность – определённая независимость системы от внешней среды и от других систем;

– связанность, т.е. наличие связей, которые позволяют посредством переходов по ним от элемента к элементу соединить два любых элемента системы, простейшими связями являются последовательное и параллельное соединения элементов, положительная и отрицательная обратные связи;

– наличие целей (функций, возможностей), не являющихся простой суммой подцелей (подфункции, возможностей) элементов, входящих в систему;

– эмерджентность – несводимость (степень несводимости) свойств системы к сумме свойств ее элементов.

Все системы не могут существовать сами по себе, они существуют в окружающей среде. Можно говорить о том, что окружающей средой является все, что не относится к рассматриваемой системе (рис.1). [1]

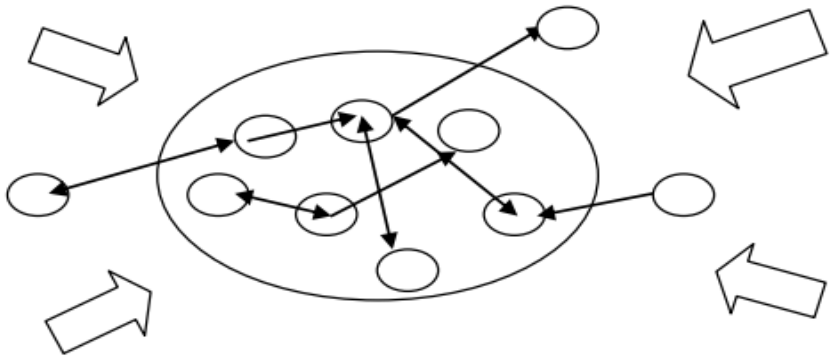


Рис. 1. Графическое изображение системы и внешней среды

Системное моделирование - это междисциплинарное исследование использования моделей для концептуализации и построения систем в бизнесе и IT-разработке.

Распространенным типом моделирования систем является функциональное моделирование с использованием специальных методов, таких как блок-схема функционального потока и IDEF0. Эти модели могут быть рас-

ширены с помощью функциональной декомпозиции и могут быть связаны с моделями требований для дальнейшего разделения систем.

В отличие от функционального моделирования, другим типом моделирования систем является архитектурное моделирование, которое использует архитектуру систем для концептуального моделирования структуры, поведения и других представлений системы.

Нотация моделирования бизнес-процессов (BPMN), графическое представление для определения бизнес-процессов в рабочем процессе, также может рассматриваться как язык моделирования систем. [2]

Моделированием называют способ, прием познания, позволяющий с

помощью одной системы, чаще всего, искусственной воспроизвести в необходимом объеме и с требуемой точностью исследуемые стороны, свойства другой более сложной системы, являющейся объектом исследования.

Модель - это физическая или абстрактная система, воспроизводящая

объект исследования и удобная для проведения экспериментов.

Удобство проведения исследований может определяться различными

факторами: легкостью и доступностью получения информации, сокращением

сроков и уменьшением материальных затрат на исследование и др.

Рассмотрим краткую классификацию видов моделирования систем

(рис. 2).

Рис. 2. Классификация видов моделирования систем

Различают моделирование физическое и математическое.

Физическое моделирование предполагает, что в качестве модели используется либо сама исследуемая система (например, в случае производственного эксперимента), либо другая система с той же или подобной физической природой. Обычно изготавливается макетный или опытный образец объекта, проводятся испытания, в процессе которых определяются его выходные параметры и характеристики, оцениваются надежность функционирования и степень выполнения технических требований, предъявленных к объекту. Если вариант технической разработки оказался неудачным, все повторяется сначала, то есть осуществляется повторное проектирование, изготовление опытного образца, испытания и т.д. Примером такого физического моделирования является продувка моделей самолетов в аэродинамических трубах. Понятно, что физическое моделирование сопряжено с большими временными и материальными затратами.

Под математическим моделированием понимается процесс установления соответствия данной реальной системы некоторой математической модели и исследование этой модели, позволяющее получить характеристики реальной системы. Математическое моделирование может быть как аналитическим, так и компьютерным.

Для аналитического моделирования характерно то, что процесс

функционирования элементов системы записывается в виде некоторых математических соотношений (алгебраических, интегральных, разностных и т.д.) или логических условий. Аналитическая модель может исследоваться: аналитически, когда стремятся получить явные зависимости для искомых характеристик системы; численно, когда, не умея решать уравнения, стремятся получить численные результаты, при конкретных исходных и начальных условиях; качественно, когда, не имея решения в явном виде, можно найти некоторые свойства решения (например, оценить устойчивость решения).

Математическая модель приближенно описывает реальный процесс,

явление или объект с помощью математических соотношений. Математические модели могут представлять собой системы дифференциальных уравнений (обыкновенных или в частных производных), системы алгебраических уравнений, разностные уравнения, линейные, нелинейные уравнения и т.д.

Компьютерное моделирование можно разделить на три вида: численное, имитационное, статистическое.

Для компьютерного моделирования характерно, что математическая

модель системы представлена в виде программы на ЭВМ или компьютерной

модели, позволяющей проводить с ней вычислительные эксперименты. При

численном моделировании для построения компьютерной модели используются методы вычислительной математики, а вычислительный эксперимент заключается в численном решении некоторых математических уравнений при заданных значениях параметров и начальных условиях. Имитационное моделирование – это вид компьютерного моделирования, для которого характерно воспроизведение на ЭВМ (имитация) процесса функционирования исследуемой системы. При этом имитируются элементарные явления,

составляющие процесс, с сохранением их логической структуры, последовательности протекания во времени, что позволяет получить информацию о состоянии системы в заданные моменты времени. Статистическое моделирование - это вид компьютерного моделирования, позволяющий получить статистические данные о процессах в моделируемой системе. [3]

Вопросы по материалам раздела

1.1.1. Организации как сложные социотехнические системы.

1.1.2. Свойство эмерджентности.

1. Что такое эмерджентность?
2. Назовите и охарактеризуйте типы эмерджентности
3. Где проявляется свойство эмерджентности?

1.1.3. Связи в системах, положительная и отрицательная обратная связь.

1. Объясните что такое связь в системах.
2. Какими бывают связи по направленности и характеру связи?
3. Зачем нужна обратная связь в системах?
4. Приведите в пример систему с использованиям обратной положительной и обратной отрицательной связью.

1.1.4. Система и модель системы.

1. Назовите типы моделирования систем.
2. Какие основные признаки имеет система?
3. Что означает целостность системы?
4. Какие виды моделирования систем существуют?
5. Как могут быть представлены математические модели?

Рекомендуемая литература

1.1.1. Организации как сложные социотехнические системы.

1.1.2. Свойство эмерджентности.

1. Родионов И. Б. Теория систем и системный анализ – Казань, Изд-во КГТУ/КГТИ, 2006.

2. Elly Vintiadis. Emergence – URL: https://iep.utm.edu/emergenc/#SH2a

3. Jochen Fromm. Types and Forms of Emergence – Germany, Universität Kassel – URL: https://arxiv.org/ftp/nlin/papers/0506/0506028.pdf

4. Emergence (From Wikipedia, the free encyclopedia) – URL: https://en.wikipedia.org/wiki/Emergence#Emergent_properties_and_processes

1.1.3. Связи в системах, положительная и отрицательная обратная связь.

1. Платформа "Bstudy - статьи для высших учебных заведений". – URL: https://bstudy.net/923703/ekonomika/svyazi_sistemy_vidy

2. Д. Фавис-Мортлок. "Трактат по геоморфологии". – 2013. – URL: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/positive-feedback

3. helpiks.org, – "Обратная связь в системе", – 2022. – URL: https://helpiks.org/6-8585.html

1.1.4. Система и модель системы.

1. Горлушкина Н.Н. Системный анализ и моделирование информационных процессов и систем. – СПб: Университет ИТМО, 2016. – 120 с.

2. Lajos Kollár, Nóra Sterbinszky Case study in system development – Notes, 2014.

3. Компьютерное моделирование: учебное пособие / В.В. Паничев, Н.А. Соловьев – Оренбург: ГОУ ОГУ, 2008 – 130 с.

Глава 2. Методология структурного анализа SADT\IDEF0

2.1. История создания

2.1.1. История создания SADT

SADT возникла в конце 60-х годов в ходе революции, вызванной структурным программированием. Когда большинство специалистов билось над созданием программного обеспечения, немногие старались разрешить более сложную задачу создания крупномасштабных систем, включающих как людей и машины, так и программное обеспечение, аналогичных системам, применяемым в телефонной связи, промышленности, управлении и контроле за вооружением. В то время специалисты, традиционно занимавшиеся созданием крупномасштабных систем, стали осознавать необходимость большей упорядоченности. Таким образом, разработчики начали формализовать процесс создания системы, разбивая его на следующие фазы:

анализ - определение того, что система будет делать, проектирование - определение подсистем и их взаимодействие, реализация - разработка подсистем по отдельности, объединение - соединение подсистем в единое целое, тестирование - проверка работы системы, установка - введение системы в действие, функционирование - использование системы.

Эта последовательность всегда выполнялась итерационно, потому что система полностью никогда не удовлетворяла требованиям пользователей, поскольку их требования часто менялись. И, тем не менее, с этой моделью создания системы, ориентированной на управление, постоянно возникали сложности. Эксплуатационные расходы, возникавшие после сдачи системы, стали существенно превышать расходы на ее создание и продолжали расти с огромной скоростью из-за низкого качества исходно созданной системы. Некоторые считали, что рост эксплуатационных расходов обусловлен характе-

ром ошибок, допущенных в процессе создания системы. Исследования показали, что большой процент ошибок в системе возник в процессе анализа и проектирования, гораздо меньше их было допущено при реализации и тестировании, а цена (временная и денежная) обнаружения и исправления ошибок становилась выше на более поздних стадиях проекта. [1]

Например, исправление ошибки на стадии проектирования стоит в 2 раза, на стадии тестирования - в 10 раз, а на стадии эксплуатации системы - в 100 раз дороже, чем на стадии анализа. На обнаружение ошибок, допущенных на этапе анализа и проектирования, расходуется примерно в 2 раза больше времени, а на их исправление - примерно в 5 раз, чем на ошибки, допущенные на более поздних стадиях. Кроме того, ошибки анализа и проектирования обнаруживались часто самими пользователями, что вызывало их недовольство.

Традиционные подходы к созданию систем приводили к возникновению многих проблем. Не было единого подхода. Привлечение пользователя к процессу разработки не контролировалось. Проверка на согласованность проводилась нерегулярно или вообще отсутствовала. Результаты одного этапа не согласовывались с результатами других. Процесс с трудом поддавался оценкам, как качественным, так и количественным. Утверждалось, что когда создатели систем пользуются методологиями типа структурного программирования и проектирования сверху вниз, они решают либо не поставленные задачи, либо плохо поставленные, либо хорошо поставленные, но неправильно понятые задачи. Кроме того, ошибки в создании систем становились все менее доступны выявлению с помощью аппаратных средств или программного обеспечения, а наиболее катастрофические ошибки допускались на ранних этапах создания системы. Часто эти ошибки были следствием неполноты функциональных спецификаций или несогласованности между спецификациями и результатами проектирования. Проектировщики знали, что сложность систем возрастает и что определены они часто весьма слабо. Рост объ-

ема и сложности систем является жизненной реалией. Эту предпосылку нужно было принять как неизбежную. Но ошибочное определение системы не является неизбежным: оно - результат неадекватности методов создания систем.

Согласно Левитту (2000), SADT является «частью серии структурированных методов, которые представляют собой набор методов анализа, проектирования и программирования, которые были разработаны в ответ на проблемы, с которыми столкнулся мир программного обеспечения с 1960-х по 1980-е годы. период времени большая часть коммерческого программирования выполнялась на COBOL и Fortran , затем на C и BASIC . Было мало указаний по «хорошим» методам проектирования и программирования, и не было стандартных методов документирования требований и проектов. Системы становились все больше и сложнее, и разработка информационных систем становилась все труднее и труднее, как способ помочь управлять большим и сложным программным обеспечением[4][2].

Методы, подобные SADT, на начальных этапах создания системы позволяли гораздо лучше понять рассматриваемую проблему. А это сокращает затраты как на создание, так и на эксплуатацию системы, а кроме того, повышает ее надежность. SADT - это способ уменьшить количество дорогостоящих ошибок за счет структуризации на ранних этапах создания системы, улучшения контактов между пользователями и разработчиками и сглаживания перехода от анализа к проектированию. [1]

SADT был разработан и испытан в полевых условиях в период с 1969 по 1973 год Дугласом Т. Россом и SofTech, Inc. [2] [3]. Первое ее крупное приложение было реализовано в 1973 г. при разработке большого аэрокосмического проекта, когда она была несколько пересмотрена сотрудниками SofTech, Inc. В 1974 г. SADT была еще улучшена и передана одной из крупнейших европейских телефонных компаний. Появление SADT на рынке произошло в 1975 г. после годичного оформления в виде продукта. К 1981 г.

SADT уже использовали более чем в 50 компаниях при работе более чем над 200 проектами, включавшими более 2000 людей и охватывавшими дюжину проблемных областей, в том числе телефонные сети, аэрокосмическое производство, управление и контроль, учет материально-технических ресурсов и обработку данных.[1] В 1981 году был опубликован формализм IDEF0 , основанный на SADT, который еще сильнее увеличил популярность данной модели. [5]

Вопросы по материалам раздела

1. Какие причины вызвали необходимость появления новых методов проектирования систем?
2. Кем и когда была разработана методология SADT?

“Рекомендуемая литература”

1. Методология структурного анализа и проектирования SADT URL: http://www.interface.ru/fset.asp?Url=/case/sadt0.htm
2. D. Marca, C. McGowan, Structured Analysis and Design Technique, McGraw-Hill, 1987Gavriel Salvendy (2001). Handbook of Industrial Engineering: Technology and Operations Management.. p.508.
3. D. T. Ross: Structured Analysis (SA): A Language for Communicating Ideas. IEEE Transactions on Software Engineering, SE-3(1), pp. 16-34.
4. Dave Levitt (2000):Introduction to Structured Analysis and Design, Archived
5. Gavriel Salvendy (2001). Handbook of Industrial Engineering: Technology and Operations Management.. p.508.

2.1.2. История создания IDEF0

Постоянное усложнение производственно-технических и организационно-экономических систем – фирм, предприятий, производств, и др. субъек-

тов производственно-хозяйственной деятельности - и необходимость их анализа с целью совершенствования функционирования и повышения эффективности обусловливают необходимость применения специальных средств описания и анализа таких систем. Эта проблема приобретает особую актуальность в связи с появлением интегрированных компьютеризированных производств и автоматизированных предприятий.

В США это обстоятельство было осознано еще в конце 70-ых годов, когда ВВС США предложили и реализовали Программу интегрированной компьютеризации производства ICAM (ICAM - Integrated Computer Aided Manufacturing), направленную на увеличение эффективности промышленных предприятий посредством широкого внедрения компьютерных (информационных) технологий.

Реализация программы ICAM потребовала создания адекватных методов анализа и проектирования производственных систем и способов обмена информацией между специалистами, занимающимися такими проблемами. Для удовлетворения этой потребности в рамках программы ICAM была разработана методология IDEF (ICAM DEFinition), позволяющая исследовать структуру, параметры и характеристики производственно-технических и организационно-экономических систем (в дальнейшем, там, где это не вызывает недоразумений – систем). Общая методология IDEF состоит из трех частных методологий моделирования, основанных на графическом представлении систем:

• IDEF0 используется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции.

• IDEF1 применяется для построения информационной модели, отображающей структуру и содержание информационных потоков, необходимых для поддержки функций системы;

• IDEF2 позволяет построить динамическую модель меняющихся во времени поведения функций, информации и ресурсов системы.

К настоящему времени наибольшее распространение и применение имеют методологии IDEF0 и IDEF1 (IDEF1X), получившие в США статус федеральных стандартов. [1][2][3].

Методология IDEF0, особенности и приемы применения которой описываются в настоящем Руководящем документе (РД), основана на подходе, разработанном Дугласом Т. Россом в начале 70–ых годов и получившем название SADT (Structured Analysis & Design Technique - метод структурного анализа и проектирования). Основу подхода и, как следствие, методологии IDEF0, составляет графический язык описания (моделирования) систем, обладающий следующими свойствами.

• Графический язык - полное и выразительное средство, способное наглядно представлять широкий спектр деловых, производственных и других процессов и операций предприятия на любом уровне детализации.

• Язык обеспечивает точное и лаконичное описание моделируемых объектов, удобство использования и интерпретации этого описания.

• Язык облегчает взаимодействие и взаимопонимание системных аналитиков, разработчиков и персонала изучаемого объекта (фирмы, предприятия), т.е. служит средством «информационного общения» большого числа специалистов и рабочих групп, занятых в одном проекте, в процессе обсуждения, рецензирования, критики и утверждения результатов.

• Язык прошел многолетнюю проверку и продемонстрировал работоспособность как в проектах ВВС США, так и в других проектах, выполнявшихся государственными и частными промышленными компаниями.

• Язык легок и прост в изучении и освоении.

• Язык может генерироваться рядом инструментальных средств машинной графики; известны коммерческие программные продукты, поддерживающие разработку и анализ моделей - диаграмм IDEF0, например, про-

дукт Design/IDEF 3.7 (и более поздние версии) фирмы Meta Software Corporation (США).

Перечисленные свойства языка предопределили выбор методологии IDEF0 в качестве базового средства анализа и синтеза производственно-технических и организационно-экономических систем, что нашло свое отражение в упомянутых федеральных стандартах США.

В связи с расширяющимся применением информационных технологий и, в частности, CALS-технологий в народном хозяйстве Российской Федерации в настоящем РД приводятся основные сведения о методологии IDEF0 и графическом языке описания моделей, а также некоторые практические рекомендации по разработке таких моделей.[3]

Вопросы по материалам раздела

1. На что Программа интегрированной компьютеризации производства ICAM была направлена?

2. Для чего используется IDEF0?

Рекомендуемая литература

1. INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0). Draft Federal Information Processing Standards Publication 183 ,1993 December 21.

2. INTEGRATION DEFINITION FOR INFORMATION MODELING (IDEF1X), Draft Federal Information Processing Standards Publication 184 1993 December 21.

3. МЕТОДОЛОГИЯ ФУНКЦИОНАЛЬНОГО МОДЕЛИРОВАНИЯ IDEF0. URL: https://advanced-quality-tools.ru/assets/idef0-rus.pdf

2.2. Базовые элементы нотации

Нотация IDEF0 – используют для документирования процессов производства и отображения информации об использования ресурсов на каждом из этапов проектирования систем. На (рис. 1) представлена графическая диаграмма и нотации IDEF0 - пример реализован в системе Business Studio, которая включает в себя функции программы для построения IDEF0. [1]

Бизнес-процессы в нотации IDEF0 представляются в форме прямоугольника, а стрелки отражают связь с другими процессами и внешней средой. Особенностью нотации является:

– Возможность декомпозировать процессы и, таким образом, строить иерархические модели бизнес-процессов;

– Выделение четырех типов стрелок: три типа входов – вход, управление и механизм (это позволяет более гибко описывать логику использования входов в процессе в целях последующего анализа), и выход.

1. Вход (Input) – материал или информация которые используется или преобразуется блоком для получения результата (выхода).
2. Выход (Output) – результат выполнения функции (материал или или информация). Данный вид дуг выходит из правой стороны блока.
3. Управление (Control) – условия, правила, стратегии, стандарты, которые влияют на выполнение функции. Данный вид дуг поступает на верхнюю сторону блока.
4. Механизм (Mechanisms) – ресурсы, с помощью которых выполняется работа. Это могут быть, например, денежные средства, персонал предприятия, станки. Данный вид дуг поступает на нижнюю сторону блока. [1]

Вопросы по материалам раздела

1. Куда поступают дуги механизма?
2. В какой форме представляется IDEF0?
3. Что является особенностью нотации?

Рекомендуемая литература

1. Марка Д.А., МакГоун К. Методология структурного системного анализа и проектирования SADT:Пер. с англ. – С.: Метатехнология, 2003.

2. Димов Э.М., Диязитдинова А.Р., Качков Д.А. Проектирование информационных систем: Учебное пособие. – Самара: ПГАТИ, 2003. – 78 с.

2.3. Правила моделирования

Моделирование бизнес процессов основывается на ряде принципов, которые дают возможность создать адекватные модели процессов. Их соблюдение позволяет описать множество параметров состояния процессов таким образом, чтобы внутри одной модели компоненты были тесно взаимосвязаны, в то время как отдельные модели оставались в достаточной степени независимыми друг от друга.

Главными принципами моделирования бизнес процессов являются следующие:

Принцип декомпозиции – каждый процесс может быть представлен набором иерархически выстроенных элементов. В соответствии с этим принципом процесс необходимо детализировать на составляющие элементы.

Принцип сфокусированности – для разработки модели необходимо абстрагироваться от множества параметров процесса и сфокусироваться на ключевых аспектах. Для каждой модели эти аспекты могут быть свои.

Принцип документирования – элементы, входящие в процесс, должны быть формализованы и зафиксированы в модели. Для различных элементов процесса необходимо использовать различающиеся обозначения. Фиксация элементов в модели зависит от вида моделирования и выбранных методов.

Принцип непротиворечивости – все элементы, входящие в модель процесса должны иметь однозначное толкование и не противоречить друг другу.

Принцип полноты и достаточности – прежде чем включать в модель тот или иной элемент, необходимо оценить его влияние на процесс. Если элемент не существенный для выполнения процесса, то его включение в модель не целесообразно, т.к. он может только усложнить модель бизнес-процесса. [1]

Правила SADT включают:

Ограничение количества блоков на каждом уровне декомпозиции (правило 3–6 блоков, ограничение когнитивной сложности, т.к. человек при большем количестве факторов начинает их интуитивно делить на группы);

Связность диаграмм (композиция в нумерации блоков);

Уникальность меток и наименований (отсутствие повторяющихся имен);

Синтаксические правила для графики (блоков и дуг);

Разделение входов и управлений (правило определения роли данных).

Отделение организации от функции, т.е. исключение влияния организационной структуры на функциональную модель.[2]

Правила IDEF0

Диаграмма, лежащая на вершине иерархии, называется контекстной.

На этой диаграмме вся система представляется в виде единого функционального блока.

Следующей в иерархии является диаграмма декомпозиции контекстной диаграммы. На ней функциональный блок контекстной диаграммы декомпо-

зируется на составляющие его функциональные блоки. Каждый из этих блоков может иметь свою диаграмму декомпозиции.

Количество блоков на каждом уровне декомпозиции ограничено (может быть от 3 до 6)

Диаграммы связаны по номерам блоков;

Метки и наименования уникальны;

Входы и управления разделены по роли данных;

Исключено влияние организационной структуры на функциональную модель. [3]

Вопросы по материалам раздела

В чем суть принципа декомпозиции?

Назовите хотя бы 2 правила SADT

Назовите хотя бы 2 правила IDEF0

Рекомендуемая литература

1.KPMS 2007-2022 https://www.kpms.ru

2.Электронный конспект лекций, Григорьев А.В., Донецьк, ДонНТУ – 2010.

3.http://www.idef.com

2.4. Достоинства и недостатки методологии

Методология IDEF0 может использоваться для моделирования широкого круга систем и определения требований и функций, а затем для разработки системы, которая удовлетворяет этим требованиям и реализует эти функции. Для уже существующих систем SADT может быть использована

для анализа функций, выполняемых системой, а также для указания механизмов, посредством которых они осуществляются.

Не смотря на то, что в настоящее время появляются десятки новых методологий моделирования деятельности предприятия и взглядов на ее архитектуру, IDEF0 сохраняет актуальность для задач совершенствования предприятий и организаций.

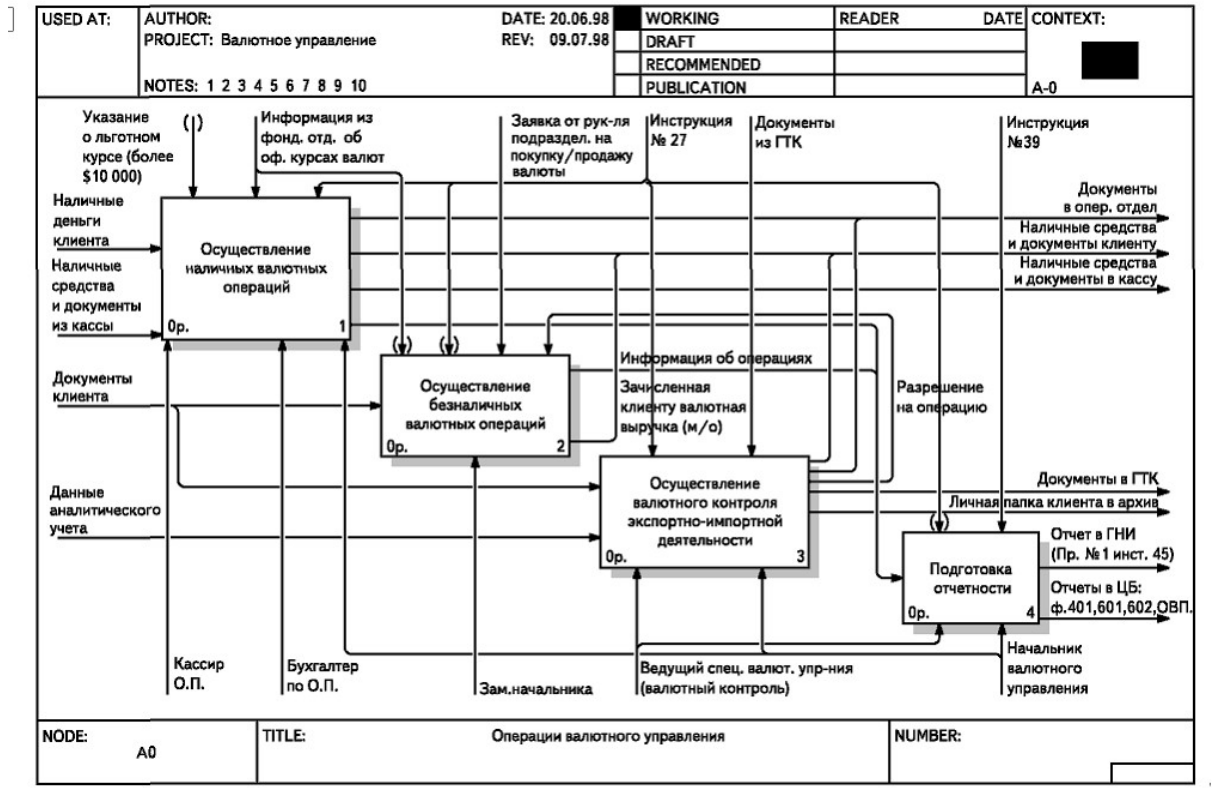


Рис. 1. Пример IDEF0 модели

Модель IDEF0 разворачивается одновременно слева направо и сверху вниз, по диагонали. Объекты, расположенные левее / выше, доминируют над теми, которые находятся правее / ниже. Доминирующие объекты могут включать в себя зависимые: например, доставка заказа – это элемент, входящий в состав более масштабного процесса управления заказами. Также доминирующие объекты могут являться предшествующими этапами для зависимых: получение заявки – согласование заявки. [1]

Графические элементы:

Прямоугольники – действия или этапы.

Стрелки – ресурсы, исполнители, необходимые для совершения действия или прохождения этапа.

Главное достоинство IDEF0 – крайне высокая степень детализации, можно создать модель, которая будет учитывать на каждом этапе практически все ресурсы, сотрудников, которые потребуются даже для самых сложных алгоритмов. Недостатком является то, что графическая модель занимает очень много места, её тяжело читать, не имея специальных навыков.

Ещё один минус: с помощью IDEF0 лучше всего описываются модели, где бизнес-процесс представляет собой одну цепочку, без развилок. Если на пути он встречает множественные “или”, работать с IDEF0 становится очень сложно. [2]

Еще один недостаток IDEF0 – сложность увязки моделей нескольких процессов (например, сбыта и производства) в случае создания отдельных моделей для каждого из этих процессов. Но это скорее техническое несовершенство, которое можно устранить при помощи предварительных договоренностей о правилах моделирования. [3]

Еще некоторые достоинства IDEF0:

Опыт внедрения ИС показывает, что методология IDEF0 позволяет повысить производительность труда и уменьшить вероятность появления ошибок при синтезе систем.

Использование единого языка для представления деятельности предприятия и внешней среды позволяет получать процессные модели, которые отражают точку зрения потребителя.

Глобальная информатизация общества только усиливает спрос на возможности, которые обеспечиваются IDEF0.

Долгая история его использования для решения различных задач государственных и коммерческих предприятий.

Продолжает использоваться и рекомендоваться в качестве стандарта описания деятельности организации и предприятия. [1]

Характеристика	Описание
Преимущества IDEF0:	· широкая известность стандарта среди аналитиков, консультантов и программистов
	· относительная простота изучения и применения стандарта при описании бизнес-процессов
	· лаконичность и высокая информативность получаемых моделей
	· стандарт соответствует требованиям ISO 9000
Недостатки IDEF0:	· возможность только линейного описания процессов - на диаграмме невозможно отразить действия, выполняемые в случае, если процесс отклонился от своего идеального варианта
	· для чтения и интерпретации диаграмм необходимы определенные знания стандарта

Рис. 2. Преимущества и недостатки IDEF0 в виде таблицы

Сергей Тимофеев. “Преимущества методологии IDEF0”. – 2021 – URL: https://itstan.ru/funk-strukt-analiz/preimushhestva-metodologii-idef0.html

Елена Гайдукова. “Нотации бизнес-процессов IDEF0. EPC. BPMN.” . – 2020 – URL: https://www.comindware.com/ru/blog-нотации-бизнес-процессов-idef0-epc-bpmn/

“Преимущества и недостатки использования IDEF0”. – 2014 – URL: https://studopedia.su/11_92013_preimushchestva-i-nedostatki-ispolzovaniya-IDEF.html

Глава 3. Модели данных

3.1. Моделирование данных

Что такое моделирование данных?

Моделирование данных-это процесс построения диаграмм потоков данных. При создании новой или альтернативной структуры базы данных разработчик начинает с диаграммы того, как данные будут поступать в базу данных и из нее. Эта блок-схема используется для определения характеристик форматов данных, структур и функций обработки баз данных для эффективной поддержки требований к потоку данных. После создания и развертывания базы данных модель данных продолжает жить, чтобы стать документацией и обоснованием того, почему существует база данных и как были спроектированы потоки данных.

Модель данных, полученная в результате этого процесса, обеспечивает структуру взаимосвязей между элементами данных в базе данных, а также руководство по использованию данных. Модели данных являются основополагающим элементом разработки программного обеспечения и аналитики. Они обеспечивают стандартизированный метод для последовательного определения и форматирования содержимого базы данных в разных системах, позволяя различным приложениям совместно использовать одни и те же данные. [1]

Модели данных строятся с учетом потребностей бизнеса. Правила и требования определяются заранее с помощью обратной связи от заинтересованных сторон бизнеса, чтобы их можно было включить в разработку новой системы или адаптировать в ходе итерации существующей.

Данные могут быть смоделированы на различных уровнях абстракции. Процесс начинается со сбора информации о бизнес-требованиях от заинтересованных сторон и конечных пользователей. Эти бизнес-правила затем преобразуются в структуры данных для разработки конкретного дизайна базы данных. Модель данных можно сравнить с дорожной картой, планом архитектора или любой формальной диаграммой, которая облегчает более глубокое понимание того, что разрабатывается. [2]

Типы моделей данных.

Как и любой процесс проектирования, проектирование баз данных и информационных систем начинается с высокого уровня абстракции и становится все более конкретным и конкретным. Модели данных, как правило, можно разделить на три категории, которые различаются в зависимости от степени их абстракции. Процесс начнется с концептуальной модели, перейдет к логической модели и завершится физической моделью. Каждый тип модели данных более подробно обсуждается ниже:

- Концептуальные модели данных.

Они также называются моделями предметной области и предлагают общее представление о том, что будет содержать система, как она будет организована и какие бизнес-правила задействованы. Концептуальные модели обычно создаются в рамках процесса сбора исходных требований к проекту. Как правило, они включают классы сущностей (определяющие типы объектов, которые важно представлять бизнесу в модели данных), их характеристики и ограничения, взаимосвязи между ними и соответствующие требования к безопасности и целостности данных. Любая нотация, как правило, проста;

- Логические модели данных.

Они менее абстрактны и содержат более подробную информацию о концепциях и отношениях в рассматриваемой области. Используется одна из нескольких формальных систем обозначения моделирования данных. Они указывают атрибуты данных, такие как типы данных и их соответствующие длины, и показывают взаимосвязи между сущностями. Логические модели данных не определяют никаких технических системных требований. Этот этап часто опускается в agile или DevOps практика. Логические модели данных могут быть полезны в высокопроцессуальных средах реализации или для проектов, ориентированных на данные по своей природе, таких как проектирование хранилища данных или разработка системы отчетности;

- Физические модели данных.

Они предоставляют схему того, как данные будут физически храниться в базе данных. Как таковые, они наименее абстрактны из всех. Они предлагают доработанный дизайн, который может быть реализован в виде реляционной базыданных, включая ассоциативные таблицы, иллюстрирующие отношения между сущностями, а также первичные ключи и внешние ключи, которые будут использоваться для поддержания этих отношений. Физические модели данных могут включать специфические свойства системы управления базами данных (СУБД), включая настройку производительности.[2]

Зачем использовать модель данных?

Основной целью использования модели данных являются:

- Гарантирует, что все объекты данных, требуемые базой данных, будут точно представлены. Пропуск данных приведет к созданию ошибочных отчетов и приведет к неправильным результатам.
- Модель данных помогает проектировать базу данных на концептуальном, физическом и логическом уровнях.
- Структура модели данных помогает определить реляционные таблицы, первичные и внешние ключи и хранимые процедуры.

- Он обеспечивает четкое представление о базовых данных и может быть использован разработчиками баз данных для создания физической базы данных.
- Также полезно идентифицировать отсутствующие и избыточные данные.
- Хотя первоначальное создание модели данных является трудоемким и трудоемким, в долгосрочной перспективе это делает модернизацию и обслуживание вашей ИТ-инфраструктуры дешевле и быстрее. [3]

Инструменты для моделирования данных

Сегодня широко используются многочисленные коммерческие и CASE-решения с открытым исходным кодом, в том числе различные инструменты моделирования данных, построения диаграмм и визуализации. Вот несколько примеров:

- erwin Data Modeler — это инструмент моделирования данных, основанный на языке IDEF1X, который теперь поддерживает и другие нотации, включая нотацию для размерного моделирования.
- Enterprise Architect — это инструмент визуального моделирования и проектирования, который поддерживает моделирование корпоративных информационных систем и архитектур, программных приложений и баз данных. Он основан на объектно-ориентированных языках и стандартах.
- ER/Studio — это программа для проектирования баз данных, совместимая с некоторыми из самых популярных СУБД. Она поддерживает как реляционное, так и размерное моделирование данных.
- Бесплатные инструменты моделирования данных включают решения с открытым исходным кодом, такие как Open ModelSphere. [4]

Вопросы к теме “Моделирование данных”:

1) Что подразумевает под собой моделирование данных?
2) Сколько всего типов моделей данных?
3) Какие основные цели преследует моделирования данных?

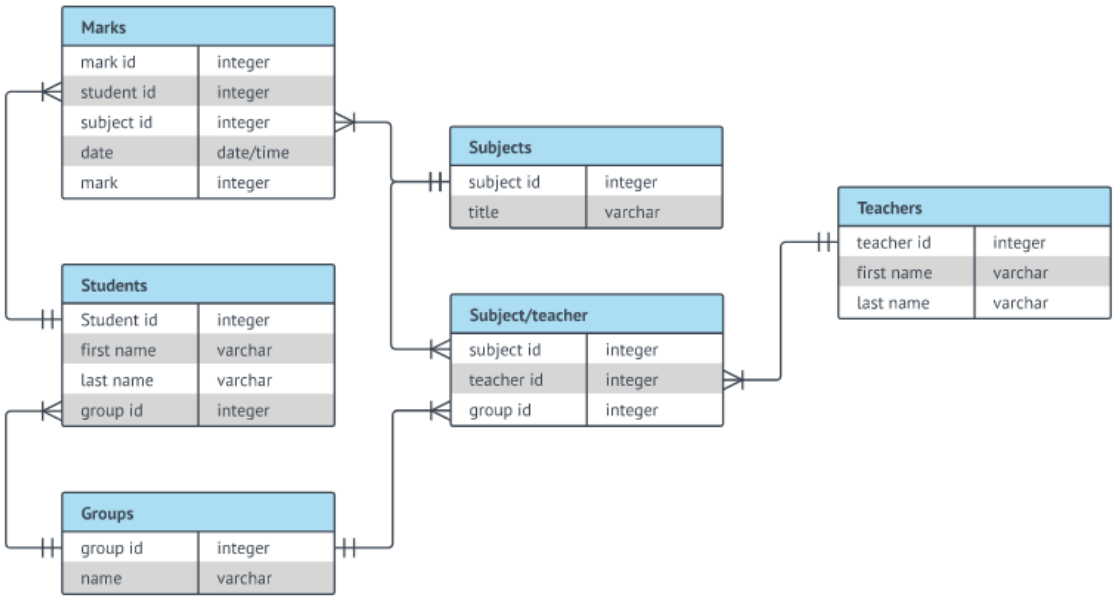
Список литературы:

1) https://www.sap.com/cis/index.html?url_id=auto_hp_red..
2) https://www.ibm.com/cloud/learn/data-modeling
3) https://www.guru99.com/data-modelling-conceptual-logical.html
4) https://habr.com/ru/post/554388/?

3.2. ER-диаграммы (Сущность-Связь), атрибуты, мощность связи

ER-диаграммы (Сущность-Связь)

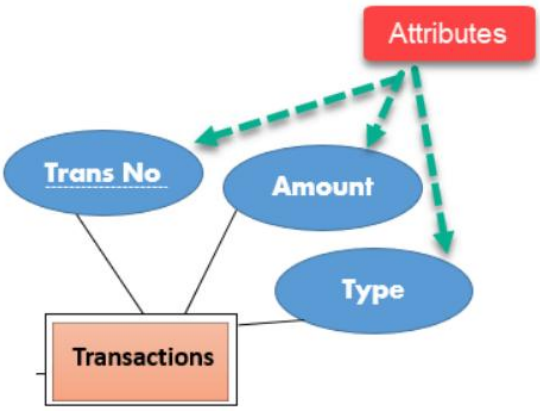
Схема «сущность-связь» (также ERD или ER-диаграмма) — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения, информационных систем для бизнеса. ER-диаграммы (или ER-модели) полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные линии, для отображения сущностей, их атрибутов и связей. Эти диаграммы устроены по тому же принципу, что и грамматические структуры: сущности выполняют роль существительных, а связи — глаголов.



Атрибуты

Это свойство с одним значением либо типа объекта, либо типа отношения.

Например, у лекции могут быть атрибуты: время, дата, продолжительность, место и т. д.



Типы атрибутов

1. Простой атрибут (Характеризует сущность, а также отношения между двумя или более элементами.)

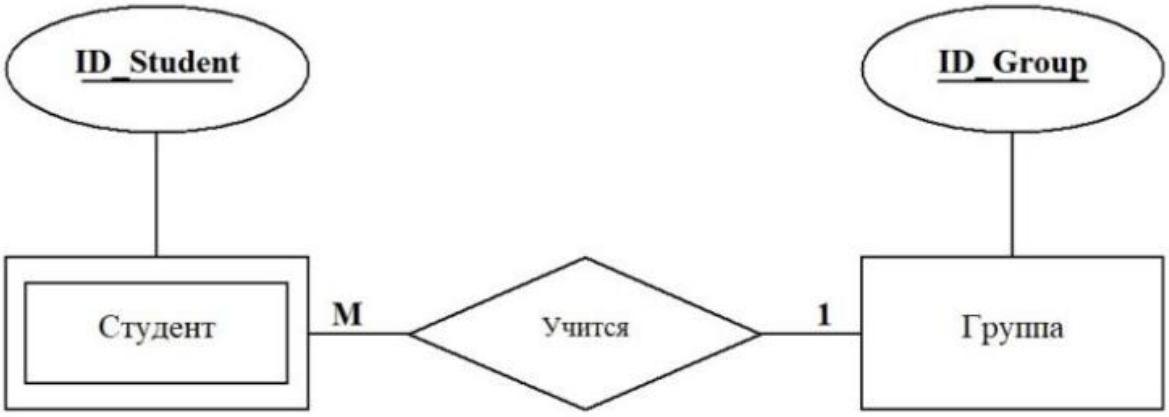
2. Составной атрибут (Составной атрибут можно разбить. Например, полное имя можно разбить на ФИО).

3. Производный атрибут (Атрибут, чье значение можно вычислить, опираясь на значения связанных с ним атрибутов. Например, возраст нельзя хранить напрямую. Вместо этого он должен быть получен из даты рождения этого сотрудника.)

4. Многозначный атрибут (Атрибут, которому может быть присвоено несколько значений. Например, у студента может быть более одного номера мобильного телефона, адреса электронной почты и т. д.)

Мощность связи

Мощность определяет числовые атрибуты связи между двумя сущностями или наборами сущностей.



Типы связей

1. Отношения «один к одному» или 1:1 (Пример: Один студент может зарегистрироваться на несколько курсов. Тем не менее, все эти курсы имеют одну линию обратно к этому одному студенту.)

2. Отношения «один ко многим» или 1:M(Например, один класс состоит из нескольких учеников.)

3. Отношения«многие к одному» или M:1 (Например, многие ученики принадлежат к одному классу.)

4. Отношения «многие ко многим» или M:N (Например, студенты как группа связаны с несколькими преподавателями, а преподаватели могут быть связаны с несколькими студентами.)

Литература :

- https://www.guru99.com/er-diagram-tutorial-dbms.html
- https://www.lucidchart.com/pages/ru/erd-%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0

Глава 4. Семейство методологий IDEF

4.1. IDEF1

Information Modeling — методология моделирования информационных потоков внутри системы, позволяющая отображать и анализировать их структуру и взаимосвязи. [1]

IDEF1X (IDEF1 Extended) — Data Modeling — методология моделирования баз данных на основе модели «сущность-связь». Применяется для построения информационной модели, которая представляет структуру информации, необходимой для поддержки функций производственной системы или среды.[1]

Метод IDEF1, разработанный Т. Рэйми (T. Ramey) на основе подходов П. Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме. В настоящее время на основе совершенствования методологии IDEF1 создана её новая версия — методология IDEF1X. Она разработана с учётом таких требований, как простота изучения и возможность автоматизации. [1]

Применение этой методологии позволяет решить следующие задачи:

• выяснить структуру и содержание существующих потоков информации;

• определить, какие проблемы вызваны недостатком управления соответствующей информацией;

• выявить информационные потоки, требующие дополнительного управления для эффективной реализации модели.[4]

Модель IDEF1 включает в рассмотрение автоматизированные компоненты, базы данных и соответствующую им информацию, реальные объекты (сотрудники, помещения и т. д.).[4]

В отличие от методов разработки структур баз данных (например, IDEF1X), IDEF1 является аналитическим методом и используется для выполнения следующих действий:

• определение самой информации и структуры её потоков;

• определение существующих правил и законов, по которым осуществляется движение информационных потоков, а также принципов управления ими;

• выяснение взаимосвязей между существующими информационными потоками;

• выявление проблем, возникающих вследствие недостатка качественного информационного менеджмента.[4]

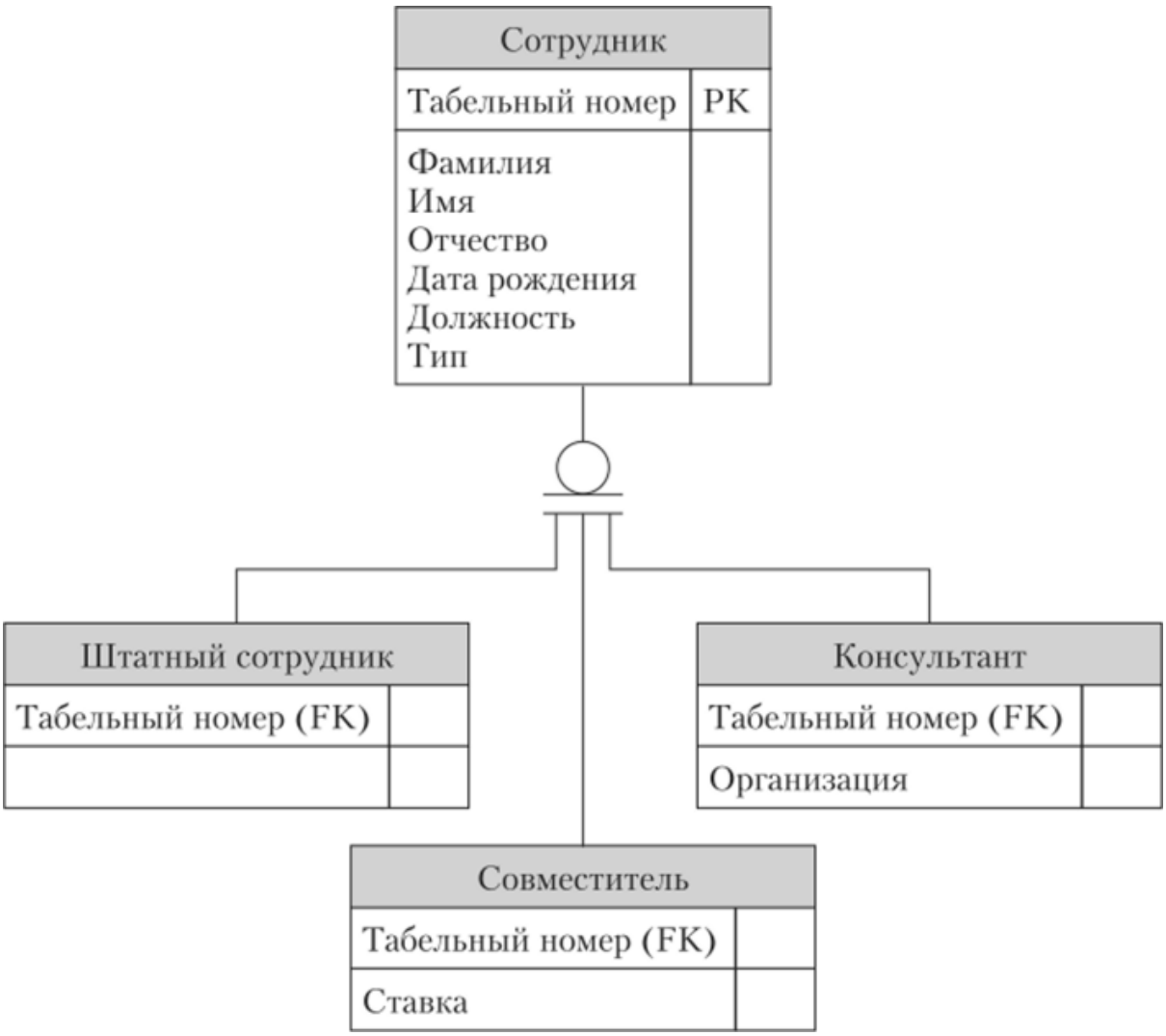


Рис 1. Пример диаграммы IDEF1X.

4.2. IDEF2

IDEF2 (Simulation Model Design) – Динамическое моделирование развития системы - данный метод позволяет построить динамическую модель меняющегося во времени поведения функций, информации и ресурсов производственной системы или среды. Данная модель используется редко. В основном востребована на предприятиях, где необходимо описать непрерывную деятельность на конвейерах или аналогичные функции.[3]

В связи с весьма серьёзными сложностями анализа динамических систем от этого стандарта практически отказались, и его развитие приостановилось на самом начальном этапе.[3]

В настоящее время присутствуют алгоритмы и их компьютерные реализации, позволяющие превращать набор статических диаграмм IDEF0 в динамические модели, построенные на базе «раскрашенных сетей Петри» (CPN - Color Petri Nets).[3]

Модель разбивается на четыре подмодели:

• подмодель возможностей, которая описывает агентов;

• подмодель потока сущностей, которая описывает трансформацию сущностей;

• подмодель распределения ресурсов, которая описывает распределение агентов для проведения трансформаций;

• подмодель управления системой, которая описывает внешние воздействия.[3]

Преимуществом методики является то, что набор диаграмм может быть непосредственно переведён в имитационную модель.[3]

Наверное, основной проблемой на пути развития и внедрения этой методологии является то, что бизнес-аналитики обычно плохо знают имитационное моделирование (и наоборот).[3]

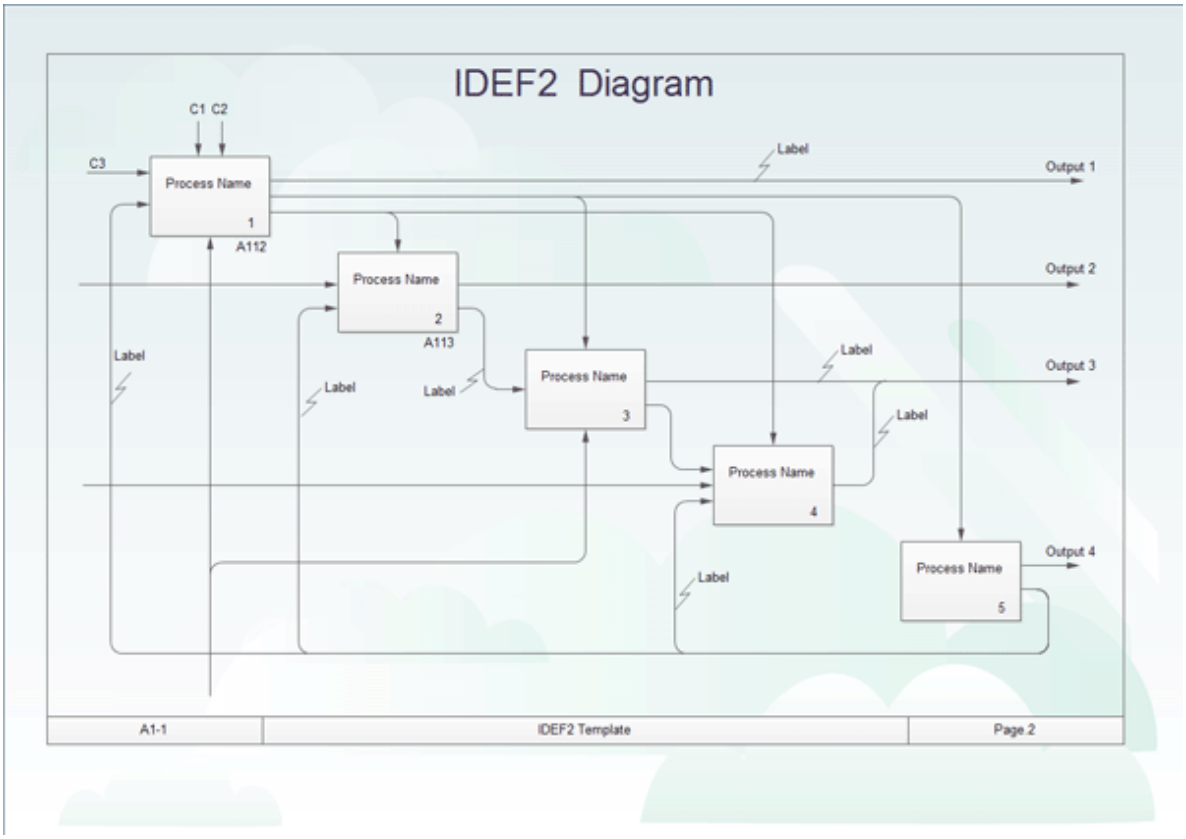


Рис 2. Пример диаграммы IDEF2.

4.3. IDEF3

Process Description Capture (Документирование технологических процессов) — методология документирования процессов, происходящих в системе (например, на предприятии), описывает сценарий и последовательность операций для каждого процесса. IDEF3 имеет прямую взаимосвязь с методологией IDEF0 — каждая функция (функциональный блок) может быть представлена в виде отдельного процесса средствами IDEF3. [1]

IDEF3 предназначена для документирования процессов системы, последовательности их операций, детализирующих функции, описанные в IDEF0. Данный подход более структурирован, чем IDEF0, и определяет схемы последовательности процессов, их ветвления и слияния, что также полезно в случае использования модели для дальнейшей работы уже с методами

имитационного анализа (рис. 4.4), где перекрестки J1 и J4 — асинхронный «И», перекрестки J2 и J3 — асинхронный «ИЛИ», 1.2—1.6 — активности.[2]

IDEF3 оперирует понятиями «единиц работы», которые объединяются но принципу временного предшествования либо использования результатов одной единицы работы в качестве входных потоков для другой.[2]

Основа методологии - сценарий (scenario) бизнес-процесса, осуществляющий описание последовательности изменений свойств объекта в рамках рассматриваемого процесса.[5]

Исполнение каждого сценария сопровождается соответствующим документооборотом, который состоит из двух основных потоков — документов, определяющих структуру и последовательность процесса, и документов, отображающих ход его выполнения.[5]

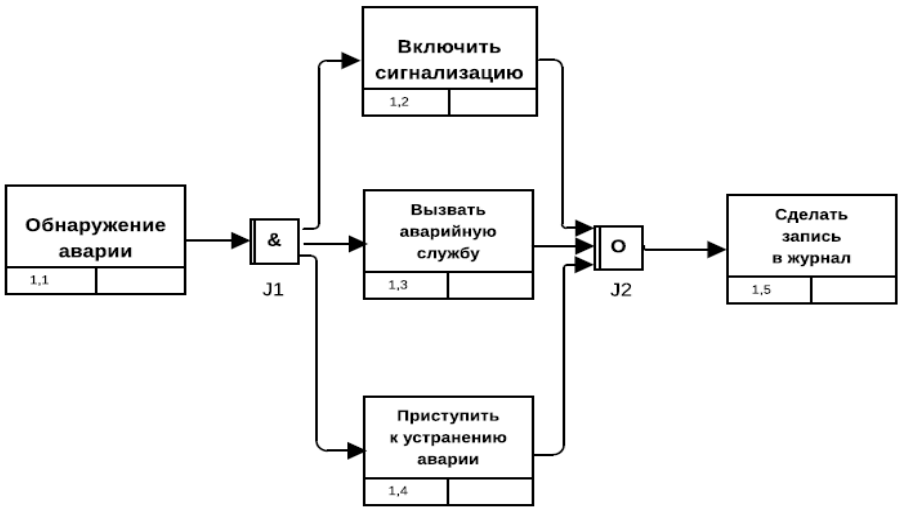


Рис 3.Пример диаграммы IDEF3

Вопросы по материалам раздела

1. Какие задачи позволяет решить применение методологии IDEF1X?

2. Почему развитие IDEF2 приостановилось на самом начальном этапе?

3. С какой методологией IDEF3 имеет прямую взаимосвязь?

Рекомендуемая литература

1. IDEF: From Wikipedia, the free encyclopedia. – URL : https://en.wikipedia.org/wiki/IDEF

2. Семейство методологий IDEF: From Studme.org. - URL: https://studme.org/184130/informatika/semeystvo_metodologiy_idef

3. IDEF1 (Information Modeling): From Студопедия. -URL: https://studopedia.ru/4_117282_IDEF-Information-Modeling.html

4. IDEF2 (Simulation Model Design): From Студопедия. -URL: https://studopedia.ru/4_117283_IDEF-Simulation-Model-Design.html

5. IDEF3 (Process Description Capture): From Студопедия. -URL: https://studopedia.ru/4_117284_IDEF-Process-Description-Capture.html

4.4 Методология построения объектно-ориентированных систем IDEF4

IDEF4 - это объектно-ориентированный метод проектирования различных систем. Он был разработан для обеспечения перехода от предметной области и требований к объектно-ориентированным моделям, где отражается их структура, принципы взаимодействия. Объекты проектируются с достаточной детализацией, что дает возможность анализировать их исходную сущность. IDEF4 обеспечивает связь и дает возможность перейти от основного анализа объектов к их детальному рассмотрению и реализации. IDEF4

обеспечивает связь между результатами базового анализа предметной области и реализацией системы.

Философия объектно-ориентированного проектирования подчеркивает, что разделение аспектов проектирования на внешние и внутренние приводит к большему успеху, потому что это позволяет повторно использовать компоненты проектирования, обеспечивать параллельное проектирование, модульность конструкции

IDEF4 - метод многомерного подхода к объектно-ориентированному программному построению системы, в котором конструкция состоит из следующих элементов:

- уровень проектирования (на уровне системы, на уровне приложений, и базовый (нижний) уровень дизайна);
- определение статуса объекта дизайна (предметная область в приложении, на этапе обеспечения связи, в программном обеспечении описания предметной области);
- построение (дизайн) моделей взаимодействия (статические, динамические и модели поведения);
- расчетное обоснование моделей и уточнение ее конструктивных особенностей, начиная от общего к частному.

IDEF4 предусматривает дизайн моделей в трех отдельных слоях:

(1) проектирование системы

(2) разработка приложений

(3) базовый (нижний) уровень дизайна (рис. 2.18).

Это трехслойная организация уменьшает сложность конструкции. При проектировании системы слой обеспечивает связь с другими системами. Слой разработки приложений отображает интерфейсы компонентов системы, которая конструируется. Эти компоненты включают в себя коммерческие приложения, ранее разработанные и реализованные, а также приложения, ко-

торые будут разработаны. Базовый (нижний) уровень дизайна представляет основные объекты системы.

IDEF4 Статус явлений и объектов, любой созданный объект в IDEF4 может быть отмечен как основной (начальный), переходный или завершенный. Это позволяет проследить процесс создания системы от начала до завершения.

IDEF4 Дизайн Моделей*:* IDEF4 использует три дизайн-модели и обоснования компонентов в моделях. Статические Модели (SM) определяют неизменяемые во времени отношения между объектами (например, наследование). Динамические Модели (DM) определяют связи между объектами и переходы между состояниями объектов. Модели Поведения (БМ) определяют отношения между соответствующими объектами в процессе их функционирования. Расчетное обоснование компонентов моделей подразумевает представление системы сверху вниз, что позволяет дать представление об охвате этих трех моделей и документах, которые используются для обеспечения процесса основного дизайна системы.

Статические модели, Динамические модели, Модели поведения и расчетное обоснование их компонент ложатся в основу базового (нижнего) уровня дизайна объектов. Вложенные слои могут быть построены в пределах каждого слоя, что позволяет уменьшить сложность систем.

IDEF4 - это многократная повторяющаяся процедура, состоящая из разделения, классификации/спецификации, сбора (соединения), моделирования и повторного перераспределения (перестановки) деятельности в рамках системы . Сначала система разделяется на объекты, каждый из которых либо сопоставляется с существующими объектами, либо для которых разрабатывается внешняя спецификация. Внешняя спецификация позволяет более корректно представить внутреннюю спецификацию.

После классификации/спецификации объекты соединяются в совокупность действий (бизнес-процессов) (т.е. статические, динамические и пове-

денческие модели подробно описывают различные аспекты взаимодействия между объектами). В процессе разработки моделей и систем важно моделировать сценарии и события для того, чтобы выявить недостатки конструкций.

На основе выявленных недостатков проектировщик может переделать существующие модели и моделировать их до тех пор, пока они не станут корректными.

У объектов есть «поведение», которое описывает роль объекта в системе и «статус», которое описывает значение характеристики объекта.

Объекты идентифицируются в описании требований приложения. Они делятся на категории:

- физический объект, например, дерево, автомобиль, самолет, человек;
- объект-роль с особенностями поведения, которая устанавливает его значение, например, медсестра, пожарник или профессор;
- событие-объект, представляющий собой возникновение действий, например, встреча, поставка и др.;
- транзакции /взаимодействие - объекты, обеспечивающие взаимодействие других объектов, например, канал связи для пересылки сообщения;
- спецификация процедуры - объект, который представляет собой набор инструкций.

IDEF4 явления и объекты могут существовать в четырех различных абстракциях: приложения и предметные области объектов-предметов, объекты в натуральном выражении, спецификации объектов и программные объекты. В первых трех стадиях объекты отражаются так, как они существуют в природе. Программные объекты - это любые характеристики явлений.

4.5 Методика IDEF5 построения онтологической модели

IDEF5 (Ontology Description Capture) – данный метод позволяет разрабатывать, изучать и поддерживать онтологию моделируемой системы. Термин «онтология» включает в себя каталог терминов области знаний; правила, объясняющие, как термины могут комбинироваться, создавая при этом корректные ситуации в области знаний и согласованные выводы, используемые в моделируемой системе.

Для поддержания процесса построения онтологий в IDEF5 существуют специальные онтологические языки:

· **Схематический язык** (Schematic Language - **SL**) – это наглядный графический язык, специально предназначенный для изложения компетентными специалистами в рассматриваемой области системы основных данных в форме онтологической информации.

· **Язык доработок и уточнений** (Elaboration Language - **EL**) представляет собой структурированный текстовой язык, который позволяет детально характеризовать элементы онтологии.

Всего существует четыре основных вида схем, которые наглядно используются для накопления информации об онтологии в достаточно прозрачной графической форме:

1. **Диаграмма классификации** обеспечивает механизм для логической систематизации знаний, накопленных при изучении системы. Существует два типа таких диаграмм:

a. Диаграмма строгой классификации (Description Subsumption - **DS**),

b. Диаграмма естественной или видовой классификации (Natural Kind Classification - **NKC**).

2. **Композиционная схема** (Composition Schematics) является механизмом графического представления состава классов онтологии и фактически представляют собой инструменты онтологического исследования по принципу «Что из чего состоит».

3. **Схемы взаимосвязей** (Relation Schematics) позволяют разработчикам визуализировать и изучать взаимосвязи между различными классами объектов в системе. В некоторых случаях схемы взаимосвязей используются для отображения зависимостей между самими же классовыми взаимосвязями.

4. **Диаграмма состояния объекта** (Object State Schemantic) позволяет документировать тот или иной процесс с точки зрения изменения состояния объекта. В происходящих процессах могут произойти два типа изменения объекта: объект может поменять свое состояние или класс.

Пример графической диаграммы:

Преимущества:

· На начальном этапе графический язык SL может быть очень полезен для формулировки начальных требований к онтологии и определения вектора разработки более подробной онтологии на текстовом языке IDEF5 или в любом другом средстве.

· В рамках IDEF5 изучение онтологии достаточно просто и понятно.

Недостатки:

· Онтология и анализ знаний о предметной области является довольно обширной и трудоемкой темой.

· Проблема графического языка в том, что с его помощью нельзя достаточно четко сформулировать некоторые отношения (аксиомы) онтологии, но для этого можно использовать текстовый язык IDEF5.

4.6. IDEF6

IDEF6 (Design Rational Capture Method) – данный метод позволяет использовать рациональный опыт проектирования. Назначение IDEF6 состоит в облегчении получения «знаний о способе» моделирования, их представ-

ления и использования при разработке систем управления предприятиями. Под «знаниями о способе» понимаются причины, обстоятельства, скрытые мотивы, которые обуславливают выбранные методы моделирования. Метод IDEF6 акцентирует внимание именно на процессе создания модели.

Основные характеристики:

· Метод позволяет обосновать необходимость проектируемых моделей, выявить причинно-следственные связи и отразить это в итоговой документации системы.

Для чего используется:

· Предназначение заключается в том, чтобы методически обосновать целесообразность проектирование информационных систем и выявить причинно-следственные связи.

· Назначение стандарта состоит в структурировании «знаний о способе» моделирования, их представления и использования при разработке информационных систем. Акцент внимания именно на процессе создания модели.

Преимущества:

· Предпринимает попытки выявления логики, лежащей в основе решения и конечного дизайна.

· Четкое установление целесообразного дизайна помогает избежать повторения прошлых ошибок, предоставляет прямые результаты последствий предлагаемых изменений в конструкции, заставляет яснее изложить цели и предположения и в результате помогает определить окончательные спецификации системы.

· Четкое выявление мотивов, почему выбран и принят конкретный проект и дизайн, стратегия реализации системы на уровне предприятия, информационных систем, является необходимым для поддержания жизненного цикла самой системы.

В соответствии с целью обоснования дизайна ставятся следующие задачи:

1. 1) обеспечить эволюционный процесс интеграции информационных систем на предприятии;
2. 2) дать возможность использовать параллельные инженерные методы в развитии информационных систем;
3. 3) поддержать наилучшую интеграцию в течение жизненного цикла системы и связанных с ней явлений;
4. 4) облегчить реинжиниринг деловых процессов путем использования решений в бизнес - кейсах;
5. 5) обеспечить эффективное отслеживание решений..

Используемая литература:

1. https://books.ifmo.ru/file/pdf/1720.pdf
2. https://infostart.ru/1c/articles/1430187/
3. https://monographies.ru/en/book/section?id=11248
4. https://ozlib.com/944493/tehnika/metodologiya_postroeniya_obektno_orientirovannyh_sistem_idef4

1.4.7 IDEF 7 Information System Auditing — Аудит информационных систем. Этот метод определён как востребованный, однако так и не был полностью разработан.

1.4.8 IDEF 8 User Interface Modeling — Метод разработки интерфейсов взаимодействия оператора и системы (пользовательских интерфейсов). Современные среды разработки пользовательских интерфейсов в большей степени создают внешний вид интерфейса. IDEF 8 фокусирует внимание разработчиков интерфейса на программировании желаемого взаимного поведения

интерфейса и пользователя на трех уровнях: выполняемой операции (что это за операция); сценарии взаимодействия, определяемом специфической ролью пользователя (по какому сценарию она должна выполняться тем или иным пользователем); и, наконец, на деталях интерфейса (какие элементы управления, предлагает интерфейс для выполнения операции)

1.4.9 IDEF 9 Business Constraint Discovery method (Scenario-Driven IS Design) — Метод исследования бизнес ограничений был разработан для облегчения обнаружения и анализа ограничений в условиях которых действует предприятие. Обычно, при построении моделей описанию ограничений, оказывающих влияние на протекание процессов на предприятии уделяется недостаточное внимание.

Знания об основных ограничениях и характере их влияния, закладываемые в модели, в лучшем случае остаются неполными, несогласованными, распределенными нерационально, но часто их вовсе нет. Это не обязательно приводит к тому, что построенные модели нежизнеспособны, просто их реализация столкнется с непредвиденными трудностями, в результате чего их потенциал будет не реализован. Тем не менее в случаях, когда речь идет именно о совершенствовании структур или адаптации к предсказываемым изменениям, знания о существующих ограничениях имеют критическое значение.

Глава 5. Методология UML

5.1. Объектно-ориентированное моделирование

Объектно-Ориентированное Моделирование-использует язык моделирования, соответствующий UML- «стандарту», и позволяет создавать модели постоянной и переменной структуры из «ориентированных» и «неориентированных» блоков, с поведением, определяемым внутренними машинами состояний.

Понятие объектно-ориентированного моделирования (ООМ), безусловно, связано с объектно-ориентированным программированием (ООП). Этот подход к разработке программных средств, появившийся в середине 1980-х гг., первоначально был направлен на разрешение проблем, возникающих в результате неизбежного роста и усложнения программ, а также задач обработки данных и манипулирования ими. В то время стало очевидным, что традиционные методы процедурного программирования не способны справиться ни с растущей сложностью программ и их разработки, ни с необходимостью повышения их надежности. При этом вычислительные и расчетно-алгоритмические задачи, особенно в области обеспечения бизнеса, постепенно стали уходить на второй план, а первое место стали занимать задачи именно обработки данных и манипулирования ими

Фундаментальными понятиями ООП являются понятия класса и объекта

Фундаментальными понятиями ООП являются понятия класса и объекта. При этом под *классом* понимается некоторая абстракция совокупности *объектов*, которые имеют общий набор свойств и обладают одинаковым поведением. Каждый объект в этом случае рассматривается как экземпляр соответствующего класса. Объекты, которые не имеют полностью одинаковых свойств или не обладают одинаковым поведением, по определению, не могут

быть отнесены к одному классу. Хотя приведенное определение класса может быть уточнено на основе учета других понятий ООП, оно является общим и достаточным для проведения ООМ.

Важная особенность классов состоит в возможности их организации в виде некоторой иерархической структуры, которая по внешнему виду напоминает схему классификации понятий формальной логики. В связи с этим следует отметить, что каждое понятие в логике имеет объем и содержание. Под объемом понятия понимают все другие мыслимые понятия, для которых исходное понятие может служить определяющей категорией или частью. Содержание же понятия составляет совокупность всех его признаков и атрибутов, отличающих данное понятие от всех других. В формальной логике известен закон обратного отношения: если содержание понятия А содержится в содержании понятия В, то объем понятия В содержится в объеме понятия А.

Иерархия понятий строится следующим образом. В качестве наиболее общего понятия или категории берется понятие, имеющее наибольший объем и, соответственно, наименьшее содержание. Это самый высокий уровень абстракции для данной иерархии. Затем общее понятие некоторым образом конкретизируется, тем самым уменьшается его объем и увеличивается содержание. Появляется менее общее понятие, которое на схеме иерархии будет расположено на уровень ниже исходного понятия. Этот процесс конкретизации понятий может быть продолжен до тех пор, пока на самом нижнем уровне не будет получено понятие, дальнейшая конкретизация которого невозможна либо нецелесообразна.

Объект — это элемент реального мира в объектно-ориентированной среде, который может иметь физическое или концептуальное существование. Каждый объект имеет —

- Идентичность, которая отличает его от других объектов в системе.
- Состояние, определяющее характерные свойства объекта, а также

значения свойств, которыми обладает объект.

- Поведение, которое представляет видимые извне действия, выполняемые объектом, с точки зрения изменения его состояния.

Объекты можно моделировать в соответствии с потребностями приложения.

Вопросы:

1 Как вы поняли для себя понятие объектно-ориентированное моделирование?

2 Какое физическое или концептуальное существование имеет каждый объект ?

Рекомендуемая литература

1. Wikipedia 1
2. МОДЕЛИРОВАНИЕ ПРОЦЕССОВ И СИСТЕМ 2
3. Объектно-Ориентированное Моделирование в среде Rand Model Designer 3

Инструменты моделирования

В настоящее время существует ряд инструментов моделирования, поддерживающих полноценную технологию ООМ и не ориентированных на какую-то узкую прикладную область. Эти инструменты можно разделить на две группы:

1) инструменты, ориентированные на язык UML и, очень важную его часть, формализм машин состояния (гибридных автоматов);

2) инструменты, ориентированные на язык Modelica (механизмы, поддерживающие технологию «физического моделирования).

К первой группе относятся такие среды моделирования как Ptolemy (университет Беркли, США), AnyLogic (The AnyLogic Company, СанктПетербург) и Rand Model Designer (исследовательская группа MVSTUDIUM Group, СПбГПУ) . Сюда же следует отнести «вечно молодую» среду Matlab, которая сейчас является сложно устроенной системой компонентов «MATLAB+Simulink+StateFlow+ToolBoxes». Признать среду Matlab объектно-ориентированной можно только с большими оговорками – полноценно технологию ООМ поддерживает лишь подсистема StateFlow. Однако это не мешает пакету Matlab занимать ведущее место в современном практическом моделировании.

Ко второй группе относятся среды, разработанные в рамках европейского проекта «Modelica», такие как Dymola, OpenModelica, MathModelica и другие. Мы считаем среды, основанные на технологии ОММ, наиболее перспективными, и будем в дальнейшем говорить только о них. Современная среда моделирования, претендующая на универсальность, должна позволять разрабатывать в рамках технологии ООМ следующие виды моделей:

- однокомпонентные непрерывные модели;
- однокомпонентные дискретно-событийные модели;
- однокомпонентные гибридные модели;
- многокомпонентные модели с непрерывными, дискретными или гибридными компонентами и ориентированными связями («блочные модели»);
- многокомпонентные модели с непрерывными, дискретными или гибридными компонентами и неориентированными связями («физические модели»);
- многокомпонентные модели с переменным составом компонентов и переменной структурой связей. [1]

Язык Modelica — это язык для моделирования киберфизических систем, поддерживающий причинно-следственную связь компонентов, управляемых математическими уравнениями, для облегчения моделирования на основе первых принципов. Он обеспечивает объектно-ориентированные конструкции, которые облегчают повторное использование моделей и может быть удобно использован для моделирования сложных систем, содержащих, например, механические, электрические, электронные, магнитные, гидравлические, тепловые, управляющие, электрические или технологические компоненты. [2]

Язык Modelica поддерживает «физическое моделирование», но использует свою концепцию объектов, не совпадающую с концепцией языка UML, а также специальные конструкции для описания дискретных событий в гибридных моделях. [1]

Унифицированный язык моделирования (UML) — это универсальный, развивающийся язык моделирования в области разработки программного обеспечения, предназначенный для обеспечения стандартного способа визуализации проектирования системы. [3]

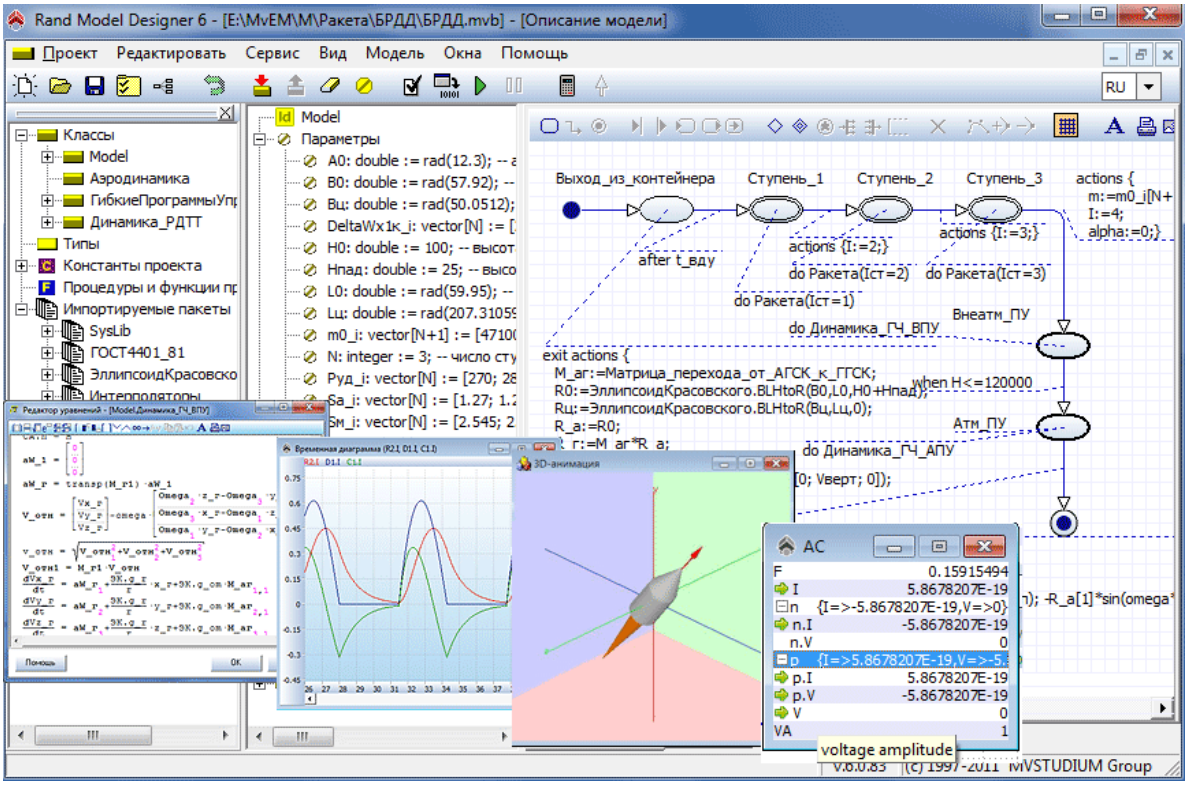
Практика моделирования показывает, что машина состояний языка UML все же гораздо удобнее и нагляднее для описания дискретно-событийных и гибридных моделей, чем описание, предложенное авторами языка Modelica, и что существует много практически значимых гибридных моделей, с которыми трудно справиться, используя язык Modelica. Кроме того, в рамках подхода Modelica достаточно сложно естественным образом моделировать системы с переменным составом. [1]

Rand Model Designer - это высокопроизводительная визуальная среда для разработки компонентных моделей сложных динамических систем. (рис. 1) Rand Model Designer использует образный, интуитивно понятный объектно-ориентированный язык моделирования высокого уровня, позволяющей быстро и качественно создавать сложные модели.

Rand Model Designer позволяет разрабатывать непрерывные, дискретные и гибридные (непрерывно-дискретные) модели и проводить интерактивные вычислительные эксперименты с разработанными моделями.

Rand Model Designer предназначен для пользователей, проводящих многочисленные сложные и трудозатратные вычислительные эксперименты, или создающих новые программные приложения, в основе которых лежат математические модели.

Продукт сочетает в себе интуитивно понятную среду объектно-ориентированного физического моделирования и мощные механизмы символьных вычислений, которые позволяют ускорять процесс подготовки качественных моделей. [4]



В инструменте Rand Model Designer сделана попытка соединить сильные стороны обоих направлений: поддерживать «физическое моделирование» как это предложено в языке Modelica, и использовать при этом объектную парадигму и машину состояний языка UML. «Расплатой» за это решение явилась необходимость выполнения части анализа совокупной системы

уравнений на стадии выполнения модели при каждом переключении. Оказалось, что этот анализ можно проводить с помощью алгоритмов «линейной сложности», и создаваемые с помощью RMD промышленные модели из компонентов с ненаправленными связями успешно работают в реальном времени. [1]

Вопросы по материалам раздела

1. На какие группы можно разделить инструменты моделирования?
2. Какие среды программирования относятся к этим группам?
3. Перечислите минусы использования языка Modelica.

Рекомендуемая литература

1. https://www.researchgate.net/publication/329784729_Object-Oriented_Modeling_with_Rand_Model_Designer
2. https://modelica.org/modelicalanguage
3. https://en.wikipedia.org/wiki/Unified_Modeling_Language
4. https://www.panvasoft.com/rus/27157/

5.2. О языке графического описания UML

UML (англ. Unified Modeling Language — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур. [3]

UML – это язык для визуализации, специфицирования, конструирования и документирования артефактов программных систем. Язык представля-

ет словарь и правила комбинирования входящих в него слов в целях коммуникации. Язык моделирования – это язык, словарь и правила которого сосредоточены на концептуальном и физическом представлении системы. UML – стандартное средство представления «чертежей» программного обеспечения. Моделирование необходимо для понимания системы. При этом ни одна модель не является абсолютно достаточной. Напротив, чтобы понять большинство систем, кроме самых тривиальных, часто требуется множество взаимосвязанных моделей. В отношении программных систем это означает, что необходим язык, средствами которого можно описать архитектуру системы с различных точек зрения, причем на протяжении всего жизненного цикла ее разработки. Словарь и правила такого языка, как UML, говорят о том, как создавать и читать хорошо согласованные модели, но не говорит о том, какие именно модели в каких случаях требуется создавать. Это задача всего процесса разработки программного обеспечения. Хорошо организованный процесс должен сам подсказать, какие потребуются рабочие продукты, какие ресурсы понадобятся для их создания и управления ими, как их использовать для оценки выполненной работы и управления проектом в целом. [2]

UML – язык визуализации

С точки зрения многих программистов, промежуток времени между размышлениями о реализации проекта и их изложением в коде стремится к нулю. Вы думаете – значит, вы кодируете. И действительно, некоторые вещи лучше всего выражаются непосредственно в коде на языке программирования, потому что текст программ – самый прямой и короткий путь написания выражений и алгоритмов. Но и в этих случаях программист на самом деле занимается моделированием, хотя и делает это мысленно. Он может даже делать наброски некоторых идей – на доске или салфетке. Однако при этом возникают некоторые проблемы. Во-первых, обсуждение таких концептуальных моделей с другими участниками разработки чревато ошибками и непо-

ниманием, если только все участники дискуссии не говорят на одном языке. Как правило, при разработке проектов предприятиям приходится создавать в этих целях свои собственные языки, и вам трудно понять, о чем идет речь, если вы посторонний или новичок в группе. Во-вторых, некоторые вещи, касающиеся программных систем, трудно выразить, пытаясь строить модели лишь средствами текстовых языков программирования. Например, назначение иерархии классов можно понять, внимательно изучив код всех классов в иерархии, но воспринять всю структуру целиком не получится. Аналогично, изучая код, можно исследовать физическое представление и распределение объектов в Web-ориентированной системе, но нельзя сразу «схватить» его целиком. В-третьих, если разработчик, который писал этот код, никогда не воплощал в нем модели, существовавшие в его голове, то информация о них может быть потеряна навсегда и в лучшем случае частично восстановлена на основе существующей реализации, когда этот разработчик перейдет на другую работу. Описание моделей на UML позволяет решить третью проблему: явная модель облегчает общение. Некоторые вещи лучше моделировать в тексте, другие – графически. В действительности во всех интересных системах существуют структуры, которые невозможно выразить на языке программирования. UML – графический язык, позволяющий решить вторую из описанных выше проблем. UML – нечто большее, чем просто набор графических символов. Каждый из этих символов имеет четко определенную семантику. И это значит, что один разработчик может описать модель на UML, а другой разработчик и даже инструментальное средство – однозначно интерпретировать ее. Это решает первую из упомянутых проблем.

UML – язык специфицирования

В данном контексте специфицирование – это построение точных, недвусмысленных и полных моделей. В частности, UML позволяет специфици-

ровать все важные решения, касающиеся анализа, дизайна и реализации, принимаемые в процессе разработки и внедрения программных систем. [2]

UML – язык конструирования

UML не является визуальным языком программирования, но его модели могут быть непосредственно ассоциированы с различными языками программирования. А это значит, что существует возможность отобразить UML-модель на такой язык, как Java, C++ или Visual Basic, а при необходимости даже на таблицы реляционной базы данных либо объекты, хранящиеся в объектно-ориентированной базе данных. Те вещи, которые проще выразить графически, выражаются на UML, а те, что легче выразить в виде текста, – на языке программирования. Отображение модели на язык программирования позволяет осуществить прямое проектирование (forward engineering) – генерацию кода на языке программирования из модели UML. Обратное также возможно: вы можете восстановить модель UML на основе существующей реализации. В обратном проектировании (reverse engineering) нет никакой магии. Если только вы не закодировали информацию в реализации, она теряется при переходе от модели к коду. Поэтому обратное проектирование, выполняемое инструментальными средствами, все же требует определенного вмешательства человека. Комбинация этих двух путей – прямого и обратного проектирования – обеспечивает возможность работы как с графическим, так и с текстовым представлениями; при этом обеспечивается согласованность между ними. В дополнение к прямому отображению UML благодаря своей выразительности и однозначности позволяет непосредственно исполнять модели, имитируя поведение проектируемых систем, а также управляя действующими системами. [3]

UML – язык документирования

Успешные компании, специализирующиеся на программном обеспечении, помимо исполняемого кода производят и другие продукты, включая следующие (но не ограничиваясь ими):

1. требования;
2. архитектуру;
3. проектные решения (дизайн);
4. исходный код;
5. проектные планы;
6. тесты;
7. прототипы;
8. релизы (версии)

В зависимости от уровня культуры разработки, принятой в компании, некоторые из этих продуктов выражаются более формально, чем другие. Перечисленные продукты – это не только поставляемые составные части проектов; они необходимы для управления, оценки результатов и взаимодействия в процессе разработки системы и после ее внедрения. UML предназначен для документирования архитектуры системы и всех ее деталей. Кроме того, это язык для выражения требований к системе и описания тестов. И наконец, он подходит для моделирования работ на этапе проектирования и управления версиями.

Где может использоваться UML?

UML прежде всего предназначен для моделирования и разработки программных систем. Наиболее эффективно его применение в следующих областях:

1) корпоративные информационные системы;
2) банковские и финансовые услуги;
3) телекоммуникации;
4) транспорт;

5) оборона, авиация и космонавтика;

6) розничная торговля;

7) медицинская электроника;

8) наука;

9) распределенные Web-сервисы.

Но сфера применения UML не ограничена моделированием программного обеспечения. Его выразительность позволяет вести работу и над непрограммными системами – в частности, продумывать документооборот юридической системы, структуру и функционирование системы здравоохранения, системы управления воздушным движением, а также проектировать аппаратные средства.

Вопросы по материалам раздела

1. Где может использоваться UML? Назовите несколько областей.
2. UML - это язык *какого* описания?
3. На какие языки существует возможность отобразить UML-модель?

Рекомендуемая литература

4. Unified Modeling Language User Guide, The Grady Booch James Rumbaugh Ivar Jacobson [1]

5. Гради Буч, Джеймс Рамбо, Ивар Якобсон Язык UML. Руководство пользователя [2] (перевод)

6. Wikipedia [3]

5.3. Виды диаграмм UML

Диаграммы поведений

Поведенческие диаграммы показывают, что должно происходить в системе. Они описывают, как объекты взаимодействуют друг с другом для создания функционирующей системы.

Они делятся на 7 типов:

1. состояний
2. вариантов использования
3. обзора взаимодействия
4. коммуникации
5. последовательности
6. синхронизации
7. действий [2]

Диаграммы действий

Диаграммы действий представляют рабочие процессы в графическом виде. Они могут использоваться для описания бизнес-процесса или рабочего процесса любого компонента системы. Иногда диаграммы активности используются в качестве альтернативы диаграммам конечных автоматов.

Диаграммы состояний

Диаграммы конечных автоматов аналогичны диаграммам действий, хотя обозначения и использование немного меняются. Их также иногда называют диаграммами состояний или диаграммами диаграмм состояний. Они очень полезны для описания поведения объектов, которые действуют поразному в зависимости от состояния, в котором они находятся в данный момент

Диаграммы вариантов использования

Как наиболее известный тип диаграмм поведенческих типов UML, диаграммы вариантов использования дают графический обзор участников, участвующих в системе, различных функций, необходимых этим участникам, и того, как эти различные функции взаимодействуют. Это отличная отправная

точка для обсуждения любого проекта, потому что вы можете легко определить основных участников и основные процессы системы.

Диаграммы обзора взаимодействия

Обзорные диаграммы взаимодействия очень похожи на диаграммы деятельности. В то время как диаграммы деятельности показывают последовательность процессов, диаграммы обзора взаимодействия показывают последовательность диаграмм взаимодействия. Они представляют собой набор диаграмм взаимодействия и порядок, в котором они происходят. Как упоминалось ранее, существует семь типов диаграмм взаимодействия, поэтому любая из них может быть узлом на диаграмме обзора взаимодействия.

Диаграммы коммуникации

В UML 1 они назывались диаграммами совместной работы. Диаграммы связи аналогичны диаграммам последовательности, но основное внимание уделяется сообщениям, передаваемым между объектами. Одна и та же информация может быть представлена с помощью диаграммы последовательности и разных объектов.

Диаграммы последовательности

Диаграммы последовательности в UML показывают, как объекты взаимодействуют друг с другом и в каком порядке происходят эти взаимодействия. Важно отметить, что они показывают взаимодействия для конкретного сценария. Процессы представлены вертикально, а взаимодействия показаны стрелками.

Диаграммы синхронизации

Временные диаграммы очень похожи на диаграммы последовательности. Они представляют поведение объектов в заданный период времени. Если это только один объект, схема проста. Но если задействовано более одного объекта, для отображения взаимодействий между объектами в течение этого периода времени используется временная диаграмма.[1]

Вопросы:

1. Назовите несколько типов диаграмм поведений.
2. Что представляют собой диаграммы действий?
3. Что показывают диаграммы поведения?

Рекомендуемая литература:

1. https://creately.com/blog/diagrams/uml-diagram-types-examples/
2. https://ru.wikipedia.org/wiki/UML
3. https://en.wikipedia.org/wiki/Unified_Modeling_Language

Структурные диаграммы

Структурные диаграммы показывают элементы моделируемой системы - её объекты.[4]

Структурные диаграммы делятся на 7 видов:

1. классов

2. компонентов

3. композитной/составной структуры

4. развёртывания
5. объектов
6. пакетов

7. профилей[1]

Диаграмма классов

Диаграммы классов являются основным элементом любого объектно-ориентированного решения. Они показывает классы в системе, атрибуты и операции каждого класса, а также отношения между каждым классом. В большинстве инструментов моделирования класс состоит из трех частей. Имя сверху, атрибуты посередине и операции или методы внизу. Различные отношения между классами показаны разными типами стрелок. [4]

Диаграмма компонентов

— статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п. [1]

Диаграмма композитной/составной структуры

Диаграмма композитной/составной структуры (Composite structure diagram) — статическая структурная диаграмма, демонстрирует внутреннюю структуру классов и, по возможности, взаимодействие элементов (частей) внутренней структуры класса.

Диаграммы композитной структуры могут использоваться совместно с диаграммами классов. [1]

Диаграмма развёртывания

Диаграмма развертывания показывает оборудование вашей системы и программное обеспечение в этом оборудовании. Диаграммы развертывания полезны, когда ваше программное решение развертывается на нескольких компьютерах, каждый из которых имеет уникальную конфигурацию. [4]

Диаграмма объектов

Диаграмма объектов (Object diagram) — демонстрирует полный или частичный снимок моделируемой системы в заданный момент времени. На диаграмме объектов отображаются экземпляры классов (объекты) системы с указанием текущих значений их атрибутов и связей между объектами. [1]

Диаграмма пакетов

Диаграмма пакетов (Package diagram) — структурная диаграмма, основным содержанием которой являются пакеты и отношения между ними. Жёсткого разделения между разными структурными диаграммами не проводится, поэтому данное название предлагается исключительно для удобства и не имеет семантического значения (пакеты и диаграммы пакетов могут присутствовать на других структурных диаграммах). Диаграммы пакетов служат, в первую очередь, для организации элементов в группы по какому-либо признаку с целью упрощения структуры и организации работы с моделью системы. [1] Пакет - это вид группировки элементов модели и диаграмм. [5]

Диаграмма профилей

диаграммы профилей предоставляют возможность расширить функционал UML. Они основаны на дополнительных стереотипах и помеченных

значений (Tagged values), применяемых к элементам UML, коннекторам и их компонентам. Профиль (Profile) это собрание таких расширений, которые вместе описывают конкретную задачу моделирования и облегчают построение моделей в этой области [2]

Рекомендуемая литература

1. https://ru.wikipedia.org/wiki/UML#%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2

2. https://sparxsystems.com/enterprise_architect_user_guide/15.2/model_domains/profile_diagram.html

3. https://qna.habr.com/q/174615

4. https://creately.com/blog/diagrams/uml-diagram-types-examples/

5. https://openu.ru/Books/UML/Package.asp

Учебное издание

Николай Михайлович **Рашевский**
Татьяна Владимировна **Ерещенко**

МОДЕЛИ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ И СИСТЕМ

Учебное пособие

Редактор *Л. Н. Рыжих*

Темплан 2021 г. (учебники и учебные пособия). Поз. № 19.
Подписано в печать 00.00.2021. Формат 60x84 1/16. Бумага газетная.
Гарнитура Times. Печать офсетная. Усл. печ. л. 4,0. Уч.-изд. л. 5,05.
Тираж 100 экз. Заказ

Волгоградский государственный технический университет.
400005, г. Волгоград, просп. В. И. Ленина, 28, корп. 1.

Отпечатано в типографии ИУНЛ ВолгГТУ.
400005, г. Волгоград, просп. В. И. Ленина, 28, корп. 7.