ПРИОРИТЕТНЫЙ НАЦИОНАЛЬНЫЙ ПРОЕКТ «ОБРАЗОВАНИЕ»
РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

К.А. ПУПКОВ

ПРИМЕНЕНИЕ CALS-ТЕХНОЛОГИЙ ДЛЯ ПОВЫШЕНИЯ КАЧЕСТВА ИЗДЕЛИЙ

Учебное пособие

Москва

2008

Инновационная образовательная программа
Российского университета дружбы народов

«Создание комплекса инновационных образовательных программ и формирование инновационной образовательной среды, позволяющих эффективно реализовывать государственные интересы РФ через систему экспорта образовательных услуг»

Экспертное заключение –

кандидат технических наук, доцент *Н.А. Малахов*

Пупков К.А.

Применение CALS-технологий для повышения качества изделий: Учеб. пособие. – М.: РУДН, 2008. – 105 с.

В учебном пособии рассматриваются с единых позиций процессы разработки, проектирования и создания изделий. Требованием времени является такая организация разработки образцов техники, которая обеспечивала сопровождение их от замысла конструктора, научной проработки до реального образца, находящегося в эксплуатации, т.е. на всем жизненном цике изделия. Одной из основных задач в этом процессе является совместимость программного обеспечения, используемого при разработке. Такая совместимость программного обеспечения называется CALS-технологией.

Для студентов, обучающихся по направлению «Автоматизация и управление», а также специалистов.

Учебное пособие выполнено в рамках инновационной образовательной программы Российского университета дружбы народов, направление «Комплекс экспортоориентированных инновационных образовательных программ по приоритетным направлениям науки и технологий», и входит в состав учебно-методического комплекса, включающего описание курса, программу и электронный учебник.

Оглавление

Обозначения и сокращения

САПР – система автоматизированного проектирования
СУБД – система управления базами данных
САД – Computer Aided Design
CAE – Computer Aided Engineering
CALS – Continuonsr Acquisition and Life-Cycle Support
CAM – Computer Aided Manufacturing
CASE – Computer Aided System Engineering
CASE – Computer Aided Software Engineering
PDM – Product Data Management
SCADA – Supervisory Control and Data Acquistion
STEP – Standart for Exchange of Product Data

Введение

Возможность автоматизации процессов разработки появилась в связи с бурным развитием систем вычислительной техники. Огромные объемы памяти, высокое быстродействие, возможность ввода и вывода графической информации, обеспечение диалога разработчика (конструктора) с базой знаний и данных, хранимых в ЭВМ, позволили автоматизировать процесс проектирования и отработки изделий и систем.

В этих условиях особенно важна разработка концепции организации технологического обеспечения процессов моделирования и испытания интеллектуальных систем, основанных на комплексировании принцитпов робастного, нейро-нечеткого и адаптивного управления. Важным является также разработка теоретических основ и метолдологии реализации процессов реализации, процессов моделирования и испытаний интеллектуальных систем.

Разработка новых образцов техники представляет собой сложный процесс, состоящий из ряда этапов, связанных с формированием концептуальной модели объекта, технического задания и исходных данных для проектирования, собственно процессом проектирования, изготовлением опытных образцов и различными видами испытаний (как отдельных элементов, так и всего комплекса программного обеспечения и аппаратуры изделия и системы).

Разработка (или опытно-конструкторская разработка – ОКР) завершается передачей образца техники в серийное производство, а предшествует ей, как правило, большой объем научно-исследовательских работ, завершающихся в случае успеха разработкой проекта технического задания на ОКР.

Опытно-конструкторская разработка обычно включает в себя подготовительную стадию: формирование концептуальной модели, составление технического задания на ОКР и разработку технических предложений. Весь процесс разработки можно представить в виде схемы (рис. 1).

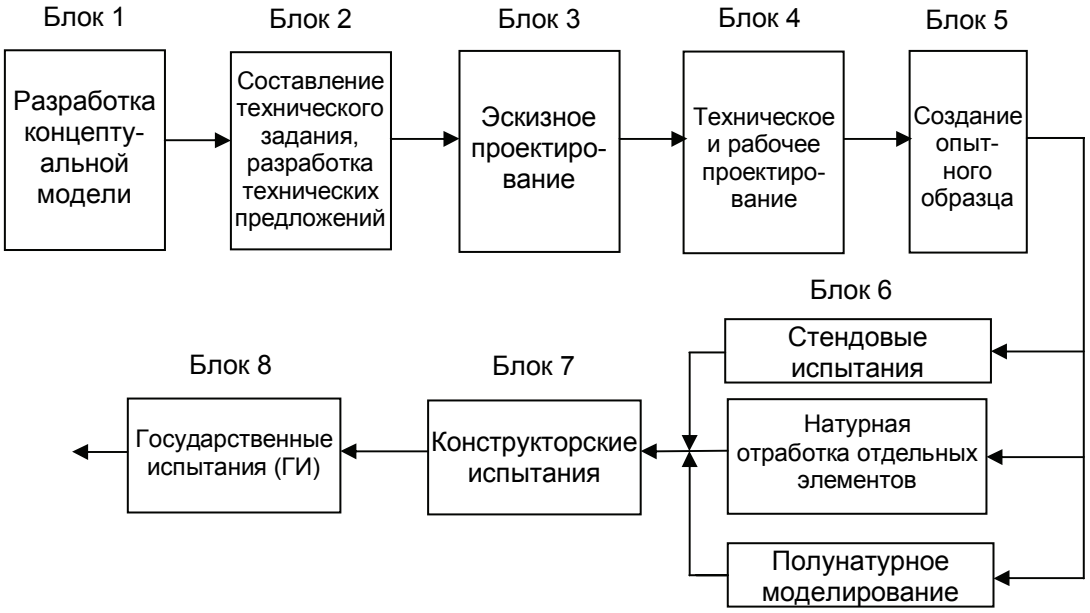

Рис. 1. Структурная схема процесса разработки изделия

Рассмотрим более детально процесс разработки изделий и систем. В блоке 1 процесса предполагается разработка облика системы (например, рассматривается оптимальное распределение технологического процесса по операциям), выбираются подходящие материалы, определяется технико-экономическое обоснование и т.п.

В блоке 2 на основе анализа концептуальной модели и других результатов научно-исследовательских работ составляется техническое задание и формируются технические предложения по вариантам построения изделия или системы. Технические предложения разрабатываются, как правило, при выполнении ОКР для сложных изделий и таких ОКР, для которых определение и обоснование возможности выполнения технических требований, технико-экономической целесообразности, объема, сроков выполнения, сложности работ к началу разработки затруднительно.

В блоке 3 реализуется эскизный проект – совокупность конструкторской документации (КД), содержащей принципиальны и конструктивные решения, дающие общее представление об устройстве и принципе работы систем, и данные, определяющие ее назначение и основные параметры.

В процессе эскизного проектирования уточняют технико-экономические показатели, рассматривают принципы построения и выбирают оптимальные варианты системы и ее частей с учетом зависимости стоимости от эффективности и масштабов производства, разрабатывают структурную и функциональную схемы системы и ее частей, определяют общие конструктивные и технологические решения, рассматривают вопросы сопряжения отдельных частей системы и ее подсистемы между собой, составляют перечень специальных технологических процессов, подлежащих разработке, проводят макетирование наиболее сложных и ответственных функциональных частей в объеме, необходимом для оценки правильности и выполнимости намеченных решений в соответствии с заданными техническими требованиями, и формируют требование на разработку необходимого технологического, контрольно-измерительного и испытательного оборудования. Разрабатывают математическую и физическую модели системы, математическое и программное обеспечения. На этом этапе разработки можно видеть также математическую модель системы, уже более конкретную и детальную по сравнению с концептуальной моделью.

В блоке 4 выполняется уточненный расчет и оценивается надежность разрабатываемой системы и ее функциональных частей освещаются вопросы эксплуатации и устранения отказов, контроля качества работы и т.д.

Технический проект (ТП) – совокупность конструкторской документации, в которой должны содержаться окончательные технические решения, дающие полное представление об устройстве создаваемой системы, и исходные данные для выпуска рабочей документации. На этапе ТП осуществляется теоретическая и экспериментальная проверка схемных, конструктивных и технологических решений, разрабатывается техническая документация, обеспечивающая последующую разработку конструкторской и технологической документации на изделие, его составные части и отдельные виды необходимого технологического оборудования, технологической оснастки, контрольно-измерительного и испытательного оборудования.

В процессе разработки ТП для уточнения принципиальных схемных решений и отработки конструкции могут изготовляться действующие макеты или экспериментальные образцы, которые могут проходить лабораторные

исследования (физическое моделирование), а при необходимости – физико-механические или натурные испытания.

Итогом рассмотрения и принятия ТП является выпуск рабочей конструкторской и технологической документации как опытного образца, так и необходимой стендовой и испытательной аппаратуры.

В соответствии с рабочей КД изготовляется опытный образец (если возможно, то несколько экземпляров, чтобы распараллелить его отработку). В схеме процессов разработки это соответствует блоку 5.

Опытный образец проходит всестороннюю отработку (блок 6): стендовые испытания, научную отработку элементов и систем и полунатурное моделирование. Совокупность этих испытаний и испытаний, проведенных в условиях, прибл□жающихся к реальным условиям эксплуатации и подтвердивших на данном уровне адекватности соответствие системы управления ТЗ, позволяет перейти к натурным испытаниям – блок 7.

Эти испытания, которые проводятся в основном в натурных условиях, должны обеспечить окончательную отработку опытного образца системы. Чтобы получить оценку системы, необходимо иметь информацию о работе ее элементов (функциональных устройств), значениях параметров, обработка и анализ которой позволяет сделать заключение о соответствии системы управления ТЗ. Государственные испытания (ГИ) – блок 9 – являются продолжением конструкторских испытаний, по их результатам производится окончательная оценка системы и принимается решение о серийном производстве.

Следует отметить, что на всех этапах разработки конструктор оперирует моделью системы, добиваясь на каждом этапе на соответствующем уровне адекватности модели техническому заданию.

Таким образом, моделирование является научной основой разработки, а использование при этом, включая решение задач оптимизации и выпуска конструкторской документации, вычислительных машин составляет суть автоматизированной разработки систем и автоматизированного проектирования (САПР).

Наряду с последовательным процессом разработки систем управления следует указать на многоэтапный иерархический процесс проектирования в каждом блоке:

на уровне системного проектирования изучается взаимодействие проектируемой системы управления с окружающей средой, определяются виды воздействия на объект управления, элементы системы управления, типы помех, дается их математическое описание в виде, удобном для проведения расчетов или математического и полунатурного моделирования (уровень системного проектирования широко используется в блоке 1);

на уровне структурного проектирования определяются типы функциональных устройств (гироскопические приборы, радиолокационные и оптико-электронные средства и т.п.), образующих систему и структуру связей между ними. Этим обеспечиваются заданные информационные и точностные характеристики системы; такое проектирование широко используется в блоке 3, где формируется структура системы управления, математические модели функциональных устройств и системы в целом для оценки динамической точности;

на уровне функционального проектирования обеспечивается выполнение элементами системы их функционального назначения на основе знания приближенный или идеализированной формы входных и внутренних сигналов;

на уровне схемотехнического проектирования прорабатывается форма сигналов для отдельных элементов системы, а также рассчитываются уточненные значения их внутренних и выходных параметров, формируется техническое задание на кабельную сеть;

на уровне проектирования компонентов определяются параметры конструкции, материалов и технологии, обеспечивающие заданные характеристики отдельных компонентов (микросхем преобразующих устройств и т.п.);

на уровне отработки систем осуществляются всесторонние исследования и испытания опытного образца, на основе результатов которых производится корректировка всех видов конструкторской документации, так как процесс разработки систем управления является процессом с обратной связью.

Заметим, что под интеллектуальной системой понимается объединенная информационным процессом совокупность технических средств и программного обеспечения, работающая во взаимосвязи с человеком (коллективом людей) или автономно, способная на основе сведений и знаний при наличии мотивации синтезировать цель, принимать решение к действию и находить рациональные способы достижения цели.

Система автоматизированного проектирования (САПР) – интеллектуальная система, целью которой является проект.

Важнейшей особенностью современных разработок, проектов и создаваемых образцов техники является их высокая сложность, определяемая различными физическими принципами функционирования, разнообразием материалов и других компонентов, различными технологиями и инструментальными средствами.

Требованием времени является такая организация разработки образцов техники, которая обеспечивала бы сопровождение изделия от замысла конструктора (идей), ее научной разработки до реального образца, находящегося в эксплуатации, т.е. на всем жизненном цикле изделия. Одной из основных задач в этом процессе является совместимость программного обеспечения, используемого при разработке. Такая совместимость программного обеспечения называется CALS-технологией. Именно этому посвящено данное учебное пособие.

2. Предпосылки и причины появления CALS-технологий

Эффективность интеграции данных о промышленных изделиях

CALS-технологии призваны служить средством, интегрирующим промышленные автоматизированные системы в единую многофункциональную систему. Целью интеграции автоматизированных систем проектирования и управления является повышение эффективности создания и использования сложной техники.

В чем выражается повышение эффективности?

Во-первых, повышается качество изделий за счет более полного учета имеющейся информации при проектировании и принятии управленческих решений. Так, обоснованность решений, принимаемых в автоматизированной системе управления предприятием (АСУП), будет выше, если ЛПР (лицо, принимающее решение) и соответствующие программы АСУП имеют оперативный доступ не только к базе данных АСУП, но и к базам данных других автоматизированных систем (САПР, АСТПП и АСУТП) и, следовательно, могут оптимизировать планы работ, содержание заявок, распределение исполнителей, выделение финансов и т.п. При этом под оперативным доступом следует понимать не просто возможность считывания данных из БД, но и легкость их правильной интерпретации, т.е. согласованность по синтаксису и семантике с протоколами, принятыми в АСУП. То же относится и к другим системам, например, технологические подсистемы должны с необходимостью воспринимать и правильно интерпретировать данные, поступающие от подсистем автоматизированного конструирования. Этого не так легко добиться, если основное предприятие и организации-смежники работают с разными автоматизированными системами.

Во-вторых, сокращаются материальные и временные затраты на проектирование и изготовление продукции. Применение CALS-технологий позволяет существенно сократить объемы проектных работ, так как описания ранее выполненных удачных разработок компонентов и устройств, многих составных частей оборудования, машин и систем, проектировавшихся ранее, хранятся в базах данных сетевых серверов, доступных любому пользователю CALS-технологии. Доступность опять же обеспечивается согласованностью форматов, способов, руководств в разных частях общей интегрированной системы. Кроме того, появляются более широкие возможности для специализации предприятий, вплоть до создания виртуальных предприятий, что также способствует снижению затрат.

В-третьих, существенно снижаются затраты на эксплуатацию, благодаря реализации функций интегрированной логистической поддержки. Существенно облегчается решение проблем ремонтопригодности, интеграции продукции в различного рода системы и среды, адаптации к меняющимся условиям эксплуатации и т.п.

Эти преимущества интеграции данных достигаются применением современных CALS-технологий.

Промышленные автоматизированные системы могут работать автономно, и в настоящее время так обычно и происходит. Однако эффективность автоматизации будет заметно выше, если данные, генерируемые в одной из систем, будут доступны в других системах, поскольку принимаемые в них решения станут более обоснованными.

Чтобы достичь должного уровня взаимодействия промышленных автоматизированных систем требуется создание единого информационного пространства в рамках как отдельных предприятий, так и, что более важно, в рамках объединения предприятий. Единое информационное пространство обеспечивается благодаря унификации как формы, так и содержания информации о конкретных изделиях на различных этапах их жизненного цикла.

Унификация формы достигается использованием стандартных форматов и языков представления информации в межпрограммных обменах и при документировании.

Унификация содержания, понимаемая как однозначная правильная интерпретация данных о конкретном изделии на всех этапах его жизненного цикла, обеспечивается разработкой онтологий (метаописаний) приложений, закрепляемых в прикладных протоколах CALS.

Унификация перечней и наименований сущностей, атрибутов и отношений в определенных предметных областях является основой для единого электронного описания изделия в CALS-пространстве.

Обзор CALS-стандартов

Центральное место в системе CALS-стандартов занимают стандарты, разработанные под эгидой Международной организации стандартизации ISO и получившие название STEP (Standard for Exchange of Product data) и номер 10303. Стандарты ISO 10303 определяют средства описания (моделирования) промышленных изделий на всех стадиях жизненного цикла.

Единообразная форма описаний данных о промышленной продукции обеспечивается введением в STEP языка Express, инвариантного к приложениям. В стандартах STEP использован ряд идей, ранее воплощенных в методиках информационного IDEF1X и функционального IDEF0 проектирования. Но роль стандартов STEP не ограничивается введением только грамматики единого языка обмена данными. В рамках STEP предпринята попытка создания единых информационных моделей (онтологий) целого ряда приложений. Эти модели получили название прикладных протоколов.

Стандарт ISO 10303 состоит из ряда документов (томов), в которых описываются основные принципы STEP, правила языка Express, даны методы его реализации, модели, ресурсы, как общие для приложений, так и некоторые специальные (например, геометрические и топологические модели, описание материалов, процедуры черчения, конечно-элементного анализа и т.п.), прикладные протоколы, отражающие специфику моделей в конкретных предметных областях, методы тестирования моделей и объектов.

Удовлетворению требований создания открытых систем в STEP уделяется основное внимание - специальный раздел посвящен правилам написания файлов обмена данными между разными системами, созданными в рамках STEP-технологии.

Развитие CALS-технологий находит выражение в разработке серий и других стандартов. Это ISO 13584 Parts Library (сокращенно P_Lib), ISO 14959 Parametrics, ISO 15531 Manufacturing management data (Mandate), ISO 18876 Integration of industrial data for exchange, access, and sharing (IIDEAS), ISO 8879 Standard Generalized Markup Language (SGML).

Стандарты Parts Library (P-LIB) содержат обзор и основные принципы представления данных о стандартных компонентах промышленных изделий. В

этих стандартах представлены в виде библиотек данные о семействах таких типовых широко используемых компонентов изделий, как болты, подшипники, электронные компоненты и т.п., с целью использования этих данных в различных системах автоматизированного проектирования. В P-LIB содержатся также правила использования, интерфейса и модификации библиотечных описаний. Цель стандарта - обеспечить инвариантный для приложений механизм оперирования частями библиотеки.

Благодаря ISO 13584 различные прикладные САПР могут разделять данные из обобщенных баз, беспрепятственно обмениваться данными о типовых компонентах. Описание библиотечных моделей дается на языке Express. Для описания структуры частей, вводимых определений и других текстовых фрагментов используется язык SGML. Поведенческие модели электронных компонентов могут быть выражены с помощью языка VHDL.

Стандарты P-LIB состоят из нескольких частей.

Часть 1 содержит обзор и основные принципы серии стандартов.

Часть 10 посвящена концептуальной модели, а часть 24 - логической модели построения библиотек. Библиотеки могут компоноваться из данных от разных поставщиков. В части 26 определяются поставщики библиотек, в части 31 описан программный интерфейс. Описание методологии структуризации семейств содержится в части 42. Протоколам обмена посвящены части, начинающиеся с номера 101. Часть под номером 101 содержит протокол обмена геометрической параметризованной информацией; часть под номером 102 - протокол обмена согласованными с STEP данными.

Стандарты Parametrics введены в 1996 г. в связи с тем, что стандарты STEP в недостаточной мере учитывали особенности современных САПР в части представления параметризованных моделей изделий и обмена параметризованными данными.

Рабочая группа ISO по Parametrics решает как краткосрочные, так и перспективные задачи. Первые из них касаются удовлетворения потребностей геометрического проектирования и машинной графики в сегодняшних САПР, в которых широко используются параметризованные модели. Вторые касаются попыток распространения идей параметризации на более ранние этапы проектирования и на более широкий круг моделей и процедур проектирования, имеющих не только геометрический характер.

Стандарты Mandate посвящены представлению данных, относящихся к функционированию предприятий, управлению территориально распределенными производственными системами, обмену данными о производстве с внешней для предприятия средой.

Часть стандарта, обозначаемая ISO 15531-21, содержит обзор и основные принципы представления данных о промышленной продукции. Содержание этой части характеризуется следующими ключевыми словами: системы промышленной автоматизации и интеграция, промышленные данные, обмен данными об управлении производством, обмен данными с внешней средой.

Том ISO 15531-31 посвящен обзору и основным принципам использования данных о производственных ресурсах. Излагаются модель, форма и атрибуты представления данных о производственных ресурсах, об управлении их применением.

Том ISO 15531-41 содержит обзор и основные принципы управления потоками производственных данных.

В настоящее время число различных CALS-стандартов довольно велико. В связи с этим предпринимаются попытки их большей гармонизации. Так, в рабочей группе WG10 подкомитета SC4 ISO разрабатывается стандарт ISO 18876 "*Integration of industrial data for exchange, access, and sharing"* (IIDEAS). Его назначение - обеспечение взаимодействия приложений и организаций, использующих разные стандарты, интеграция данных и моделей, получаемых из разных источников, разрабатываемых в разных САПР. Предусматриваются возможности согласования моделей, выраженных с помощью разных языков моделирования и форматов, например, таких, как SGML, XML, EXPRESS. Средства интеграции - специальные интеграционные модели и методы создания, распространения, обновления моделей, их связи с прикладными протоколами.

Оформление технической документации на создаваемые изделия в CALS-технологиях должно выполняться на основе языков разметки SGML (Standard Generalized Markup Language) или XML (eXtensible Markup Language). Язык SGML входит в семейство стандартов ISO 8879 и предназначен для унификации представления текстовой информации в автоматизированных системах.

Язык XML стал основой технологий семантической сети (Semantic Web), в которой появляется возможность интеграции информации из разных источников с распознаванием семантики данных. Вместе с унифицированным способом указания информационных ресурсов URI (Uniform Resource Identifiers) язык XML составляет нижний уровень в иерархии средств семантического вэба. Над этим уровнем расположены уровень модели RDF (Resource Description Framework - среда описания ресурсов), которая служит для определения и использования метаданных, описывающих информационные ресурсы. RDF определяет набор ресурсов и свойств с обозначением их смысла, наборы могут быть использованы для описания новых RDF-словарей. Словарь RDF определяется RDF-схемой (RDFS). RDF-схема представляет собой систему типов для Semantic Web. Над уровнем RDF находится уровень языков онтологий, к которым, в частности, относятся языки OWL, DAML, OIL. В целях стандартизации технологий Semantic Web международный консорциум W3C разработал ряд спецификаций, посвященных XML, RDF (RDFS), OWL и др. Работы по развитию технологий CALS и Semantic Web составляют пока разные направления, однако есть немало причин для пересечения этих направлений.

Стандарт MIL-STD-1840C посвящен представлению и обмену данными в CALS-технологиях. Основные положения этого стандарта признаны в России и представлены в документе Р50.1.027-2001. Стандарт определяет международные, национальные, военные стандарты и спецификации для электронного обмена информацией между организациями или системами. В нем к стандартам и спецификациям технологий CALS отнесен ряд стандартов таких, как вышеназванные стандарты STEP, SGML, а также стандарты шифрования данных и электронной подписи, кодирования аудио и видео данных, спецификации MIME электронной почты и т.п.

В соответствии с MIL-STD-1840C документы могут быть SGML-документами, обменнми файлами на языке Express, для представления иллюстраций и текста допускается использование ряда других форматов. Так, для предачи и представления в технических руководствах иллюстративного материала (схем, рисунков) в соответствии с американским стандартом MIL-PRF-28003 можно использовать формат BMP, но более экономичен формат JPEG. Для 2D чертежей (но не в САПР) рекомендуется использовать формат CGM (Computer Graphics Metafile), ранее введенный в ISO/IEC 8632. Растеризация выполняется в

соответствии с рекомендацией MIL-PRF-28002. Стандартный растровый формат - TIFF. Отметим, что документы MIL-PRF-28000 и MIL-PRF-28001 посвящены соответственно форматам IGES и SGML. Формат IGES (Initial Graphics Exchange Specification), утвержденный в качестве стандарта в начале 80-х годов, был предшественником STEP, но он был ориентированным в основном на описание геометрических свойств изделий.

В структуре документа выделяют реквизитную и содержательную части. В реквизитной части записываются метаданные в виде списка идентификаторов атрибутов и их значений, а также сведения об электронных подписях документа. Содержательная часть состоит из одного или более блоков данных, каждый блок имеет собственно передаваемые данные и их описание.

Электронная цифровая подпись (ЭЦП) представляет собой хэш-функцию передаваемого документа, закодированную составителем документа закрытым ключом по асимметричной схеме. Прочитать ЭЦП можно с помощью открытого ключа, но подделать подпись, не зная закрытого ключа, практически нельзя.

К технологиям CALS относят также технологии интегрированной логистической поддержки изделий (ИЛП). Под ИЛП понимают комплекс мер и процедур, направленных на снижение общих затрат на всех этапах жизненного цикла изделий (ЖЦИ), прежде всего на этапе эксплуатации. В понятие ИЛП входят:

- Определение инфраструктуры системы обслуживания изделий в период эксплуатации, в том числе планирование процедур обслуживания и проектирование ремонтопригодной техники, разработка средств обслуживания сложной техники параллельно с разработкой самого изделия;
- Определение требований и обучение обслуживающего персонала;
- Поддержка связей между производителем и потребителем путем доступа потребителя к интегрированной базе данных изделия с целью упрощения диагностики состояния и ремонта изделий, а также получения изготовителем данных о неисправностях и отказах с целью принятия мер по повышению надежности изделий;
- Классификация и кодификация изделий и материалов, необходимые для упрощения поиска нужных данных в справочниках и БД, исключения дублирования проектов, ускорения составления заявок на поставки комплектующих и т.п.;
- Традиционные логистические процедуры, такие как упаковка, складирование, транспортировка изделий.

В настоящее время в качестве основного стандарта ИЛП признан стандарт Великобритании 00-60.

Реализация ИЛП осуществляется в рамках систем ИЛП. Следует отметить, что ИЛП тесно связана с обеспечением управления качеством продукции в соответствии со стандартами серии ISO 9000.

Для презентаций проектов и для обучения персонала, занимающегося обслуживанием и эксплуатацией изделий, создаются технические руководства IETM - Interactive Electronic Technical Manual (или IETP - Interactive Electronic Technical Publication) и учебные пособия (ICW - Interactive Courseware). В них содержатся описания изделий, технологии эксплуатации, поясняются приемы обслуживания, методы диагностики и ремонта. В частности, в технических руководствах должны быть сведения о планировании регламентных работ, типовых отказах, способах обнаружения неисправностей и замены неисправных

компонентов, испытательном оборудовании, способах заказа материалов и запасных частей и т.п.

Эксплуатационные документы должны создаваться в соответствии с концепциями, развиваемыми в CALS, обеспечивая повышенные удобства и эффективность освоения и эксплуатации сложной техники. Концепция создания и сопровождения электронной эксплуатационной документации получила название технологии IETM или ИЭТР (интерактивных электронных технических руководств).

В технологиях CALS к эксплуатационной документации IETM предъявляются повышенные требования. Это прежде всего представление документов в электронном виде, открытость пособий и руководств, т.е. их приспособленность к внесению изменений и конвертированию форматов, должная степень интерактивности и управления данными, адаптация учебного материала к конкретным запросам пользователей, малые затраты на создание документов для новых версий изделий.

В ИЭТР используют классификацию документов. По одной из существующих систем классификации выделяют следующие классы ИЭТР:

Класс 1 - Бумажно-ориентированные электронные документы. Это отсканированные страницы и электронные копии бумажных руководств. Преимущества: большие объемы бумажной документации заменяет компактный электронный носитель. Недостатки: отсутствуют какие-либо новые функции по сравнению с возможностями бумажных руководств.

Класс 2 - Неструктурированные документы, текстовые электронные документы. Преимущества: возможность использования аудио- и видеофрагментов, графических изображений и возможность осуществлять поиск по тексту документа. Недостатки: ограниченные возможности обработки информации.

Класс 3 - Структурированные документы. Начиная с класса 3, руководства представляют собой документы, имеющие три компонента: структура, оформление и содержание. Кроме того, начиная с класса 3, ИЭТР имеют стандартизированный интерфейс пользователя. Преимущества: существует возможность стандартизировать структуру, оформление и пользовательский интерфейс руководств (например, в соответствии с отраслевыми стандартами на эксплуатационную документацию), стандартизированный интерфейс пользователя позволяет облегчить работу с ИЭТР. Недостатки: при создании руководств к сложным промышленным изделиям появляются проблемы управления большим объемом информации.

Класс 4 - Интерактивные базы данных. Преимущества: можно создавать технические руководства большого объема. Недостатки: отсутствие системы диагностики изделия.

Класс 5 - Интегрированные базы данных. Дают возможность прямого взаимодействия с электронными модулями диагностики изделий, что существенно облегчает обслуживание и ремонт изделия. Преимущества: возможность проведения диагностики изделия. Недостатки: очень высокая стоимость создания. Вариант использования конкретного класса ИЭТР, в общем случае, зависит от сложности изделия, от финансовых и технических возможностей пользователя.

В соответствии с другой из существующих классификаций IETM выделяют 6 классов. Класс 0 относится к обычным документам, переведенным в электронный вид (например, с помощью редактора Word) и предназначенным для архивации.

Класс 1 относится к документам, части которого индексированы и доступны по ссылкам из оглавления. Документы класса 2 - файлы в коде ASCII, внутри которых применена разметка с помощью тегов, что позволяет осуществлять навигацию внутри пособия. Документы класса 3 отличаются тем, что в них применена разметка с помощью языка SGML.

Документы классов 0-3 являются линейными в том смысле, что в них, как и в обычных бумажных пособиях, материал излагается последовательно страница за страницей. В отличие от них документы класса 4 имеют не линейную, а иерархическую структуру, и предназначены для интерактивных презентаций. Развитие класса 4 в направлении увеличения степени интеллектуализации приводит к классу 5, в котором имеются средства формирования версий пособий, адаптированных к запросам и уровню подготовленности пользователя.

В технологиях IETM используется ряд стандартов. Кроме стандарта ISO 8879 (SGML), здесь находят применение стандарт ISO 10744 (HyTime - Hypermedia / Time-based Document Structuring Language), спецификации MIL-87268...87270 и др. Так, документ MIL-M-87268 (Interactive Electronic Technical Manual Content) определяет общие требования к содержанию, стилю, формату и средствам диалогового общения пользователя с интерактивными электронными техническими руководствами. В спецификации MIL-D-87269 содержатся требования к базам данных для интерактивных электронных технических руководств и справочников, описаны методы представления структуры и состава промышленного изделия и его компонент на языке SGML, даны шаблоны документов на составные части технической документации, перечислены типовые элементы документов.

Спецификация AECMA 1000D - технология представления технической документации, признанная в авиационной промышленности (AECMA - European Association of Aerospace Constructors). В основе AECMA 1000D, как и в старших классах IETM, лежит декомпозиция представляемого материала на модули. Модули включают идентификационную и содержательную секции, записанные на языках SGML или HyTime с иллюстрациями в форматах CGM или JPEG, и хранятся в специальной БД. Предусмотрена автоматическая простановка гиперссылок (для этого имеются соответствующие программные средства).

Для унификации структуры документов и правил деловой переписки прежде всего в торговых операциях Организация Объединенных Наций приняла в 1986 г. спецификации EDIFACT (Electronic Data Interchange For Administration, Commerce and Transport). Это международный стандарт для представления и обмена электронными данными, которые могут группироваться в сегменты, смысл которых частично описан в стандарте, но может быть обусловлен договоренностью между партнерами.

Особенности проектирования радиоэлектронной аппаратуры находят отражение и в форматах обмена данными. Как отмечено выше, основные методики функционального и логического проектирования электронных устройств основаны на использовании языка VHDL (*Very high-speed integrated circuits* Hardware *Design Language*), получившего статус международного стандарта IEEE 1076 в 1987 г. При конструкторском проектировании для описания топологии СБИС и печатных плат широко применяются форматы EDIF (Electronic Design Interchange Format) и CIF (Caltech Intermediate Format).

Развитие методологии моделирования на базе языка VHDL привело в 1999 г. к принятию стандарта IEEE 1076.1, посвященного смешанному (mixed mode) моделированию. Отметим, что смешанным принято называть аналого-цифровое моделирование, т.е. исследование моделей, в которых используются как

непрерывные, так и дискретные величины. Объединение стнадартов IEEE 1076 и 1076.1 в одном документе VHDL-AMS (VHDL Analog and Mixed Signal) позволило унифицировать описание моделей не только систем электрической природы, но также систем механических, гидравлических, тепловых, а также систем с физически разнородными компонентами.

В CALS-технологиях представлены не только вопросы описания данных и организации информационных обменов, но и вопросы моделирования приложений. Для выполнения начальных шагов моделирования сложных слабоструктурированных приложений рекомендуется использовать методики объектного моделирования на базе языка UML (Unified Modeling Language), функционального моделирования систем IDEF0, информационного моделирования IDEF1X. В частности, методики IDEF0 и IDEF1X представлены в федеральных рекомендациях США соответственно FIPS 183 и FIPS 184.

Стандарты управления качеством промышленной продукции

Международные стандарты серии ISO 9000 разработаны для управления качеством продукции, их дополняют стандарты серии ISO 14000, отражающие экологические требования к производству и промышленной продукции. Хотя эти стандарты непосредственно не связаны с CALS-стандартами, их цели - совершенствование промышленного производства, повышение его эффективности - совпадают. Среди CALS-стандартов, тесно связанных с политикой обеспечения качества, заметную роль играет стандарт ISO 15288 «Системная инженерия: модели и процессы жизненного цикла систем». В стандарте ISO 15288 описаны процедуры, которые рекомендуется выполнять при бизнес-планировании на различных этапах ЖЦИ сложных аппаратно-программных систем.

Очевидно, что управление качеством тесно связано с его контролем. Контроль качества традиционно основан на измерении показателей качества продукции на специальных технологических операциях контроля и выбраковкой негодных изделий. Однако есть и другой подход к управлению качеством, основанный на контроле качественных показателей не самих изделий, а проектных процедур и технологических процессов, используемых при создании этих изделий. Такой подход более эффективен. Он требует меньше затрат, поскольку позволяет обойтись без 100% контроля продукции и, благодаря предупреждению появления брака, снижает производственные издержки. Именно этот подход положен в основу стандартов Международной организации стандартизации ISO 9000, принятых ISO в 1987 г. и проходящих корректировку приблизительно каждые пять лет.

Таким образом, методической основой для управления качеством являются международные стандарты серии ISO 9000. Они определяют и регламентируют инвариантные вопросы создания, развития, применения и сертификации систем качества в промышленности. В них устанавливается форма требований к системе качества в целях демонстрации поставщиком своих возможностей и оценки этих возможностей внешними сторонами.

Основными причинами появления стандартов ISO 9000 были потребности в общем для всех участников международного рынка базисе для контроля и управления качеством товаров. Американское общество контроля качества определило цели ISO 9000 как помощь в развитии международного обмена

товарами и услугами и в кооперации в сфере интеллектуальной, научной, технологической и деловой активности.

В стандартах ISO 9000 используется определение качества из стандарта ISO 8402, в котором под *качеством* продукции подразумевается своевременное удовлетворение требований заказчика при приемлемой цене. Вводится понятие системы качества (QS - Quality System), как документальной системы с руководствами и описаниями процедур достижения качества. Другими словами, система качества есть совокупность организационной структуры, ответственности, процедур, процессов и ресурсов, обеспечивающая осуществление общего руководства качеством.

Система качества обычно представляет собой совокупность трех слоев документов. Слои содержат: 1) описание политики управления для каждого системного элемента (организация, ответственные, контроль); 2) описание процедур управления качеством (что, где, кем и когда должно быть сделано); 3) тесты, планы, инструкции и т.п.

Сертификация предприятий по стандартам ISO 9001-9003 выполняется некоторой уполномоченной внешней организацией. Наличие сертификата качества - одно из важных условий для успеха коммерческой деятельности предприятий.

Стандарты серии ISO 9000 управления качеством промышленной продукции делятся на первичные, вторичные и поддерживающие.

В свою очередь, *первичные* стандарты делятся на внешние и внутренние. Внешние стандарты инвариантны к приложениям, они описывают требования, соблюдение которых гарантирует качество при выполнении контрактов с внешними заказчиками. Внутренние стандарты предназначены для внутреннего использования, они описывают мероприятия по управлению качеством внутри компании.

ISO предлагает следующие внешние стандарты:

ISO 9001 - модель качества, достигаемого при проектировании, производстве, обслуживании;

ISO 9002 - сокращенная по сравнению с ISO 9001 модель (без процессов проектирования);

ISO 9003 - модель качества при финальном тестировании продукции.

Вторичные стандарты включают в себя:

ISO 9000 - основные понятия, руководство по применению ISO 9001;

ISO 9004 - элементы систем управления качеством.

В стандарте ISO 9004 содержатся 20 основных требований к качеству, называемых *системными элементами*. Системные элементы разделены на группы, относящиеся к производству, транспортировке и постпроизводственным операциям, документации продукции, маркетингу. Например, при производстве контролируются планирование, процедуры, программы и инструкции для управления и улучшения производственных процессов. При маркетинге контролируются такие системные элементы, как функциональное описание продукции, организация обратной связи с заказчиками (отслеживание и анализ рекламаций).

Поддерживающие стандарты предназначены для развития и установки систем качества:

ISO 10011 - аудит, критерии для аудита систем качества;

ISO 10012 - метрологическое подтверждение качества;

ISO 10013 - пособие для развития руководств по управлению качеством.

Часть этих стандартов утверждена в качестве государственных стандартов РФ. В частности, это:

ГОСТ Р ИСО 9001-96 "Системы качества. Модель обеспечения качества при проектировании, разработке, производстве, монтаже и обслуживании";

ГОСТ Р ИСО 9002-96 "Системы качества. Модель обеспечения качества при производстве, монтаже и обслуживании";

ГОСТ Р ИСО 9003-96 "Системы качества. Модель обеспечения качества при окончательном контроле и испытаниях".

В настоящее время разработана новая версия стандартов серии ISO 9000 под названием ISO 9000:2000 Quality management systems (Системы управления качеством), в которую включены документы:

ISO 9000:2000 Fundamentals and vocabulary (Основы и терминология);

ISO 9001:2000 Requirements (Требования);

ISO 9004:2000 Guidelines for performance improvement (Руководство по развитию).

Главные отличия новой версии от предыдущей обусловлены стремлением упростить практическое использование стандартов, направлены на их лучшую гармонизацию и заключаются в следующем.

В стандарте ISO 9001 минимизируется объем требований к системе качества. Стандарты ISO 9002-9003 из новой версии исключаются. Расширяется круг контролируемых ресурсов, в их число включены такие элементы, как информация, коммуникации, инфраструктура. 20 элементов качества из стандарта ISO 9004 сворачиваются в 4 группы: распределение ответственности (management responsibility); управление ресурсами (resource management); реализация продукции и услуг (product and/or service realization); измерения и анализ (measurement, analysis, and improvement).

Стандарты ISO 14000 посвящены проблеме выполнения промышленными предприятиями экологических требований. По своей целевой направленности они близки стандартам управления качеством промышленной продукции ISO 9000, которые служат базой для ISO 14000. Стандарты ISO 14000 являются также системой управления влиянием на окружающую среду, они, как и ISO 9000, реализуются в процессе сертификации предприятий, задают процедуры управления и контроль документации, аудит, подразумевают соответствующее обучение и сбор статистики. Кроме требований заказчиков и покупателей, здесь воплощаются внутренние требования организации.

3. STEP-технологии

Структура стандартов STEP

При разработке стандартов STEP были поставлены цели обеспечения единообразного описания и интерпретации данных в автоматизированных системах на различных этапах жизненного цикла изделий. К разработке стандартов STEP под эгидой ISO был привлечен ряд ведущих компаний и специалистов фирм в разных отраслях промышленности.

Основу STEP составляет язык Express. Это язык унифицированного представления данных и обмена данными в компьютерных средах. Язык

инвариантен к приложениям. Хотя он разрабатывался с ориентацией прежде всего на описание жизненных циклов промышленной продукции, области его применения значительно шире.

В STEP используются следующие важные понятия:

AAM - *Application Activity Model*; это функциональная модель IDEF0 для определенного приложения;

ARM - *Application Requirements Model*; это модель, представляющая данные с точки зрения пользователя. В частности, в этой модели данные могут быть выражены как средствами, типичными для приложения, так и с использованием синтаксиса языка Express.

AIM - *Application Interpreted Model*; это ARM модель, переведенная в STEP представление с использованием ряда унифицированных в STEP понятий, закрепленных в интегрированных ресурсах.

AP - *Application Protocol*; это STEP стандарт, отражающий специфику конкретного приложения;

SDAI - *Standard Data Access Interface*; программный интерфейс к базе данных, разделяемой рядом прикладных систем (в том числе CAD/CAM системами) и представленной на языке Express. SDAI представляет собой унифицированный набор процедур доступа к базе данных, используется в STEP средах для организации обменов между приложениями через общую базу данных.

STEP - это совокупность стандартов и состоит из ряда томов. Тома имеют свои номера *N* и обозначаются как "часть *N*" или ISO 10303-*N*. К настоящему времени разработано более сотни томов, часть из них имеет статус проектов, часть уже утверждена в качестве стандартов ISO.

Том 1 (ISO 10303-1) - вводный стандарт, выполняющий роль аннотации всей совокупности томов. В этом стандарте вводится ряд терминов, используемых в других стандартах, например, таких как продукт (product), приложение (application), проектные данные (product data), модель (model), модели AAM, AIM, ARM, прикладной протокол (AP), интегрированный ресурс (integrated resource), элемент функциональности (unit of functionality - UoF).

Тома 11- 14 - методы описания (Description methods),

Тома 21- 29 - методы реализации (Implementation methods),

Тома 31-35 – основы тестирования моделей (Conformance testing methodology and framework),

Тома 41- 50 - интегрированные основные ресурсы (Integrated generic resources),

Тома 101 –108 - интегрированные прикладные ресурсы (Integrated application resources),

Тома 201- 236 - прикладные протоколы (Application protocols),

Тома 301- 332 - абстрактные тестовые наборы (Abstract test suites),

Тома 501 – 520 - прикладные компоненты (Application interpreted constructs).

Ряд томов переведен на русский язык и представлен в виде национальных стандартов России. Это, например, ГОСТ Р ИСО 10303-1-99, посвященный обзору и основополагающим принципам STEP, ГОСТ Р ИСО 10303-11-99 - справочное руководство по языку Express, ГОСТ Р ИСО 10303-21-99 - то же по обменному файлу, ГОСТ Р ИСО 10303-41-99 - описание интегрированных родовых ресурсов. Перечисленные документы соответствуют стандартам ISO 10303-1, ISO 10303-11, ISO 10303-21, ISO 10303-41.

Методы описания

Первая группа документов - тома с номерами в диапазоне с 11 до 19, тома предназначены для описания диалектов языка Express.

N=11: *Express language reference manual*. Основное руководство по языку Express. Содержит также описания расширения Express-C базового языка и графического варианта языка Express-G. Базовый язык приспособлен для описания и передачи статических свойств объектов приложений, т.е. параметров структур и ограничений. Поэтому Express-C включает средства описания динамических свойств объектов (добавлено описание событий и транзакций). Для наглядности представления языковых конструкций в Express предусмотрены графические средства изображения моделей, в качестве которых может использоваться специальное дополнение Express-G (графичесский Express). Express-G - язык диаграмм, напоминающий язык описания информационных моделей в методике IDEF1X.

N=12: *Express-I Language Reference Manual*. Express-I - расширение языка, предназначенное для описания отдельных экземпляров данных.

N=14: Express-X: промежуточный язык, используемый для описания соответствий между типами данных в заданной исходной Express-схеме и создаваемыми новыми ее вариантами (views); в качестве views могут использоваться форматы с описанием того же множества сущностей, что и в Express-схеме, например, формат IGES.

Разрабатываются дополнения, относящиеся к следующим диалектам языка:

Express-M: *Mapping definition languag*e; язык, аналогичный Express-X, служит для описания соответствий между сущностями и атрибутами некоторых моделей, представленных в виде схем на языке Express. Например, этими схемами могут быть два разных прикладных протокола, имеющих частично общие данные, или две схемы одного приложения, но созданные разными лицами (при отсутствии соответствующего AP). Одна схема есть схема-источник, другая - целевая схема. Целевых схем может быть несколько при одной схеме-источнике. Предложения Express-M транслируются на язык C, результирующая программа представляет собой совокупность обращений к функциям базы данных SDAI в STEP-среде. Другими словами, транслятор относится к системе SDAI (см. протокол ISO10303-22), а Express-M можно рассматривать, как язык 4GL для обращений к функциям базы данных SDAI.

Express-P: *Process definition language*; язык диаграмм для представления процессов, методов и коммуникационных структур.

Express-V: язык, предназначенный для получения ARM представлений из AIM моделей, другими словами, для описания процедур поиска экземпляров Express-объектов, отвечающих заданным условиям, и доступа к ним, например, при создании новых ARM. Эти создаваемые ARM-представления обычно не требуют столь всестороннего описания приложения, как в AIM, и потому могут быть существенно проще. В Express-V имеются: 1) схема-источник (AIM), обычно это прикладной протокол, например, AP203; 2) схема-цель, задающая сущности, которые должны быть в создаваемой частной модели; 3) схема отображения нужных сущностей из источника в цель. На языке Express-V описываются условия (в виде клозов WHEN) такого отображения. берется подходящая уже существующая AIM, как источник, все совпадающие объекты переводятся в ARM, далее описываются оригинальные объекты. Дополнительной возможностью реализаций Express-V является обратное отображение специфики создаваемой ARM в исходную AIM с целью развития прикладных протоколов.

Для возможности применения языка Express должны быть разработаны методы реализации (Implementation Methods), которые могут быть представлены средствами файлового взаимодействия, построением БД, интерфейсом с языками программирования.

Методы реализации

Вторую группу (тома с номерами 21...29) называют "Методы реализации", она служит для реализации межпрограммного информационного обмена между прикладными системами в STEP-среде. Предусмотрены межпрограммные связи с помощью обменного файла и доступа к БД.

N=21: *Clear Text Encoding of the Exchange Structure (physical transfer file format)*; стандарт устанавливает правила оформления обменного файла. Обменный файл играет в STEP важную роль; если собственно на языке Express определены сущности, то именно в обменном файле задаются экземпляры этих сущностей. Прикладные программы для связи со STEP средой должны читать и генерировать обменные файлы.

N=22: *Standard Data Access Interface Specification*; содержит описание SDAI - системы представления данных и доступа к данным конкретных прикладных систем (чаще всего это CAD/CAM системы). Данные, участвующие в межпрограммных связях, образуют SDAI-модели. В SDAI системе предусматривается компилятор кода, конвертирующего эти модели в SDAI базу данных, а также функции обращения к этой базе данных. Возможно непосредственное построение прикладных систем, работающих с SDAI базой данных.

Тома с номерами *N* = 23…29 устанавливают правила обращения к данным в SDAI базе данных на языках программирования C++, C, Java, на языке передачи данных в системах распределенных вычислений IDL, языке разметки XML.

Прикладные протоколы

Прикладные протоколы создаются для однозначного понимания спецификаций приложений разными пользователями информационных моделей.

Прикладным протоколом в STEP называют информационную модель определенного приложения, которая описывает с высокой степенью полноты множество сущностей, имеющихся в приложении, вместе с их атрибутами, и выражена средствами языка Express. Предполагается, что эта модель содержит в себе описание данных любой конкретного объекта соответствующего приложения, т.е. практические информационные модели прикладных задач оказываются частными случаями прикладных протоколов. Другими словами, прикладной протокол выражает онтологию приложения, поскольку под онтологией понимают совокупность концепций, объектов, отношений и ограничений, выражающих семантику определенной предметной области.

Прикладные протоколы в стандарте ISO 10303 содержатся в томах, начиная с *N*=201. Прикладные протоколы принято обозначать аббревиатурой AP с указанием номера, например, AP203, AP214. Для связи прикладной системы со STEP используемые ею данные должны быть описаны в соответствующем AP.

Как правило, прикладной протокол включает большое число сущностей и их атрибутов, описания AP составляют десятки страниц на языке Express или десятки рисунков на языке Express-G. Поэтому целесообразно использовать

приемы группирования тесно взаимосвязанных сущностей для более лаконичной характеристики АР. Такими группами являются единицы функциональности (UoF - Units of Functionality). Используют также понятие классов (CC – Conformance Classes) для классификации используемых моделей.

Ниже дана краткая характеристика большинства имеющихся к настоящему времени прикладных протоколов. Их число может расширяться за счет разработки новых протоколов.

AP201: *Explicit draughting*; явное черчение. При использовании протокола оперируют такими понятиями, как структура чертежа, аннотация, геометрическая форма детали, группирование. В число сущностей входят спецификация, утверждение, номер листа, организация-исполнитель, слой, вид и т.п.

AP202: *Associative draughting*; ассоциативное черчение. Протокол, относящийся к описанию конструкторской документации. В протоколе фигурируют данные, в значительной мере пересекающиеся с данными протокола AP201 и сгруппированные по UoF следующим образом:

1) структура документации (иерархия, заголовки, утверждающие подписи);
2) связь с изделием (версия, изготовитель);
3) аннотация формы изделия (2D или 3D CAD-модель);
4) связь модели с ее визуализацией (масштаб);
5) форма аннотации (месторасположение аннотации, символы, заполняемые позиции);
6) оформление документов (шрифты, цвета);
7) размеры (допуски);
8) группирование деталей по тем или иным признакам.

AP203: *Configuration controlled design*; проектирование с управлением конфигурацией. Это один из важнейших прикладных протоколов. В нем унифицированы геометрические модели, атрибуты и спецификации: сборок; 3D поверхностей, разделенных на несколько классов; параметры управления версиями и внесением изменений в документацию и др.

Описание протокола AP203 на языке Express представляет собой схему, в которой можно выделить следующие части.

1. Ссылки на заимствованные из стандартов ISO 10303-41, 42 и 44 интегрированные ресурсы. Это ссылки на такие сущности, как контексты приложения и продукции, свойства изделий, массо-габаритные характеристики, расположение координатных осей, типы кривых и поверхностей, указатели статуса контракта, предприятия, исполнителей, даты и т.п.

2. Описания некоторых обобщенных типов, объединяющих с помощью оператора SELECT ряд частных типов.

3. Описания сущностей, выражающих конструкции изделий. Представлены 6 классов геометрических моделей. *Класс 1* предназначен для задания состава изделий без описания геометрических форм. *Класс 2* включает каркасные модели с явным описанием границ, например, в виде координат точек и определяемых с их помощью линий. В *классе 3* каркасные модели дополнены топологической информацией, т.е. данными о том, как поверхности, линии или точки связаны друг с другом. *Класс 4* служит для описания поверхностей произвольной формы. *Классы 5 и 6* включают твердотельные модели, так называемые BREP (Boundary representation). К первому из них относятся тела, границы которых аппроксимированы полигональными (фасеточными) поверхностями, состоящими из плоских участков. В классе 6 поверхности, ограничивающие тела, могут быть

как элементарными (плоскими, квадратичными, тороидальными), так и представленными моделями в форме Безье, B-сплайнов и др.

4. Описание других используемых сущностей, относящихся к конфигурации изделия, например, таких как вносимое в проект изменение с соответствующими атрибутами.

AP204. *Mechanical design using boundary representation*; конструирование механических деталей на основе твердотельной модели. В протоколе введены такие сущности, как имя изделия, шифр, версия, сборочный узел, модель (элементарная, фасеточная или универсальная BREP-модель), цвет, ширина линий представления и т.п.

AP205: *Mechanical design using surface representation*; конструирование механических деталей на основе поверхностной модели. Ряд понятий, используемых в этом протоколе, аналогичен понятиям протокола AP204, но используются поверхностные модели с границами, представленными геометрически или топологически.

AP207: *Sheet metal die planning and design*; проектирование штампов для листовой штамповки.

AP208: *Life cycle management - Change process*; управление процессами изменений в жизненном цикле (управление конфигурацией). Включает идентификацию фактов (недостатков), требующих внесения изменений, их причин, определяет действия по устранению недостатков и лиц, вносящих изменения.

AP209: *Composite and metallic structural analysis and related design*; анализ композитных и металлических конструкций; комбинирование данных геометрии и управления конфигурацией с данными для анализа, например, по методу конечных элементов. Поддерживаются статический и частотный анализ, 3*D* сеточные модели для анализа по МКЭ, вводятся определения свойств сборок, средства для представления свойств композитных и однородных материалов.

AP210: *Electronic assembly, interconnect and packaging design*; компоновка и проектирование межсоединений в электронной аппаратуре, управление конфигурацией и представление данных о печатных платах и сборках при их проектировании и при передаче данных на производственную стадию. В протоколе используются данные о форме и материале изделия, размещении компонентов и имеющихся ограничениях, проводящих и изолирующих слоях, вносимых изменениях в проект и т.д.

AP211: *PCA Integrated Diagnositics and test*; тестирование и диагностика электронной аппаратуры.

AP212: *Electrotechnical design and installation*; проектирование и монтаж электротехнических изделий. В протоколе описываются электротехнические системы на стадиях проектирования, монтажа, поставки. Имеются средства для представления функциональной декомпозиции систем, физического размещения оборудования и кабельных соединений, информационного обмена между частями систем, документирования, управления конфигурацией и др. (Но в протоколе не рассматриваются вопросы изготовления, моделирования, тестирования аппаратуры). Примеры используемых в стандарте объектов: электротехнические системы и приборы, функциональный продукт, место размещения (installation_location), сигнал, терминал, проект, контракт, интерфейс, цепь, соединение (connectivity), порт. Отдельную группу составляют объекты, представляемые графически, и др.

В протоколе описывается ряд опций, которые могут быть использованы в моделях. Состав этих опций зависит от класса формы. Всего в протоколе 4 класса (CC - conformance classes):

CC1 - проектные данные (классификация, конфигурация, документация с двумерными схемами, структура) без функциональных аспектов и инсталляции;

CC2 – класс 1 с добавлением функциональной информации (распределение функций между частями системы, информационные потоки и др.);

CC3 - класс 1 с информацией об инсталляции (двумерные чертежи с геометрической и пространственной информацией, схемы размещения оборудования);

CC4 – полная совокупность данных – единиц функциональности протокола AP212, т.е. объединение CC1, CC2 и CC3.

AP213: *Numerical control process plans for machined parts*; проектирование обработки на оборудовании с числовым программным управлением. В протоколе введены средства для описания производственных операций, технологического оборудования и инструментов, материалов, геометрических форм и допусков изделий, рабочих мест, сопроводительных административных данных.

AP214: *Core Data for Automotive Mechanical Design Processes*; основные данные для проектирования механических частей автомобилей. Имеются средства для представления данных по структуре и геометрии изделий, презентации проектов, моделированию, производственным процессам (числовое управление, допуски, материалы) и др.

В стандарте введено 19 классов моделей (Conformance Classes), классы различаются видом модели (поверхностная, твердотельная, каркасная), наличием данных по кинематике, допускам, управлению конфигурацией.

Геометрические группы родственных понятий (сущностей, атрибутов), фигурирующих в приложении, сведены в AP214 в несколько UoF, имеющих непустые пересечения. Это:

G1: wireframe_model_2*d*, включающая такие сущности, как геометрическая модель, точка, линия, кривая, гипербола, B-сплайн, 2*D* каркасная модель и др.;

G2; wireframe_model_3*d* с аналогичными сущностями, но в пространстве 3*D*;

G3: connected_surface_model, предназначена для представления топологически ограниченных поверхностных моделей, эта группа включает ряд сущностей из G2 и G8 и таких, как кривая или точка на поверхности, цилиндрическая и тороидальная поверхности, конструктивная геометрия и др.

G4: faceted_b_rep_model, относится к BREP моделям с деталями, имеющими планарные поверхности и внутренние пустоты. Понятия точки, линии, плоскости взято из G3 и G5, другие сущности – замкнутая фасеточная оболочка, твердотельное BREP многообразие (manifold solid B-rep) и др.

G5: b_rep_model - представление одного или более тел, каждое из которых состоит из замкнутых внешней и внутренних оболочек. Геометрия поверхностей выражена кривыми. Большинство понятий аналогично используемым в G3.

G6: compound_model - модели поверхностные, твердотельные, каркасные с топологически представленными соединениями. Примеры использования: выделение в телах зон с различными свойствами, частей сварной конструкции и т.п.

G7: csg_model, или более полное название solid model using Constructive Solid Geometry – получение модели с помощью булевых операций над заданными телами. Наряду с понятиями из предыдущих UoF здесь фигурируют понятия блок, примитив, результат булевой операции и др.

G8: geometrically_bounded_surface_model UoF – геометрически ограниченная поверхностная модель.

Среди других UoF можно отметить:

S2: element_structure - элементы структуры и аннотаций структуры, например, слой, образец, аспект формы, преобразование 2D или 3D, точность, расположение осей и т.п.

S5: work_management с такими сущностями, как операция, метод операции, контракт, порядок работ, изменение.

S6: classification с понятиями классификации атрибутов и систем, иерархии и пунктов классификаций.

S7: specification_control - управление спецификациями предназначено для описания свойств продуктов, имеющих большое число вариантов. Описываются классы продуктов, категории характеристик, способы декомпозиции продукции, ее функции, вводятся сущности конфигурация, проектное ограничение, проектное решение, пункт решения, вариант размещения, спецификация и т.п.

AP215. *Ship arrangement*: расположение частей судна. Затрагиваются такие аспекты, как декомпозиция на пространственно выделенные части (например, грузовые отсеки, машинное отделение, каюты, переборки), форма корпуса, водоизмещение и т.п.

AP216. *Ship moulded form*; форма судна. Описываются общие характеристики, размеры, гидростатика, протяженные внутренние поверхности, геометрия надстроек.

AP217. *Ship piping;* система судовых трубопроводов. Протокол относится к геометрии трубопроводов, их прочности, материалам, анализу потоков, управлению конфигурацией при их проектировании.

AP218. *Ship structures*; устройство судна. В этом приложении рассматриваются характеристики внутреннего устройства судна.

AP220: *PCA Manufacturing Planning*; производство печатных плат и сборок. Вводятся средства для представления 2*D* геометрии размещения компонентов, допусков, операций изготовления, измерения_и т.п.

AP221: *Functional data and their schematic representation for process plant*; функциональная модель и ее схемное представление для производственных процессов. Протокол предназначен для описания иерархического построения предприятий химического, нефтеперерабатывающего производства, атомной энергетики. Рассматриваются состав оборудования, система трубопроводов, характеристики потоков в них.

AP223: *Exchange of design and manufacturing product information for casting parts*; обмен проектными и технологическими данными для литейного производства. В протоколе предусмотрены следующие аспекты приложения: литье в песчаные формы, моделирование процессов литья, литейное оборудование и материалы, процессы плавления, заливки, охлаждения, экстракции, контроль и тестирование.

AP224: *Mechanical product definition for process plans using machining features*; описание механических деталей для планирования станочной обработки. Имеются средства для описания особенностей конструкции деталей (например, отверстий, бобышек, буртов), требований к качеству обработки, свойств материалов, геометрической формы и др. В протоколе выделены следующие основные единицы функциональности: особенности объекта обработки и свойства обрабатываемых заготовок (UoF включает такие сущности, как выступы, фаски, отверстия, путь обработки, параметры материала,

обрабатываемой поверхности, процесса и др.), характеристики обработки (сущности, задающие форму и размеры материала, удаляемого при обработке), допуски на контролируемые параметры, характеристики профиля (сущности, позволяющие по 2D профилям получать 3D формы), управляющая документация (например, требования заказчика, порядок использования ресурсов), внесение исправлений в документацию, административные данные (автор, организация, утверждение), реквизиты (описание заказа на необходимые производственные ресурсы).

AP225: *Building elements using explicit shape representation*; элементы строительных конструкций с явным представлением их формы.

AP226. *Ship mechanical systems*; корабельное механическое оборудование. С помощью определений этого протокола описываются силовые установки, электрогенераторы, насосы, компрессоры, соединения компонентов, их функции, параметры шума и вибраций и т.п.

AP227. *Plant spatial configuration*; пространственная конфигурация предприятий.

AP228: *Building services: Heating, ventilation, and air conditioning*; инженерные службы в строительстве - теплоснабжение, вентиляция, кондиционирование воздуха.

AP231. *Process design and process specifications of major equipment*; проектирование и описание основного оборудования. Протокол относится к концептуальному проектированию, контролю, моделированию, материалам оборудования предприятий химической и нефтегазовой промышленности.

AP232. *Technical data packaging core information and exchange*; представление и обмен технических данных. Протокол посвящен взаимодействию систем управления данными разных проектирующих систем. Объектами описания служат проектные данные как выраженные средствами прикладных протоколов, так и не соответствующие стандартам STEP. Это чертежи, программы для оборудования с ЧПУ, модели проектируемых объектов, спецификации, бизнес-документация и др.

AP233. *Systems engineering data representation* – системы представления инженерных данных. Имеются в виду данные (единицы функциональности), характеризующие состояния системы и ее параметры (например, цена, производительность, надежность, технологичность, контролепригодность и т.п.), связанные с требованиями к продукту, его функциональной архитектурой, поведением, управлением конфигурацией. Рассматриваются как количественные, так и лингвистические (в том числе нечеткие) переменные вместе с единицами измерения.

Типовые фрагменты информационных моделей

В прикладных протоколах широко используются типовые фрагменты информационных моделей, встречающиеся более чем в одном приложении. Эти фрагменты называют интегрированными общими и прикладными ресурсами.

Четвертая группа стандартов STEP (тома с номерами 41...50) "Интегрированные общие ресурсы" описывает общие для приложений части моделей.

Тома с номерами 101 по 199 отведены для документов, относящихся к более специальным средствам, называемым интегрированными прикладными ресурсами (Integrated application resources).

Группа стандартов с номерами, начинающимися с N = 501, служит для описания данных о геометрических элементах и моделях некоторых конкретных типовых объектов и конструкций, часто используемых в ряде интегрированных ресурсов и прикладных протоколов. Например, описания геометрических объектов в виде поверхностей Безье или *B*-сплайнов могут использоваться во многих прикладных протоколах. Поэтому подобные общие описания вынесены в группу прикладных компонентов.

Ниже приведены краткие сведения об основных протоколах STEP, описывающих интегрированные ресурсы и прикладные компоненты.

***N*=41**: *Fundamentals of product description and support*; основы описания и поддержки изделий. В протоколе определяются такие понятия и группы сущностей, как продукт, представление формы (shape_representation), операция (action), контекст - аспект описания (application and product context), статус утверждения (approval), контракт, дата, типы документов, исполнители (организации и персоналии), единицы измерения длин, площадей, масс, температур и др.

***N*=42**: *Geometric and topological representation*; представление геометрии и топологии. В стандарте определен ряд сущностей, их набор близок к набору, используемому в таком стандарте, как IGES. В частности, используются следующие понятия: положение координатной оси (axis_placement), модели кривых в форме B-сплайна (b_spline_curve) и Безье (bezier_curve), модели поверхности в форме B-сплайна (b_spline_surface), рационального B-сплайна (rational_b_spline_surface) и Безье (bezier_surface), точка в декартовых координатах (cartesian_point), преобразование декартовых координат (cartesian_transformation_operator_3d), геометрический аспект (geometric_representation_context), полигональная поверхность (offset_surface), поверхность вращения (surface_of_revolution) и др.

***N*=43**: *Representation structures*; представление структур. Средства описания аспектов документации, ее атрибутов, составных частей и т.п.

***N*=44**: *Product structure configuration*; конфигурация структуры изделий.

***N*=45**: *Materials*; представлены свойства материалов.

***N*=46**: *Visual Presentation*; визуализация. Стандарт построен на базе положений, ранее принятых в стандартах GKS (Graphic Kernel System) и PHIGS (Programmer's Hierarchical Interactive Graphic System). Вводятся группы терминов, относящихся к представлению (presentation), визуализации (visualization), цвету и др.

***N*=47**: *Shape variation tolerance*; допуски.

***N*=48**: *Form features*; свойства форм.

***N*=49**: *Process structure and properties*; структура процессов и свойства.

***N*=101**: *Draughting*; определяются сущности, относящиеся к процедурам черчения.

***N*=104**: *Finite element analysis*; анализ по методу конечных элементов. Описание стандарта на языке Express состоит из нескольких схем. В одной из них задаются геометрические аспекты модели. Здесь описываются следующие типы данных: система координат (декартова, цилиндрическая, сферическая); виды конечных элементов (объемный, поверхностный 2*D* или 3*D,* участок 2*D* или 3*D* кривой), форма элемента (линейный, квадратичный, кубический); степень свободы; параметры и дескрипторы элементов, позиция элемента, свойства элементов (например, масса, момент инерции, жесткость), материал и его свойства (плотность, эластичность, тепловое расширение), группа элементов и др.

В другой схеме основное внимание уделено математическим представлениям. Например, здесь фигурируют такие сущности и типы данных, как тензоры; переменные, характеризующие напряжения; приложенные нагрузки; погрешности; шаги анализа и т.п.

***N*=105**: *Kinematics*; кинематика.

***N*=106**: *Building construction core model*; основная модель строительных конструкций.

***N*=107:** *Engineering analysis core ARM*; ядро инженерного анализа.

***N*=501**: *Edge-based wireframe*; каркасная модель на основе граней..

***N*=502**: *Shell-based wireframe*; каркасная модель на основе оболочек.

***N*=503** (CD): *Geometrically bounded 2D wireframe*; 2D каркасная модель с геометрически заданными границами.

***N*=504**: *Draughting annotation*; чертежные аннотации.

***N*=506**: *Draughting elements*; чертежные элементы.

***N*=507:** *Application interpreted construct: Geometrically bounded surface*; геометрически ограниченные поверхности.

N=508: Application interpreted construct: Non-manifold surface - Поверхности,описанные не в-BREPS-форме.

N=509: Application interpreted construct: Manifold surface - BREPS-поверхности.

***N*=510**: *Geometrically bounded wireframe*; геометрически ограниченная модель поверхности.

***N*=511**: *Topologically bounded surface*; топологически ограниченная модель поверхности.

***N*=512**: *Faceted boundary representation*; полигональное представление поверхностей деталей..

***N*=515**: *Constructive solid geometry*; конструктивная геометрия.

4. Лингвистическое обеспечение CALS-технологий

Состав лингвистического обеспечения

Языки и форматы данных в CALS-системах предназначены для представления данных об изделиях и процессах на различных этапах жизненного цикла промышленной продукции.

Основным языком, используемым в САПР для описания моделей изделий, является язык Express. Кроме него, для описания геометрических моделей и обмена графическими данными в САПР используют ряд других форматов, среди них довольно популярен формат IGES (Initial Graphics Exchange Specification) и форматы, разработанные отдельными фирмами и получившие распространение вместе с продуктами САПР этих фирм. Таковы, например, формат DXF (компания Autodesk) или языки, используемые в графических ядрах машиностроительных САПР.

В пакетах иллюстративной и презентационной графики в соответствии со стандартом MIL-STD-1840C разрешено применять форматы BMP, JPEG, CGM.

Особое место в автоматизированных системах занимает языки и средства, используемые для создания и редактирования разнообразной документации, имеющей текстовый характер с возможными добавлениями графической и мультмедийной информации. К лингвистическому обеспечению описания и

обработки документов относят средства, реализующие следующие группы операций над документами:

- представление (визуализация) информации;
- передача документов между пользователями в распределенных средах;
- создание, редактирование, удаление данных.

Представление информации осуществляется с помощью языков разметки. Исторически первым среди них был язык SGML (Standard Generalized Markup Language), основа которого была разработана Ч.Гольтфарбом, Э.Мошером и Р.Лори в IBM еще в 60-е годы, а в 80-е годы SGML стал основой стандарта ISO 8879. Язык HTML (HyperText Markup Language) разработан Тимом Бернерсом Ли в 1989 г. В 1996 г. консорциум W3C предложил язык XML (eXtensible Markup Language), ставший основным языком структурирования документов в Web-технологиях. В качестве средств форматирования документов при их выводе на экран дисплея используют каскадные таблицы стилей (CSS) или язык XSL.

Для моделирования структуры документов, доступа к ним и их обработки созданы специфик□ция DOM, языки XPath и XQuery.

Язык Express

Базовый язык Express является объектно-ориентированным, имеет универсальный характер, его можно использовать для описания статических структур и их свойств в различных предметных областях, несмотря на то, что язык разрабатывался прежде всего в качестве средства представления моделей промышленных изделий на разных этапах их жизненного цикла.

Описание некоторого приложения на языке Express в рамках стандарта STEP называют *моделью (model)*. В модели декларируются множества понятий и объектов, входящих в приложение, свойства и взаимосвязи объектов.

Модель состоит из одной или нескольких частей, называемых *схемами (schema)***.** Схема - раздел описания, являющийся областью определения данных. В ней вводятся необходимые типы данных. При описании свойств типов данных могут применяться средства процедурного описания - процедуры, функции, правила, константы.

Описание схемы начинается с заголовка, состоящего из служебного слова **schema** и идентификатора - имени схемы. Далее следует содержательная часть - тело схемы. Описание заканчивается служебным словом **end_schema** (в этом и последующих примерах служебные слова языка Express выделены полужирным шрифтом, в скобках < и > записываются нетерминальные символы)**:**

schema <имя схемы>;

<тело схемы>;

end_schema;

В языке Express-G схема представляется прямоугольником с разделительной горизонтальной линией, над этой линией записывается имя схемы, как это показано на рис. 2.

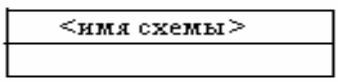

Рис. 2. Изображение схемы в языке Express-G

В теле схемы декларируются *типы данных (Data Type)*. Тип данных - это множество значений некоторой величины или множество объектов (набор экземпляров). В языке Express используются следующие типы данных: сущность *(Entity),* простой (*Simple Type*), агрегативный (*Aggregation Data Type*),

определяемый (*Defined Data Type*), нечисловой (*Enumeration Data Type*) и выделяемый (*Select Data Type*) типы.

Сущность - тип данных, представляющий набор концептуальных или реальных физических объектов с некоторыми общими свойствами. Сущности используют для описания объектов приложений. Свойства сущности выражают в виде *атрибутов (Attributes)*. К характеристикам сущностей относятся также ограничения, накладываемые на значения атрибутов или на отношения между атрибутами.

Описание сущности начинается со служебного слова **entity**, за которым следуют идентификатор сущности, описания ее атрибутов и возможно также правил. Каждый из атрибутов представлен его идентификатором и типом:

entity <имя сущности**>;**

<идентификатор атрибута>:<тип атрибута>;

. . .

end_entity;

Например, задание прямой линии (line) в виде двух инцидентных точек p0 и p1 (атрибутов типа point) выглядит следующим образом:

entity line;

p0,p1: point;

end_entity;

Атрибуты и переменные сами могут быть сущностями, так тип атрибутов p0 и p1 предыдущего примера декларируется, как сущность, атрибутами которой в случае пространства 3*D* являются геометрические координаты x,y,z:

entity point;

x,y,z: **real**;

end_entity;

В Express-G сущности изображаются прямоугольниками, внутри прямоугольника записывается имя сущности (рис. 3).

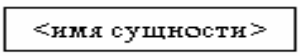

Рис. 3. Изображение сущности в языке Express-G

Если свойство является необязательным для данной сущности, то его выражают так называемым *необязательным (optional) атрибутом*. В его описании перед типом атрибута добавляется служебное слово **optional**

<идентификатор атрибута>: **optional** <тип атрибута>;

Изображение атрибутов в Express-G поясняет рис. 4, из которого, в частности, следует, что атрибут представлен прямоугольником, а связи "сущность-атрибут" или "сущность-сущность" отображаются линиями, причем в случае связи с **optional** атрибутом используется пунктирная линия. Направление связи обозначается окружностью на конце линии, ведущей к атрибуту. Имя атрибута записывается рядом с этой линией. В прямоугольнике атрибута записывается тип атрибута.

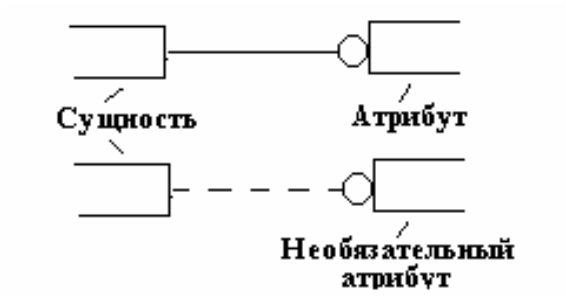

Рис. 4. Изображение атрибутов в языке Express-G

Некоторые из атрибутов могут определяться через другие атрибуты. Тогда атрибуты, выражаемые через другие атрибуты, называют *порожденными (derived)*, что отображается служебным словом **derive** в декларации атрибута. Например, описание окружности, кроме обязательных атрибутов, которыми в нижеследующем примере выбраны радиус и центр окружности, может включать порожденный атрибут площадь круга:

entity point**;**
x,y,z: **real;**
end_entity;
entity circle;
center: point;
radius: **real;** - - явные атрибуты center, radius
derive
area: **real** := pi*radius**2; **(*** порожденный атрибут area *)
end_entity;

Отметим, что между символами (* и *) записывается *комментарий* - произвольный текст по усмотрению автора модели. Если комментарий умещается в одной строчке, то достаточно перед его текстом поставить двойной дефис (- -).

К *простым типам данных* относятся следующие типы:

integer (целые числа);

real (вещественные числа);

number - тип, объединяющий типы **integer** и **real**;

logical - его значениями могут быть **true, false** или **unknown** (неопределенность);

Boolean - с возможными значениями **true** или **false**;

binary - последовательность битов 1 или 0;

string - строка символов.

Изображения простых типов на языке Express-G показаны на рис. 5.

Для **binary** и **string** в круглых скобках можно указать максимально возможное число элементов множества, например, если строка A может включать до 24-х символов, то:

A: **string**(24)**;**
если ровно 24 символа, то
A: **string**(24) **fixed;**
если ограничений нет, то
A: **string;**

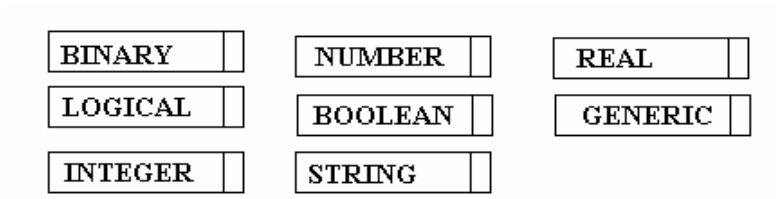

Рис. 5. Изображения простых типов в языке Express-G

Если переменная x имеет тип **binary**, то выражение x[5:7] означает биты с 5-го по 7-й в коде x.

Значения простых типов выражаются с помощью литералов. *Литералы* - это числа (целые, вещественные), двоичные коды, логические значения (**true, false**, **unknown),** фрагменты текста (строковый тип). Примеры записи литералов:

двоичный (начинается с знака %) %100101110

целое десятичное число 1052

вещественный (обязательна десятичная точка) 34.e-3 или 0.034

строковый (занимает не более одной строки) 'first name'

Агрегативный тип данных - множество элементов некоторого типа.

Различают четыре разновидности агрегативных типов, сведения о которых приведены в табл. 1.

Таблица 1

Тип данных	Упорядоченность	Различие элементов
array	Да	Необязательно
bag	Нет	Необязательно
list	Да	Обязательно
set	Нет	Обязательно

При описании типа *array* после слова **array** в квадратных скобках указываются нижняя и верхняя границы индексов. Для остальных агрегативных типов записываются не граничные значения индекса, а нижняя и верхняя границы числа элементов. Например:

F1: **array**[2:8] **of real;** (*описание семиэлементного массива F1, его элементы имеют тип **real** и нумеруются, начиная с значения индекса 2*);

F2: **list**[1:?] **of integer**; (*множество F2 содержит, по крайней мере, один элемент типа **integer**; *)

matr: **array**[1:10] **of array**[9:12] **of** atrac**;** (*массив matr состоит из 10 четырехэлементных массивов, элементы типа atrac.*)

Записи вида **array**[2:8] или **list**[1:?] в Express-G преобразуются в форму **A**[2:8] или **L**[1:?], указываемую около линии атрибута агрегативного типа после имени этого атрибута. Так, первый из вышеприведенных примеров представлен на рис. 6.

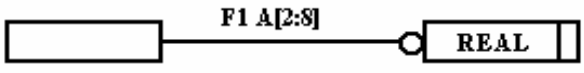

Рис. 6. Пример изображения агрегативного типа в языке Express-G

Определяемый тип данных обычно вводится пользователем для улучшения читаемости модели. *Нечисловой тип* - тип данных, экземплярами которого являются нечисловые (предметные) переменные. *Выделяемый тип* соответствует

поименованной совокупности других типов. Описание этих типов данных начинается со служебного слова **type,** за которым следует идентификатор типа и его определение. Пример описания определяемого типа:

type volume = **real;**
end_type;
entity manual**;**
name**: string;**
v1,v2,v3: volume;
end_entity;

Определение нечислового типа начинается со служебных слов **enumeration of,** после которых в скобках перечисляются элементы множества. Например:

type color = **enumeration of**
(red, green, blue);
end_type;

Ссылка на значение red теперь возможна в виде red или color.red.

Выделяемый тип соответствует одному из некоторого списка уже введенных типов. Этот список записывается после служебного слова **select.** Ссылка на имя выделяемого типа означает, что выбирается один из типов совокупности:

type a_c = **select** (one, two, three);
end_type;
. . .
proc: a_c; (* proc может быть объектом одного из типов one, two, three*)

Графические изображения определяемых, нечисловых и выделяемых типов данных показаны на рис. 7. Внутри прямоугольников, ограничиваемых пунктирными линиями, записывается имя типа.

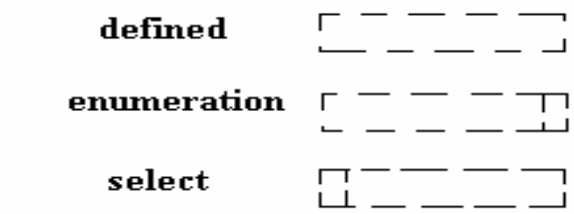

Рис. 7. Изображения определяемых, нечисловых и выделяемых типов данных в языке Express-G

Отношения агрегирования (типа целое-часть) или обобщения (функция-вариант реализации), характерные для представления структур объектов в виде альтернативных (И-ИЛИ) деревьев, в языке EXPRESS выражаются в форме отношений между типами данных. Для этого введены понятия *супертипа (supertype)*, как более общего типа, и *подтипов (subtypes)*, как подчиненных типов. На рис. 6.7 верхняя сущность относится к супертипу, а три нижних прямоугольника изображают подтипы, линии связи прямоугольников должны быть утолщенными.

Рассмотрим пример фрагмента И-ИЛИ-дерева, в котором имеется ИЛИ вершина a1 и две подчиненные ей альтернативные вершины b1 и b2. Общим атрибутом для b1 и b2 является size типа **real**, специфичный для b1 атрибут - vol типа **real**, а специфичный для b2 атрибут met типа **string**. Этот фрагмент может быть описан следующим образом:

entity a1
supertype of (oneof (b1,b2**));**
size: **real;**

end_entity;
entity b1
subtype of (a1**);**
vol: **real;**
end_entity;
entity b2
subtype of (a1**);**
met: **string;**
end_entity;

Используются также следующие правила записи супертипов и подтипов:

в случае, если a1 есть И вершина, вместо **oneof** используется зарезервированное слово **and** (в более общем случае **andor**), т.е. вторая строчка примера будет выглядеть так:

supertype of (b1 **and** b2**);**

если между подтипами нет взаимосвязи, выражаемой логической функцией (в частности, ИЛИ или И вершинами), то указание в a1 факта, что это супертип, не требуется; достаточно упоминание о подчиненности подтипов в их декларациях в виде **subtype of** (a1);

перед декларацией **supertype** 犆аписывается зарезервированное слово **abstract**, если вершине a1 не соответствуют какие-либо экземпляры сущности, т.е. если a1 введена только для указания общих для подтипов атрибутов;

у одного подтипа может быть больше одного супертипа; подтип наследует атрибуты всех своих супертипов; если в декларациях супертипов используются одинаковые идентификаторы атрибутов, то ссылка на них должна быть в виде составного идентификатора, например, a1.size.

Пример:

entity device
supertype of (oneof (transistor, diode**));**
(* device есть ИЛИ вершина И-ИЛИ-дерева с двумя альтернативами transistor и diode*)
end_entity;
entity transistor --
subtype of (device);
b: **real;**
end_entity;
entity diode
subtype of (device)**;**
r: **real;**
end_entity;

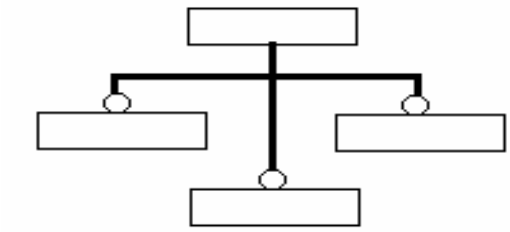

Рис. 8. Изображение супертипов и подтипов в языке Express-G

Ограничения, накладываемые на экземпляры сущности, выражаются с помощью *правил (rules)*. Правила могут быть общими или локальными.

Описание правила, общего для ряда сущностей, начинается со служебного слова **rule**, далее следуют идентификатор правила, служебное слово **for**, ссылки на сущности, на которые правило распространяется, и, наконец, собственно ограничения.

Локальные правила могут описывать ключевые атрибуты (*uniqueness rules*) или выражать ограничения, накладываемые на атрибуты некоторой сущности (*domain rules*). Например, если ключевой атрибут сущности Z есть составной атрибут X.Y, или, другими словами, одному сочетанию значений атрибутов X и Y должен соответствовать единственный экземпляр сущности Z, то

entity Z;
X: **integer**;
Y: **string**;
unique
X,Y;
end_entity;

Ограничение на атрибуты некоторой сущности выражается с помощью правила в теле этой сущности. Ограничение записывается после слова **where** в виде выражения, значениями которого могут быть **true, false** или **unknown**. Допустимыми значениями атрибута будут только те, для которых выражение принимает значение **true**. Например, можно записать, что длина вектора vect = (x,y,z) должна быть равна единице, в виде правила cons:

entity vect;
x,y,z: **real**;
where
cons: x**2 + y**2 + z**2 = 1.0;
end_entity;

Ограничение where можно использовать в определяемых типах, например:

type size =**real**;
where SELF< 12.0;
end_type;

где служебное слово **SELF** заменяет идентификатор определяемого типа, т.е. в данном примере значения size должны быть меньше 12.

Процедуры и *функции* служат для описания процедурной части модели. Как и в алгоритмических языках, используется концепция формальных и фактических параметров. Описание процедуры начинается с служебного слова **procedure**, за которым следуют идентификатор процедуры и описание формальных параметров в круглых скобках. Пример описания заголовка процедуры:

procedure eq **(**x,y: **real;** n: **integer; var** result: route**);**

Аналогично описываются функции, их отличает только описание в заголовке типа результата после закрывающей скобки:

function log **(**a**: real;** m: **integer): real;**
Локальные переменные, описанные в блоке
local
...
end_local;
действуют только в пределах данных функции или процедуры.

Ряд функций и процедур относится к стандартным и потому не требует описания во вновь разрабатываемых моделях. Отметим следующие стандартные функции:

Abs - абсолютная величина; **Sqrt** - корень квадратный; **Exp** - экспонента; **Log, Log2, Log10** - логарифмы натуральный, двоичный, десятичный соответственно; **Sin, Cos, Tan, Acos, Asin, ATan** - тригонометрические и обратные тригонометрические функции *sin, cos, tg, Arc cos, Arc sin, Arc tg*.

В число стандартных входят также функции: **BLength** - подсчет числа бит в двоичном коде; **HiBound** - верхняя граница индекса у *array* или верхняя граница числа элементов у *set, bag, list*; **LoBound** - то же в отношении нижних границ; **Length** - подсчет числа символов в строке; **Odd** - возвращает значение true, если аргумент - нечетное число; **SizeOf** - возвращает число элементов в объекте агрегативного типа; **TypeOf** - возвращает список типов, к которым принадлежит параметр этой функции; **Exists** - возвращает значение true, если аргумент этой функции входит в число атрибутов соответствующей сущности, и др.

К стандартным процедурам относятся процедуры **Insert** и **Remove** - вставка или изъятие элемента в заданной позиции у объекта агрегативного типа соответственно.

При описании алгоритмов в телах процедур и функций могут использоваться операторы *пустой (Null*), *присваивания (Assignment*), *выбора (Case*), *составной (Compound Statement*), *условный (if..then..else*), *цикла (Repeat*), *выхода из функции или процедуры (Return*), *перехода на конец цикла (Skip*).

В выражениях используются операнды, знаки операций, вызовы функций. Так, для арифметических операций над числами типа **real** применяются следующие знаки:

* - умножение, / - деление, DIV - целочисленное деление, + - сложение, - - вычитание, ** - возведение в степень, MOD – деление по модулю.

Знаки логических операций: **not** - отрицание, **and** - конъюнкция, **or** - дизъюнкция, **xor** - исключающее ИЛИ. В применении к величинам типа **logical** эти операции выполняются по правилам действий в трехзначном алфавите. Логическое выражение a1 **in** a2 принимает значение **true**, если a1 содержится в a2. Оператор **like** используется для посимвольного сравнения строк. Знаки отношений равно =, неравно <>, больше >, меньше <, больше или равно >=, меньше или равно <=. Для сравнения экземпляров сущностей используют операции "равно" и "неравно" со знаками :=: и :<>: соответственно.

Операции над множествами (типами **bag** и **set**) – пересечение (*Intersection*), объединение (*Union*), разность (*Difference*). Их знаки суть * (умножение), + (плюс), - (минус) соответственно. Оператор **Query** (A <* B | C) возвращает подмножество тех элементов из агрегативного типа B, для которых выполняется условие C, здесь A – простая переменная, используемая в C.

Знак + (плюс) по отношению к операндам типа **binary** или **string** есть знак конкатенации.

В качестве формальных параметров процедур и функций, кроме типов данных, применяемых в других конструкциях языка и охарактеризованных выше, могут использоваться обобщенные типы: *generic, aggregate* и некоторые другие. Тип *generic* формального параметра означает, что соответствующий фактический параметр может иметь любой тип данных из числа предусмотренных при описании процедуры. Аналогично тип *aggregate* обобщает агрегативные типы данных - *array, bag, list, set*. Например:

function add (a,b: **generic:** intype**): generic:** intype**;**
local
nr: **number;**
vr: **vector;**
end_local;
if ('number' **in typeof (**a**)) and (**'number' **in typeof (**b**))**
then nr := a+b;
(* функция **typeof** (a) возвращает тип аргумента a и, если этот тип есть number, то первый операнд логического выражения равен **true *)**
return (nr);
else
if ('this schema.vector' **in typeof (**a**)) and (**'this schema.vector' **in typeof (**b**)) then**
vr.i := a.i + b.i;
vr.j := a.j + b.j;
vr.k := a.k + b.k;
(* подразумевается, что декларация типа vector была произведена в схеме с именем this schema *)
return (vr);
end_if;
end_if;
end_function;

В языке Express-G специальные символы для изображения правил, процедур и функций не оговорены.

Способ описания констант очевиден из следующего примера:

constant
year: **integer:=** 1995;
start: date := date(12,16,1982); (*подразумевается, что при описании типа date указаны три атрибута месяц, число, год*)
end_constant;

Для установления интерфейса между двумя схемами вводятся спецификации интерфейса. Применяют два типа спецификаций - **use** и **reference.** Например:

schema s1;
entity par1;
name: **string**;
end_entity;
end_schema;
schema s2; (* в схеме s2 в качестве параметра x используется name из s1.par1 *)
use from s1.par1 (name **as** x);
end_schema;

Ссылки типа use отличаются тем, что декларации сущностей из другой схемы используются в данной схеме как свои локальные, в то время как reference просто позволяет обращаться к декларациям другой сущности. Ограниченность reference выражается в том, что сущности из другой схемы можно использовать только в качестве типов атрибутов в сущностях данной схемы.

В языке Express-G используются диаграммы двух уровней. На схемном уровне (schema level) изображаются схемы и их взаимосвязи в виде линий. На сущностном уровне (entity level) изображаются типы, сущности, атрибуты, а для ссылок на объекты другой схемы применяются специальные символы.

Эти символы представляют овальными фигурами. В овале записывают имя схемы-источника и имя используемого определения. В нашем примере это ссылка

на S1.par1. Овал помещается внутрь прямоугольника, в котором дополнительно указывается имя атрибута (в примере это name).

Для указания межстраничной связи, что требуется, если Express-G модель размещается более чем на одной странице, используется овальный символ, внутри которого указываются через запятую номер страницы и номер ссылки.

Примеры моделей на языке Express

Пример 1

Моделируется ситуация:

Фрагмент (fragment) 2D-изображения представлен окружностью (cyrcle) или отрезком кривой (curve), кроме того, свойством фрагмента является цвет (colour). Окружность характеризуется радиусом (rad), отрезок кривой - тремя точками (P).

Полученная модель на языке Express имеет вид:

```
schema DIAG2;

    type form = select
      (cycle,
       curve);
    end_type;

    type colour = enumeration of
       (red,
       blue,
       white);
    end_type;

    entity cycle;
        rad: real;
    end_entity;

    entity curve;
        P: ARRAY [1:3] OF point;
    end_entity;

    entity point;
        X: ARRAY [1:2] OF REAL;
    end_entity;

    entity fragment;
        figure: form;
        image: colour;
    end_entity;
  end_schema;
```

Модель на языке Express-G представлена на рис.9.

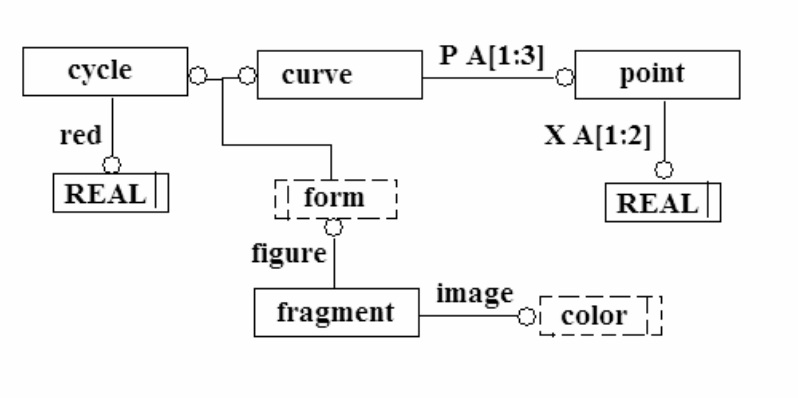

Рис. 9. Диаграмма примера 1 на языке Express-G

Пример 2

Дана модель на языке Express-G в виде схемы рис.10. Соответствующая ей модель на языке Express представлена ниже.

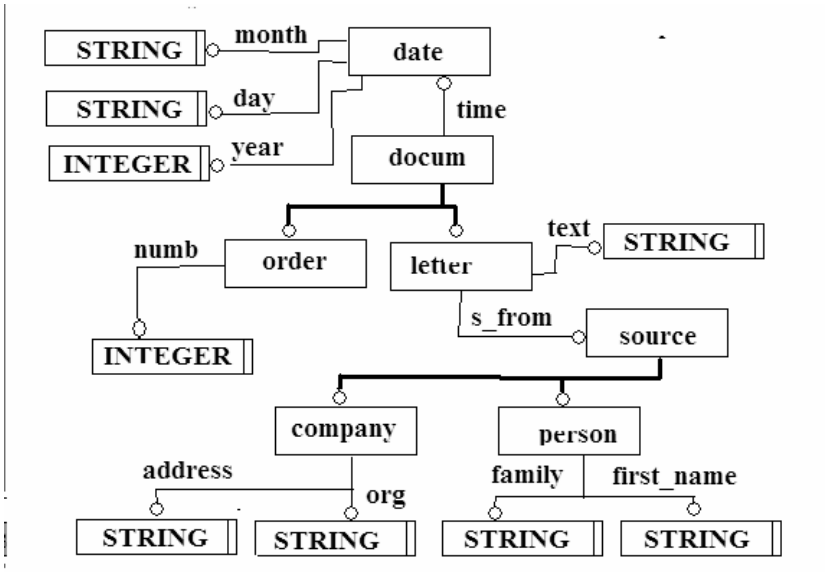

Рис. 10. Диаграмма примера 2 на языке Express-G

```
schema OFF;

  entity docum;
    time: date;
  end_entity;

  entity letter
   subtype of (docum);
    text: string;
    s_from: source;
  end_entity;
```

```
entity order
  subtype of (docum);
    numb: integer;
end_entity;

entity source;
end_entity;

entity company
  subtype of (source);
    address: string;
    org: string;
end_entity;

entity person
  subtype of (source);
    family: string;
    first_name: string;
end_entity;

entity date;
    month: string;
    day: integer;
    year: integer;
end_entity;

end_schema;
```

Пример 3.

Модель *"person_organization_schema"* на языке Express взята из 41-го тома "Интегрированные ресурсы" стандарта STEP (ISO 10303.41):

```
schema person_organization_schema;
 entity address;
  internal_location : optional label;
  street_number : optional label;
  street : optional label;
  postal_box : optional label;
  town : optional label;
  region : optional label;
  postal_code : optional label;
  country : optional label;
  facsimile_number : optional label;
  telephone_number : optional label;
  electronic_mail_address : optional label;
  telex_number : optional label;
  where
   wr1 : exists(internal_location) or exists(street_number) or exists(street)  or exists(postal_box) or
exists(town) or exists(region) or exists(postal_code) or exists(country) or exists(facsimile_number) or
exists(telephone_number) or exists(electronic_mail_address) or exists(telex_number);
 end_entity;
 entity personal_address
  subtype of (address);
```

```
 people : set[1:?] of person;
 description : text;
end_entity;
entity person;
 id : identifier;
 last_name :  optional label;
 first_name : optional label;
 middle_names :  optional list[1:?] of label;
 prefix_titles :  optional list[1:?] of label;
 suffix_titles :  optional list[1:?] of label;
 unique
  ur1 : id;
 where
  wr1 : exists(last_name) or exists(first_name);
end_entity;
end_schema;
```

Пример 4.

На рис. 11 показан небольшой фрагмент модели из прикладного протокола AP202, относящийся к сущности "Оболочка" и ее атрибутам. Свойства этой сущности - модель, тип и границы поверхности. Тип поверхности - супертип, а подтипами являются возможные варианты. Граница задана в виде контура, выраженного либо своими вершинами, либо совокупностью линий. Связи этого фрагмента с другими частями протокола AP202 на рис. 11 не показаны.

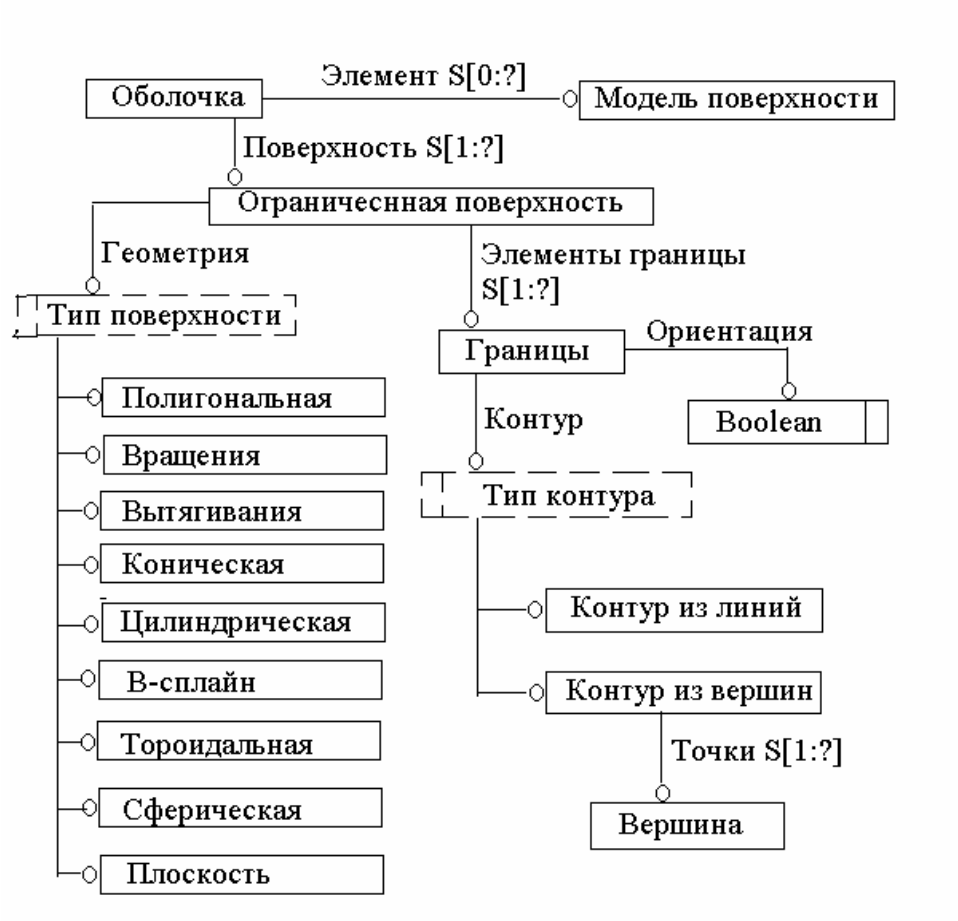

Рис. 11. Фрагмент прикладного протокола AP202 на языке Express-G

Организация в STEP информационных обменов

Возможны обмены через обменный файл и через базу данных SDAI.

Обменный файл используется при связи двух систем *A* и *B*, имеющих общие данные с различными обозначениями. Пользователь должен написать перекодировщик (например, на языке Express-X), с помощью которого отождествляются идентификаторы одних и тех же сущностей, имевших разные обозначения в схемах *A* и *B*.

Связь через интерфейс SDAI отличается от предыдущего способа обмена тем, что здесь имеет место не просто обмен, а разделение данных многими пользователями, и SDAI фактически выступает в роли метамодели для разных САПР. Другими словами, SDAI представляет собой интерфейс, содержащий набор функций, например, на языках C++ и C, для доступа к разделяемым моделям, которые могут быть представлены в виде обменного файла.

Обменный файл введен в протокол ISO10303-21 для обмена конкретными значениями атрибутов. Он состоит из головной и информационной секций. В головной секции (между служебными словами HEADER и ENDSEC) указываются:

Entity file_name - имя и некоторые другие атрибуты данного конкретного обменного файла;

Entity file_description - неформальное описание содержимого файла и требования к ПО для обработки данного файла;

Entity file_schema - схемы, для которых далее даны экземпляры сущностей;

keyword (список типов).

В информационной секции (между словами DATA и ENDSEC) указываются имена экземпляров сущностей и значения их атрибутов в виде следующих строк:

имя сущности = keyword (список параметров);

Например:

#1 = POINT(0.0,0.2,0.5); (* экземпляр сущности типа POINT
с именем 1 имеет значения параметров 0, 0.2 и 0.5 типа REAL.*)

#2 = WIDGET(.RED.); (* экземпляр сущности типа WIDGET с именем 2
имеет значение перечислимого типа RED.*)

. . .

#8 = LINE(#1,#4); (* значениями атрибутов являются экземпляры сущностей с именами 1 и 4.*)

В списке параметров значения перечисляются в том же порядке, в каком они фигурировали в описании сущности.

Сказанное иллюстрирует следующая запись обменного файла для модели примера 2.

```
Обменный файл ISO-10303-21;
        HEADER;
        FILE_DESCRIPTION((),'2;1');
        FILE_NAME('m2','2006-03-10T07:44:25',('ANONYMOUS
USER'),('ANONYMOUS ORGANISATION'),'EXPRESS Data Manager version
20050406',$,$);
        FILE_SCHEMA(('OFF'));
        ENDSEC;

        DATA;
          #1= letter(#2,'This is...',#3);
          #2= date('mart',4,2006);
          #3= company('Moscow, 2d Baumanskaya str., 5','BMSTU');
          #4= letter (#2,'Dear...',#5);
          #5= person('Petrov','Ivan');
          #6= order(#7,24);
          #7= date ('April',2,2005);
        ENDSEC;

        END-ISO-10303-21;
```

Расширение возможностей языка достигается путем введения его разновидностей. Так, в языке Express-C добавляются возможности описания событий и транзакций:

event a;

when b ⇒ c; (* здесь b - логическое выражение, c - обращение к транзакции при b =true*);

end_event;

transaction c;

local d: e;

end_local;

. . .

end_transaction;

При описании соответствия между двумя Express-моделями используются языки Express-X или Express-M. Например, в Express-M соответствие между схемой-источником A, в которой заданы атрибуты a1, a2, a3, и схемой-целью B, в которой те же атрибуты описаны идентификаторами b1, b2, b3, выражается следующим описанием:

shema map B ← A;

b1 := a1;

b2 := a2;

b3 := a3;

end_shema_map;

При отображении возможны преобразования атрибутов, например, если a1 задан в метрах, а b1 в сантиметрах, то в примере нужно записать b1 := a1*100.

Язык SGML

Одним из первых языков разметки был разработан SGML. Этот язык принят в качестве основного языка оформления технической документации, в том числе в ИЭТР, на создаваемые изделия в CALS-технологиях.

В языке SGML определяется структура документов в виде последовательности объектов (элементов) данных. Объекты данных, представляющие части документа, могут храниться в различных файлах. Стандарт SGML устанавливает такие множества символов и правил для представления информации, которые позволяют различным системам правильно распознавать и идентифицировать эту информацию. Названные множества описывают в отдельной части документа, называемой декларацией DTD (Document Type Decfinition), которую передают вместе с основным SGML-документом. В DTD указывают соответствие символов и их кодов, максимальные длины используемых идентификаторов, способ представления ограничителей для тегов, другие возможные соглашения, синтаксис DTD, а также тип и версию документа. Следовательно, SGML можно назвать метаязыком для семейства конкретных языков разметки. В частности, подмножествами SGML можно считать языки разметки XML и HTML.

Однако язык SGML сложен для освоения и использования. Поэтому с целью широкого применения разметки в документах, представляемых в Web-технологиях, на базе SGML был разработан упрощенный язык разметки HTML, и далее язык XML, который стал в сочетании с HTML основным языком представления документов в автоматизированных системах.

Язык HTML.

Описание на языке HTML представляет собой текст в формате ASCII и последовательность включенных в него команд (управляющих кодов), называемых также дескрипторами или тегами. Этот текст называют HTML-документом, или HTML-страницей, или после размещения на Web-сервере - Web-страницей. Теги расставляются в нужных местах исходного текста, они определяют шрифты, переносы, появление графических изображений, ссылки и

т.п. При использовании WWW-редакторов вставка команд осуществляется простым нажатием соответствующих клавиш.

Собственно команды имеют форму <*команда*>, где вместо слова *команда* записывается имя команды.

Структура текста в HTML-странице имеет вид[1]:

<HTML><HEAD>
<TITLE> *Заголовок текста* **</TITLE>**
</HEAD>
<BODY>
Текст HTML-документа
</BODY>
</HTML>

В клиентской области окна при просмотре появляется только текст, помещенный между тегами **<BODY>** и **</BODY>.** Заголовок между тегами **<TITLE>** и **</TITLE>** выполняет лишь служебные функции.

Приведем примеры HTML-тегов. К тегам форматирования текста (тегам компоновки) относятся:

<P> - конец абзаца;

**
** - перевод строки;

<HR> - перевод строки с печатью горизонтальной линии, разделяющей части текста;

<CENTER> - выравнивание изображения по центру страницы;

<LISTING> *Текст* **</LISTING>** - представление листингов программ;

<BLOCKQUOTE> *Текст* **</BLOCKQUOTE>** - выделение цитат;

<**FONT**> - задание типа, размера и цвета используемого шрифта, имена этих параметров (атрибутов) FACE, SIZE и COLOR соответственно.

Теги форматирования символов имеют вид <**B>, <I>, <U**>; при их использовании текст между открывающим и закрывающим тегами будет выделен соответственно полужирным шрифтом, курсивом, подчеркиванием.

Для форматирования заголовков используются теги <H1> ... <H6>:

<H1> *Текст* **</H1>** - текст печатается наиболее крупным шрифтом, используется для заголовков верхнего уровня;

<H2> *Текст* **</H2>** - для следующего уровня и т.д. вплоть до команды **<H6>;**

<PRE> *Текст* **</PRE>** - указанный текст представлен заданным при его записи шрифтом.

В HTML имеются теги форматирования списка. Это теги **<OL>** и **<UL>,** используемые для выделения пунктов списков с нумерацией или с пометкой специальным символом (например, *) соответственно. Каждый пункт в списке должен начинаться с тега **<LI>.** В словарях и глоссариях удобно применять тег **<DL>,** отмечающий начало списка, теги **<DT>** и **<DD>,** отмечающие очередной новый термин словаря и определяющий его текст соответственно.

В командах вставки графики и гипертекстовых ссылок используются адреса вставляемого или ссылочного материала, называемые URL (Uniform Resourse Locator). Ссылаться можно как на определенные места в том же документе, в котором поставлена ссылка, так и на другие файлы, находящиеся в любом месте сети. Перед простановкой внутренней ссылки, т.е. ссылки на некоторую позицию в данном файле, нужно разместить метку в этой позиции. Тогда URL есть указание этой метки, например, **URL = #a35** есть ссылка на метку a35. URL

[1] Далее при описании языка терминальные символы изображаются полужирным шрифтом, а нетерминальные - курсивом.

может представлять собой имя файла в данном узле сети или IP-имя другого узла с указанием местоположения файла в этом узле и, возможно, также метки внутри этого файла.

Строка гипертекстовой ссылки в HTML-документе имеет вид:

<A HREF="*URL***" >***Текст* **</A>**

Текст, указанный в этой строке и отображаемый на экране дисплея, будет выделен цветом или подчеркиванием. Можно ссылаться на определенное место в документе. Тогда

<A HREF="*URL#метка***">** *Текст* **</A>**

Сама метка в документе имеет вид

<A NAME="*метка***">** *Текст* **</A>**

Ссылки на фрагменты данного документа можно упростить

<A HREF="#*метка***" >***Текст* **</A>**

Тег вставки графического изображения:

<IMG SRC="*URL***"[ALIGN=TOP|MIDDLE|BOTTOM][ALT="***text***"]>,**

где *URL* указывает адрес графического изображения, **ALIGN** - параметр выравнивания, указывает место в окне для расположения рисунка; **ALT** - параметр, задающий текст, который выводится на экран вместо рисунка в текстовых браузерах. Например:

<IMG SRC = "fgr.gif"> или

<A HREF = "http://www.abc.ru/de.htm"><IMG SRC = "fgr.gif"></A>

где **fgr.gif** и **www.abc.ru/de.htm** - конкретные имена, взятые для примера.

Кроме параметров **ALIGN** и **ALT** можно использовать параметры **HEIGHT** и **WIDTH**, задающие высоту и ширину изображения (в пикселах), **HSPACE** и **VSPACE**, определяющие размер промежутка между изображением и границами страницы в горизонтальном и вертикальном направлениях, **BORDER**, задающий рамку вокруг изображения. Сами изображения должны быть в определенном формате (обычно это форматы GIF или JPEG).

Экран может быть разделен на несколько окон (областей, фреймов) с помощью парного тега **<FRAMESET>**. В каждом окне помещается содержимое файла (текст, изображение) указанием источника в теге **<FRAME>**, например

<FRAME SRC="*имя файла***">.**

Представление таблиц выполняется с помощью тегов формирования таблиц. Парные теги **<TABLE>** и **</TABLE>** служат для указания начала и конца таблицы; **<TH>** и **</TH>** - то же для шапки таблицы; **<TR>** и **</TR>** - для строки таблицы; **<TD>** и **</TD>** - для элемента таблицы. Для форматирования таблиц используются параметры, записываемые в открывающих тегах и задающие цвет фона, ширину таблицы, расположение текста в ячейках.

Имеются возможности создания на Web-странице формы, в которую пользователи могут заносить информацию, передаваемую браузером на сервер (тег **<FORM>**) или управляющую выбором из меню (тег **<INPUT>**).

Поскольку в языке HTML множество тегов ограниченное и фиксированное, действия, предусматриваемые ими, в частности, операции форматирования, реализованы в браузерах. При этом тегам, подобным **<H1>**, соответствует определенный стиль (тип, размер, цвет шрифта). Чтобы дать возможность пользователям устанавливать желаемый стиль изображения, разрабатывают таблицы стилей, представляющие информацию о параметрах стиля, и способы связывания таких таблиц с HTML-документом. Большинство браузеров поддерживают каскадные таблицы стилей CSS (Cascading Style Sheet).

Таблица CSS состоит из правил форматирования. В каждом правиле указываются тип элемента, к которому относится форматирование, и список объявлений. Список обрамляется фигурными скобками, объявления в списке разделяются точками с запятой. Каждое объявление задает значение одного из свойств отображения элемента в виде *свойство:значение*. К свойствам относятся тип (гарнитура), размер, цвет, способ выравнивания и стиль (обычный, полужирный, курсив) шрифта, цвет или рисунок фона, межстрочные интервалы, наличие рамок, взаимное расположение блоков текста и другие характеристики, обычные для управления видом изображения в текстовых редакторах. Можно вместо типа элемента указать имя оригинального вводимого стиля, имя стиля должно начинаться с точки.

Использование таблицы стилей подразумевает указание типа таблицы в разделе **<HEAD>** HTML-документа. Там же между тегами **<STYLE>** и **</STYLE>** записываются правила форматирования. Можно все правила форматирования записать в отдельном файле и тогда в HTML-документе достаточно сослаться на этот файл в специальном теге **<LINK>**. Если вводимый стиль относится лишь к части документа, используется тег **<SPAN>** с параметром **CLASS**, например:

<SPAN CLASS=*имя вводимого стиля***>** *Часть документа, представляемая вводимым стилем* **</SPAN>**

Первые версии языка HTML были достаточно простыми, но не лишенными ряда недостатков. Прежде всего нужно отметить ограниченность набора тегов, что не соответствует потребностям многих приложений. Кроме того, в тегах HTML не отделены данные, задающие структуру документа, от данных по его изображению (форматированию) на экране дисплея при просмотре с помощью браузера, что затрудняет работу с документами. В результате в новые версии языка стали вводится усовершенствования, что заметно усложнило язык, но не устранило основные недостатки. Наиболее существенными недостатками HTML являются, во-первых, невозможность отделить информацию о структуре документа от информации о форматировании, во-вторых, отсутствие в языке HTML средств, позволяющих производить такие операции обработки текста, как сортировка, поиск фрагментов по определенным признакам и т.п.

Эти недостатки устранены в языке разметки XML.

Язык XML

В настоящее время язык XML претендует на роль основного языка представления документов в информационных технологиях, его можно рассматривать как метаязык, служащий основой для создания частных языков разметки в различных приложениях. При этом XML более удобен, чем SGML, что обеспечивается устранением в XML некоторых второстепенных особенностей SGML. Описания на XML легче воспринимаются, приспособлены для использования в современных Web-браузерах при сохранении основных возможностей SGML.

Для конкретных приложений создаются свои варианты XML, называемые XML-словарями или XML-приложениями. Так, для описания текстов с специфической математической символикой разработано XML-приложение OSD (Open Software Description). Для CALS интерес представляет вариант Product Definition eXchange (PDX), посвященный обмену данными. Известны словари для

химии (CML - Chemical Markup Language), биологии (BSML - Bioinformatic Sequence Markup Language) и др.

Рзличают две степени правильности XML-документов: корректность, как соответствие синтаксическим правилам разметки, и валидность, как соблюдение ряда дополнительных правил и ограничений, Эти дополнения представляют с помощью определений типов данных **DTD** или XML-схем, в которых описывается структура XML-документа, задаются типы используемых в нем элементов и атрибутов. Для написания **DTD** используют довольно простой для освоения язык, но с ограниченными возможностями по сравнению с XML-схемами, называемыми также **XSD** (XML Schema Definition).

XML-документ состоит из пролога, корневого элемента "Документ", собственно и являющегося размеченным документом, и сведений по форматированию.

Рассмотрим построение XML-документа с использованием таблиц **DTD.**

Пролог начинается со строки

<?xml version="1.0" *дополнения* **?>,**

которая указывает используемую версию языка XML (в данном случае версия 1.0). В эту строку можно в качестве дополнения включить также объявление автономности документа

<?xml version="1.0" standalone='yes'?>,

если не предполагается связывать с документом какие-либо внешние файлы. В *дополнениях* (или в отдельной команде) может быть указана используемая кодировка, например, **encoding='ISO 8859-1'**. В пролог могут входить также одна или несколько пустых строк, строки комментария и командные строки. Форма комментария

<!- - *текст комментария* **- ->.**

Текст комментария может включать любые символы, кроме двух дефисов. Командные строки имеет вид

<?*команда* **?>**

и являются указанием XML-процессору на обработку документа.

Элемент "Документ" представляет собой иерархически организованное множество элементов. Элементом называют фрагмент исходного документа, заключенный в контейнер. Контейнер представлен тегами, обрамляющими исходный текст. Образующаяся конструкция вида

<тип* [*атрибуты*]*> содержание </тип >

есть элемент XML-документа, где ***тип*** есть имя элемента, а ***содержание*** - исходный текст. Типы элементов задаются в декларации DTD. Фрагменты могут иметь те или иные атрибуты (параметры), значения которых записываются внутри открывающего тега.

Декларация DTD выполняет ту же роль, что и в языке SGML. В ней указываются средства разметки, с помощью которых структурируют исходный документ. Декларация может быть помещена в отдельный файл и тогда в прологе нужно указать XML-процессору имя этого файла с помощью строки

<!DOCTYPE *имя_документа* **SYSTEM "***имя файла DTD***">**

Но можно декларацию DTD записать непосредственно в эту строку вместо служебного слова **SYSTEM** и имени файла *DTD*, заключив ее в квадратные скобки. Возможно также разделение DTD на внешнюю и внутреннюю части, когда адрес первой из них записывается в поле *имя файла DTD*, а вторая часть помещается после этого в квадратных скобках.

Инструкции по форматированию документа, необходимые для его визуализации с помощью браузера, могут быть заданы несколькими способами. Один из них - использование каскадных таблиц стилей CSS, таких же, какие используют для HTML-документов. В этих таблицах для каждого типа элемента указаны способы визуализации - тип, размер, цвет шрифта, расположение на экране дисплея при просмотре. Таблица CSS помещается в отдельный файл. Ссылка на этот файл в XML-документе размещается в прологе и имеет вид

<?xml-stylesheet type="text/css" href="*имя_файла***"?>,**

где *имя_файла* - имя файла с таблицей CSS.

Пример пролога XML-документа:

```
<?xml version="1.0" ?>
<!- - Это заголовок документа dictionary - ->
<?xml-stylesheet type="text/css" href="dict.css"?>
<!DOCTYPE dictionary SYSTEM "dict.dtd">
```

В заголовке записаны номер используемой версии языка XML (**version="1.0"**), имя документа (в нашем примере **dictionary**), ссылки на файлы, в которых размещены таблицы CSS (файл **dict.css**) и DTD (файл **dict.dtd**):

Декларация DTD отражает структуру XML-документа и состоит из объявлений используемых в документе типов элементов, атрибутов и сущностей.

Типы элементов задаются с помощью строк

<!ELEMENT *тип_элемента содержание***>**,

где *содержание* - либо модель элемента (тип содержимого элемента), как в строке

<!ELEMENT *тип_элемента* **(#PCDATA)>,**

либо список типов элементов, вложенных в данный элемент в иерархической структуре, как в строке

<!ELEMENT *тип_элемента*
(*список_типов_вложенных_элементов***)>** (6.1)

Модель элемента **(#PCDATA)** означает, что элемент может состоять только из символьных данных, модель **ANY** допускает любые данные, а модель **EMPTY** соответствует пустому (бесконтейнерному) элементу. Возможен также смешанный тип содержимого, когда элемент может содержать и символьные данные, и вложенные элементы (т.е. в (6.1) одним из пунктов списка будет **#PCDATA**).

Пример описания типов элементов в DTD (назначение знаков + и ?, имеющихся в примере, будет пояснено ниже):

```
<!ELEMENT metadata (name, authors, type,
  description, owner?, year?, price?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT authors (#PCDATA)+>
<!ELEMENT type (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT owner (#PCDATA)>
<!ELEMENT year (#PCDATA)?>
<!ELEMENT price (#PCDATA)?>
```

Атрибуты служат для характеристики типов элементов. С помощью типов элементов и значений их атрибутов можно сортировать части документа, устанавливать между ними отношения, выделять нужные экземпляры и т.п. Объявление атрибутов, относящихся к определенному типу элементов, имеет вид:

<!ATTLIST *тип_элемента имя_атрибута*
*тип_атрибута статус***>** (6.2)

Атрибуты могут быть строкового, маркерного или нумерованного типов. Строковый тип обозначается **CDATA** и соответствует типу данных string, т.е. атрибут типа **CDATA** в документе должен быть записан в виде строки символов в кавычках. Маркерный тип отличается от строкового тем, что на значения атрибута маркерного типа накладываются некоторые ограничения. Имеется несколько вариантов ограничений и для каждого из них выделен определенный идентификатор, записываемый в поле *тип_атрибута*. Значения атрибута нумерованного типа либо выбираются из конечного списка значений, записанного в поле *тип_атрибута*, либо представляют собой тексты предопределенного формата (например, форматов HTML, SGML и т.п.), на что указывает запись **NOTATION** *список_форматов*.

Статус может иметь четыре значения (варианта). Вариант **#REQUIRED** используется, если для соответствующих типов элементов в документе запись значения данного атрибута обязательна. Вариант **#IMPLIED** используется, если значение атрибута в документе использовать необязательно. Значение "*значение_по_умолчанию*", записываемое в поле *статус*, есть значение атрибута, используемое по умолчанию. В случае варианта **#FIXED** "*значение_по_умолчанию"* никакое другое значение атрибута, кроме указанного по умолчанию, в документе использовать нельзя.

Отметим, что в одной записи вида (6.2) может быть описано более одного атрибута.

Приведем несколько примеров объявлений атрибутов. Предварительно напомним, что размеченный текст в элементе "Документ" состоит из элементов, помещаемых в контейнеры, т.е. между парой тегов. Например, такими тегами могут быть **<item>** и **</item>, <termin>** и **</termin>, <description> и </description>, <examples>** и **</examples>**. Значения атрибутов могут включаться в открывающий тег, как это сделано для тега **<T1>** в следующем примере.

Пример 1. **<!ATTLIST T1 A1 CDATA #REQUIRED>**,

здесь элемент типа **T1** имеет атрибут **A1**, его значения относятся к типу данных **CDATA**, значение атрибута **A1** должно быть указано в открывающем теге каждого элемента типа **T1**, например

<T1 A1='high_level'> Программный комплекс CATIA </T1>

Пример 2. **<!ATTLIST book title CDATA #REQUIRED**

author CDATA #IMPLIED publisher CDATA #REQUIRED >,

Элемент типа **book** имеет три атрибута строкового типа, значения атрибута **author** указывать необязательно, два других атрибута обязательны.

Пример 3. **<!ATTLIST ray color (red|green|blue) "red">**

Атрибут **color** элемента типа **ray** может обозначать красный, зеленый или синий цвет, по умолчанию задается красный цвет.

Сущности в DTD используют для присвоения псевдонимов некоторым блокам данных, что позволяет в документе лаконично ссылаться на эти данные, на данные из ранее разработанных документов, на блок многократно используемых данных, включать в XML-документ данные различных форматов, например, графические данные.

Блок данных может быть внутренним примитивом, т.е. фразой, непосредственно записываемой в объявлении:

<!ENTITY *псевдоним* "*фраза*" **>**.

Например, с помощью **ENTITY** можно вводить аббревиатуры для словосочетаний, как в следующей строке:

<!ENTITY IEEE "Institute of Electrical and Electronics Engineers" >

В документе записи вида **&IEEE;** будут заменены на блок XML-данных **Institute of Electrical and Electronics Engineers**.

Блок XML-данных может быть внешним файлом, и тогда ссылка есть указание адреса файла:

<!ENTITY *псевдоним* **SYSTEM** "*адрес_файла*" **>** (6.3)

Например:

<!ENTITY Приложение SYSTEM "http://ifc.com/addition1.xml" >.

В документе записи вида **&Приложение;** будут заменены на XML-содержимое файла с указанным адресом.

Несколько сложнее выглядит включение в XML-документ внешних не XML-данных. В этом случае строка (6.3) имеет вид

<!ENTITY *псевдоним* **SYSTEM** "*адрес_файла*" **NDATA** *имя_нотации***>,**

где *имя_нотации* - описание формата не XML-данных или адрес программы для обработки этих данных.

Например, если нужно внутри элемента типа item разместить рисунок, данные о котором находятся в файле fig.bmp в формате BMP, то в DTD нужно добавить строки

<!ELEMENT place EMPTY>

<!ATTLIST place object ENTITY #REQUIRED>

<!NOTATION BMP SYSTEM "pbrush.exe">

<!ENTITY figure SYSTEM "fig.bmp" NDATA BMP >

Первые две строки объявляют пустой элемент **place** с атрибутом **object** маркерного типа, в котором используется идентификатор **ENTITY.** Назначение идентификатора **ENTITY** - ограничивать возможные значения атрибута значениями псевдонимов, введенных в DTD. Две следующие строки задают месторасположение графических данных (файл **fig.bmp**) и способ их обработки (программа **pbrush.exe**). В самом документе в контейнерах элементов типа **item** достаточно записать строки

<place object="figure" />,

чтобы при визуализации элементов типа **item** появилось изображение в соответствии с данными файла **fig.bmp**.

Блок данных, фигурирующий в DTD, может быть списком атрибутов, который используется в DTD многократно. Его целесообразно заменить псевдонимом, перед которым нужно записать символ **%**:

<!ENTITY % *имя перечня* '*список атрибутов*'>,

где *список атрибутов* записывается по тем же правилам, что и в объявлении **ATTLIST**.

Основные рассмотренные свойства XML-документов поясним следующим примером. Пусть исходный неразмеченный документ представляет собой фрагмент словаря, состоящий из трех пунктов (в нашем примере названия пунктов CALS, Ethernet, PDM). Каждый пункт относится к одному из понятий определенной предметной области и включает название понятия, его краткое определение и возможно некоторые поясняющие примеры.

Целесообразно использовать иерархическую структуру документа: верхний уровень относится к пунктам словаря, нижний уровень относится к элементам пункта. Принятая структура отражается в DTD.

После разметки исходного текста получаем XML-документ следующего вида:

<?xml version="1.0" ?>

<?xml-stylesheet type="text/css" href="dict.css"?>

```
<!DOCTYPE dictionary [
   <!ELEMENT dictionary (item) >
   <!ELEMENT item (termin,description,examples?) >
   <!ELEMENT termin (#PCDATA)>
     <!ATTLIST termin number ID #REQUIRED >
     <!ATTLIST termin group
      (technology|networks|software|other) #REQUIRED >
   <!ELEMENT description (#PCDATA)>
   <!ELEMENT examples (#PCDATA)>
   <!ENTITY ЛВС "локальная вычислительная сеть" >
]>
<dictionary >
<item>
  <termin number ='_14' group='technology'> CALS </termin>
  <description> - Continuous Acquisition and Lifecycle Support, информационное
сопровождение и поддержка этапов жизненного цикла промышленных
изделий. Технология взаимодействия различных автоматизированных
систем в промышленности.
  </description>
</item>
<item>
  <termin number ='_24' group='networks'> Ethernet </termin>
  <description> - &ЛВС; с методом доступа МДКН/ОК.
  </description>
  <examples>  Варианты реализации 10Base-5, 10 Base-T,
    100Base-X. Gigabit Ethernet.
  </examples>
</item>
<item>
  <termin number ='_52' group= 'technology'> PDM </termin>
  <description> - Product Data Management, управление проектными данными.
Системы PDM, называемые также системными средами, входят в состав
программного обеспечения CALS-технологий.
  </description>
  <examples> Windchill eSeries,iMAN, SmartTeam, Optegra.
  </examples>
</item>
</dictionary>
```

Нужно сделать ряд дополнительных пояснений особенностей языка XML.

Прежде всего рассмотрим, как в объявлении (1) оформляется *содержание* в виде списка типов элементов нижнего уровня, вложенных в элемент верхнего уровня.

Во-первых, этот список может быть представлен перечислением типов элементов нижнего уровня в круглых скобках через запятую:

<!ELEMENT item (termin,description,examples) > (4)

В этом случае вложенные типы **termin, description, examples** должны обязательно присутствовать в каждом элементе верхнего уровня типа **item**, причем по одному разу и в том же порядке, как они представлены в (4).

Во-вторых, элементы списка можно отделять не запятыми, а символом "вертикальная черта":

<!ELEMENT item (termin|description|examples) >

Теперь в элементе типа **item** может присутствовать единственный вложенный элемент одного из типов **termin, description, examples**.

В обоих рассмотренных случаях можно варьировать характером ограничений, используя после элементов списка (или после скобки, закрывающей часть или весь список) соответствующие знаки +, * и ?. Знак + означает, что элемент соответствующего типа (или группа элементов, типы которых перечислены в списке, заключенном в круглые скобки) может присутствовать в элементе верхнего уровня один или более раз. Знак * отличается от знака + тем, что указываемый им тип элемента может отсутствовать в элементе верхнего уровня. Знак ? отличается от знака * тем, что позволяет использовать вложенный элемент не более одного раза. Так, в нашем примере благодаря знаку ? после элемента типа **examples,** можно не во всех пунктах словаря приводить примеры, что и сделано для пункта с атрибутом **number ='_14'.** Если бы нам требовалось разрешить произвольные число и порядок следования вложенных элементов, то нужно было бы использовать описание элемента типа **item** в виде

<!ELEMENT item (termin|description|examples)+ >

В приведенном примере XML-документа атрибут **number** относится к маркерному типу. Его идентификатор **ID** означает, что этот атрибут должен иметь уникальные значения для каждого элемента **termin**, т.е. **number** является ключевым атрибутом (значения типа **ID** не должны начинаться с цифры, поэтому в примере используется знак подчеркивания).

Часто возникает необходимость включения в XML-документ символов, отсутствующих на клавиатуре компьютера (например, буквы греческого алфавита) или на символы, относящиеся к служебным символам языка. На них следует ссылаться с помощью записи

&#код_символа_ по_ ISO/IEC10646.

Кроме того, для символов &, <, >, ', " можно использовать ссылки &amp, &lt, &gt, &apos, &quot соответственно.

В нашем примере XML-документа **dictionary** используются каскадные таблицы стилей. Подробное описание этих таблиц имеется в литературе. Здесь поясним только те свойства CSS, которые нужны для нашего примера. Пусть мы хотим элементы типа **termin** выделить полужирным шрифтом (**bold**) 12-го размера с отступом первой строки на 5 мм, а элементы типа **examples** - курсивом (**italic**) 10-го размера с отступом на 10 мм. Тогда таблица CSS, помещаемая в файл **dict.css**, должна быть задана в виде:

```
item
  {display:block;}
termin
  {font-weight:bold; font-size:12pt; text-indent:5mm; font-style:normal;}
description
  {font-size:12pt;}
examples
  {display:block; font-style:italic; font-size:10pt;
   text-indent:10mm;}
```

Обращение к браузеру для просмотра нашего документа позволит увидеть текст, представленный в табл. 2.

Таблица 2

CALS - Continuous Acquisition and Lifecycle Support, информационное сопровождение и поддержка этапов жизненного цикла промышленных изделий. Технология взаимодействия различных автоматизированных систем в промышленности.

Ethernet - локальная вычислительная сеть с методом доступа МДКН/ОК.
Варианты реализации 10Base-5, 10Base-T, 100Base-X. Gigabit Ethernet.

PDM - Product Data Management, управление проектными данными. Системы PDM, называемые также системными средами входят в состав программного обеспечения CALS-технологий.
Windchill eSeries, iMAN, SmartTeam, Optegra.

Более богатые возможности форматирования предоставляет специально разработанный для XML язык форматирования XSL (eXtensible Stylesheet Language).

Указания по форматированию, выраженные средствами языка XSL, составляют XSL-таблицу, с помощью которой XML-документ преобразуется в HTML-страницу, отображаемую браузером. Использование XSL обеспечивает ряд преимуществ по сравнению с применением CSS, поскольку появляется возможность сортировать и фильтровать элементы документа при его выводе на экран. Кроме того, изменяя XSL-таблицу, можно один и тот же документ изображать по-разному в соответствии с потребностями конкретной ситуации.

В XSL шаблоны, по которым браузер определяет отображение элементов документа на экране, обрамляются выделенными для этого тегами, например, **<xsl:template>** и **</xsl:template>.** Шаблоны содержат правила, в которых указываются типы XML-элементов, к которым правило относится, и задаются инструкции отображения, например, аналогичные принятым в языке HTML. Шаблон может относиться ко всему XML-документу или к его части. В первом случае в теге **<xsl:template>** указывается атрибут **match** со значением **"/"**, т.е.

<xsl:template match="/">,

во втором случае значением атрибута **match** будет имя типа соответствующего XML-элемента.

В шаблонах можно использовать как HTML-элементы, так и XSL-элементы. Последние имеют вид

<xsl:*имя_xsl-элемента имя_параметра*="*значение_параметра*"/>.

Например, значениями параметра с именем **select** могут быть типы отображаемых XML-элементов. В качестве имен XSL-элементов используются **value-of** (выбор для отображения текущего XML-элемента, тип которого указан в параметре **select**), **for-each** (команда отображения всех XML-элементов, тип которых указан в параметре **select**) и некоторые другие. Нужно отметить, что для сортировки XML-элементов используется параметр **order-by**, указываемый в XSL-теге **<xsl:for-each>**.

Таблица стилей помещается в отдельный файл, ссылка на который обычно включается в заголовок XML-документа и имеет вид:

<?xml-stylesheet type="text/xsl" href=*путь к файлу***?>**

В XML расширены возможности гиперсвязей. Механизм связей в XML изложен в спецификациях XLink и XPointer. Первая из них посвящена связям между документами, а вторая - связям внутри документа. Простая связь аналогична href-ссылке в HTML. Но возможны также связи многонаправленные,

связи, которые не заменяют на экране просматриваемый документ данными из вызванного ресурса, а помещают эти данные в дополнительное окно или встраивают их в исходный документ. За счет введения промежуточной БД ссылок удается сохранять связи при изменении адресов страниц.

Программная поддержка языка XML обеспечивается XML-процессорами. В состав XML-процессора входит синтаксический анализатор, который проверяет правильность соблюдения правил языка, но не производит форматирования. Для форматирования документа используется другая компонента XML-процессора, поддерживающая каскадные таблицы стилей или язык форматирования XSL.

К функциям программного обеспечения, поддерживающего XML, кроме синтаксического анализа и визуализации, относятся поиск заданных фрагментов, создание, удаление, модификация элементов в XML-документе. Для поддержки этих функций в рабочей группе W3C, занимающейся вопросами Web-технологий, разрабатывается объектная модель HTML и XML-документов DOM (Document Object Model), предназначенная для создания прикладного интерфейса API (Application Program Interface) к XML-документам, и соответствующий язык запросов.

Спецификация DOM вводит модель XML-документа DOM в виде иерархии его элементов и язык запросов XPath (XML Path Language), позволяющий ссылаться на части XML-документов. Прикладной интерфейс на основе DOM позволяет прикладным программам обращаться к документам, извлекать, добавлять, изменять или удалять отдельные элементы или атрибуты. Спецификация DOM создана для использования практически с любым языком программирования.

XPath обеспечивает синтаксис и семантику для запросов и ссылок на содержимое XML-документов. Если язык SQL служит для обращений к содержимому реляционных баз данных, то язык XPath предназначен для обращений к содержимому баз XML-документов. Выражения XPath представляют собой указание пути к узлу XML-документа в иерархической DOM-модели этого документа. Так, запрос "найти элементы 'termin' " при обращении к словарю приведенного выше примера с помощью XPath-выражения

dictionary/item/termin

позволит получить список всех терминов словаря, а запрос

dictionary/item/termin[@group='networks']

список только тех терминов, у которых атрибут **group** равен **'networks'.**

Предложены и другие языки обращений к XML-документам, например гибкий язык запросов XQuery или язык XSLT (XSL Transformations). В языке XQuery запросы представляют собой последовательность выражений, задающих возвращаемые узлы, которыми могут быть элементы и атрибуты XML-документов. Язык XSLT - подмножество XSL, предназначенное для преобразования одних XML-документов в другие документы XML, HTML или документы некоторых других форматов. Как XQuery, так и XSLT используют правила языка XPath. Результатом распространения DOM на мультимедийные данные является язык SMIL - Synchronized Multimedia Integration Language.

XML-схемы и пространства имен

Альтернативой DTD является модель XML-документа, описывающая его структуру, представленная на языке XML и называемая XML-схемой.

Прежде чем знакомиться с особенностями XML-схем, нужно рассмотреть понятие пространства имен, используемое в Web-технологиях. В общем случае пространством имен называют множество имен, относящихся к определенному приложению и зафиксированное в некотором документе. Это понятие вводится с целью исключения конфликта имен, возникающего при совместном использовании документов из разных приложений в случае, если в этих приложениях введены одинаковые названия для разных объектов. Чтобы избежать конфликтов, требуется сделать имена разных объектов гарантированно различными. Для этого применяют составные имена (QNames), состоящие из идентификатора приложения (префикса) и собственно локального имени объекта. В качестве идентификатора приложения рекомендуется использовать ссылку на документ, задающий пространство имен.

Для XML-документов пространство имен — это идентифицируемое с помощью ссылки URI множество имен, используемых в XML-документах для обозначения типов элементов и именования атрибутов.

Ссылка есть URI документа, чаще всего это URL. Вместо громоздкого URI можно использовать его псевдоним. Тогда ссылка имеет вид:

псевдоним:локальное_имя

Например: rdfs:class, где rdfs есть псевдоним. Псевдоним связывается с URL пространства имен следующим образом:

<x xmlns:rdfs="http://www.w3.org/2000/01/">

т.е. для элемента "x" и его содержимого псевдоним "rdfs" заменяет ссылку

http://www.w3.org/2000/01/

Примеры некоторых часто используемых префиксов и псевдонимов:

для XML-схем псевдоним xsd: заменяет ссылку на URI http://www.w3.org/2001/XMLSchema#

для модели RDF псевдоним rdf: заменяет ссылку на URI: http://www.w3.org/1999/02/22-rdf-syntax-ns#

для языка онтологий OWL псевдоним owl: заменяет ссылку на URI: http://www.w3.org/2002/07/owl#

Вернемся к описанию XML-схем. XML-схема, как и DTD, предназначена для описаний структурных и семантических ограничений, которые должны выполняться в любом экземпляре данных, соответствующем этой модели. Характерным примером структурного ограничения для XML-документов является спецификация содержания элементов (например, элемент с именем A может содержать только элементы с именем B), а примером семантического ограничения - указание, что некоторый атрибут элемента является ключом.

В XML-схемах используют определения и объявления. С помощью определений задают новые типы элементов, а с помощью объявлений - имена и содержимое элементов и атрибутов.

Различают комплексные типы элементов, которые могут иметь вложенные элементы и атрибуты, и простые типы, которые таковых не имеют.

Комплексные типы задаются с помощью определения complexType. В них обычно содержится набор из объявлений элементов, ссылок на элементы и объявлений атрибутов. Элементы задаются с помощью объявления element, а атрибуты – с помощью объявления attribute. Объявления обычно связывают имена элементов и ограничения. Атрибут может быть объявлен с параметром

use, значениями use могут быть use="required" (обязателен), use="optional" (необязателен), или use="prohibited" (запрещен). Атрибут fixed используется для указания, что атрибут или элемент может принимать только фиксированное значение. Атрибут ref является ссылкой на ранее объявленный элемент, например < xsd:element ref="item"/>.

Простые типы определяются в данном документе или являются встроенными в язык XML-схемы. К встроенным относятся string, integer, decimal, float, date (например, запись даты в виде 2005-06-26), language, ID и ряд других. Можно накладывать ограничения на число вхождений элементов в комплексный тип с помощью атрибутов minOccurs и maxOccurs, например, minOccurs='0' означает необязательность вхождения элемента в документ (аналогично указанию ? в DTD), а maxOccurs='2' говорит о том, что элемент может входить один или два раза.

На рис. 12 представлен пример модели документа в виде, соответствующем спецификации DOM (Document Jbject Model.

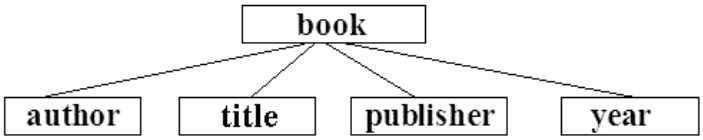

Рис. 6.12. Пример модели документа

Та же модель в виде XML-схемы (с добавлением атрибута price для элемента book) имеет вид:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
. . .
    <xsd:element name="book">
    <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="author" type="xsd:string"/>
       <xsd:element name="title" type="xsd:string"/>
       <xsd:element name="publisher" type="xsd:string"/>
       <xsd:element name="year" type="xsd:integer"/>
     </xsd:sequence>
     <xsd:attribute name="price" type="xsd:decimal"/>
    </xsd:complexType>
   </xsd:element>
. . .
</xsd:schema>
```

В самом XML-документе требуется ссылка на XML-схему описывающую его структуру. Например, ссылка на XML-схему, размещенную в файле по адресу http://www.bibl.ru/Books.xsd", в XML-документе Reference может быть выполнена с помощью указания используемого пространства имен (http://www.bibl.ru/Books) и атрибута SchemaLocation следующим образом:

```
<Reference
  xmlns="http://www.bibl.ru/Books"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bibl.ru/Books
  http://www.bibl.ru/Books.xsd">
 . . .
```

. . .
</Reference>

Средства отображения XML-документов и создания интерактивных Web-страниц

Программным средством визуализации статических Web-страниц, представленных на языке HTML, является браузер. Однако в языке XML множество тегов заранее не определено. Поэтому для просмотра XML-документа с помощью браузера осуществляется предварительная привязка XML-документа к HTML-странице. Например, привязка становится возможной, если в число тегов HTML ввести теги, идентифицирующие XML-элементы и сцепляющие их с HTML-элементами.

Для идентификации в браузере Internet Explorer пятой версии (v5) используется тег <**XML**>. Например:

<XML ID="*имя для доступа к XML-документу с HTML-страницы***"**
SRC="*имя файла с XML-документом***"> </XML>.**

Сцепление осуществляется с помощью некоторых HTML-тегов, например тега **<SPAN>**:

<SPAN DATASRC="#*имя для доступа к XML-документу с HTML-страницы***" DATAFLD="***элемент XML-документа***"> </SPAN>.**

Теперь "*элемент XML-документа*" сцеплен с тегом **<SPAN>** и может быть отображен с помощью записи строки

<SPAN *необходимые параметры стиля***>** *элемент XML-документа*
</SPAN>

Более привлекательный способ отображения XML-документа на HTML-странице связан с использованием языка XSL и соответствующей XSL-таблицы стилей.

В динамических Web-страницах визуализация меняющихся изображений, реакция системы на определенные действия пользователя или иные операции по обработке информации, сопровождающие просмотр документов, выполняются по сценариям, задаваемым на языке программирования. Как отмечено выше, применяемые при этом технологии различаются как по типу используемого языка, так и по месту исполнения программ, реализующих сценарии.

Сравнительно простые сценарии обычно представляют на языках типа JavaScript, VBScript или их разновидностях. Команды сценария непосредственно включаются в HTML-документ с помощью следующего фрагмента

<SCRIPT LANGUAGE = "javascript">
<!- - *сценарий* **//- ->**
</SCRIPT>

где **<!- -** *сценарий* **//- ->** - текст сценария, представленный в виде комментария. Браузеры, не имеющие JavaScript-обработчиков, просто игнорируют комментарий, а современные браузеры исполняют записанные в сценарии команды. Характерной особенностью применения Javascript является интерпретация и исполнение команд сценария на клиентском узле.

Более сложные сценарии требуют использования прикладных программ, написанных на языках программирования Java, C, C++ и т.п. Если прикладная программа специально создается для использования в Web-среде, целесообразно использовать язык Java.

Обращение к программам, представленным на языке Java, возможно из браузера с помощью специального элемента **<APPLET>** или заменяющего его элемента **<OBJECT>**, например:

<APPLET CODE="*имя файла аплета***">**

элементы <PARAM>, задающие параметры, передаваемые в аплет

</APPLET>.

В сценарии можно предусмотреть обращения к многим аплетам, находящимся в разных узлах сети.

В ряде случаев используемые прикладные программы не являются платформно независимыми и по этой или иным причинам должны исполняться на сервере. В этих случаях используются технологии типа CGI, ISAPI и т.п.

Обычно посредник CGI находится на сервере в специальном каталоге CGI_BIN, он является обработчиком запросов, идущих от браузера. Обращение к файлу из этого каталога, например по команде

<A HREF="*url сценария***?***передаваемые параметры***">**

ссылочная фраза

</A>,

означает запуск соответствующего обработчика. Если браузер обращается к документу не в HTML-формате, то посредник преобразует форму документа в HTML-формат и возвращает его браузеру.

Для внедрения в Web-страницу мультимедийной информации используют элемент **<OBJECT>**. В элементе используют параметры, указывающие адрес мультимедийного файла и расположение поля вывода мультимедийной информации на экране дисплея. Средства обработки мультимедийной информации, представленной в файлах большинства известных форматов таких, как AVI, MOV, MPG, MP3, WAV, MID, обычно встроены в браузеры.

5. Системные среды автоматизированных систем

Назначение системных сред

Системы автоматизированного проектирования относятся к числу наиболее сложных и наукоемких АС. Проектные данные используются в разных подсистемах САПР. Они отличаются разнообразием форм представления, характеризуют как текущее состояние проектов, так и их предыдущие версии. Наряду с выполнением собственно проектных процедур необходимо автоматизировать также управление проектированием, поскольку сам процесс проектирования становится все более сложным и зачастую приобретает распределенный характер. Большой объем данных, необходимость поддержания их целостности (достоверности и полноты), сложность управления проектированием привели к созданию в составе современных САПР специального ПО - системной среды САПР, называемой системой управления проектными данными PDM (Product Data Management).

На крупных и средних предприятиях заметна тенденция к интеграции САПР с АСУП и СДО. Для управления столь сложными интегрированными системами, для обеспечения целостности данных и доступа к ним на любом этапе жизненного цикла изделий создаются системы управления жизненным циклом изделий PLM (Product Lifecycle Management).

История систем управления проектными данными - систем PDM - непосредственно связана с развитием систем автоматизированного проектирования. Появление системных сред в САПР ознаменовало переход от использования отдельных не связанных друг с другом программ, решающих частные проектные задачи, к применению интегрированной совокупности таких программ.

Интегрирующим компонентом в 70-е гг. стала единая БД САПР. Однако попытки использовать имевшиеся в то время СУБД не приводили к удовлетворительным результатам в силу разнообразия типов проектных данных, распределенного и параллельного характера процессов проектирования, с одной стороны, и недостаточной развитости баз данных, с другой стороны.

Специализированные СУБД, ориентированные на САПР, были созданы в 80-е годы. Однако они не учитывали или в недостаточной степени удовлетворяли требованиям обеспечения целостности данных, управления потоками проектных работ, многоаспектного доступа пользователей к данным.

И лишь на рубеже 80-90 гг. появились системы управления проектными данными, названные в то время Framework или системными средами, сначала в САПР электронной промышленности, а позднее и в САПР машиностроения, где они и получили наименование PDM.

На протяжении 90-х гг. роль системных сред неуклонно повышалась. Во-первых, из-за роста сложности проектируемых объектов и необходимости сокращать сроки проектирования. Во-вторых, из-за необходимости интеграции систем проектирования с системами управления предприятием и технологическими процессами. Благодаря развитию Internet, Web- и CALS-технологий такая интеграция стала возможной в глобальном масштабе. Функции PDM стали распространяться на разные этапы жизненного цикла изделий (ЖЦИ). В результате в конце 90-х годов системы управления данными на протяжении всего ЖЦИ стали называть системами PLM.

В настоящее время термином PLM чаще всего обозначают современную концепцию (методологию) информатизации ЖЦИ, а под средствами PLM понимают PDM вместе со средствами интеграции АС, часто включая в PLM также средства самих АС проектирования и управления.

Современные системы PDM предназначены для информационного обеспечения проектирования и выполняют следующие основные функции:

- хранение проектных данных и доступ к ним, в том числе ведение распределенных архивов документов, их поиск, редактирование, маршрутизация, создание спецификаций;
- структурирование и визуализпация данных;
- управление конфигурацией изделия, т.е. ведение версий проекта, управление внесением изменений;
- управление проектированием (проектами);
- защита информации;
- интеграция данных (поддержка типовых форматов, конвертирование данных).

Хранение данных - функция *банка данных* (БнД). Банк данных состоит из СУБД и баз данных (БД). Обычно системы PDM интегрируются с независимо разработанными СУБД. Информационный обмен между программами САПР в значительной мере также реализуется с помощью банка данных.

PDM отличает легкость доступа к иерархически организованным данным, обслуживание запросов, выдача ответов не только в текстовой, но и в

графической форме, привязанной к конструкции изделия. Поскольку взаимодействие внутри группы проектировщиков в основном осуществляется через обмен данными, то в системе PDM часто совмещают функции управления данными и управления параллельным проектированием.

Системы управления базами данных

В большинстве автоматизированных информационных систем применяют СУБД, поддерживающие реляционные модели данных.

Среди общих требований к СУБД можно отметить: 1) обеспечение целостности данных (их полноты и достоверности); 2) защита данных от несанкционированного доступа и от искажений из-за сбоев аппаратуры; 3) удобство пользовательского интерфейса; 4) в большинстве случаев важна возможность распределенной обработки в сетях ЭВМ.

Первые два требования обеспечиваются ограничением прав доступа, запрещением одновременного использования одних и тех же обрабатываемых данных (при возможности их модификации), введением контрольных точек (checkpoints) для защиты от сбоев и т.п.

Банк данных в САПР является важной обслуживающей подсистемой, он выполняет функции информационного обеспечения и имеет ряд особенностей. В нем хранятся как редко изменяемые данные (архивы, справочные данные, типовые проектные решения), так и сведения о текущем состоянии различных версий выполняемых проектов. Как правило, БнД работает в многопользовательском режиме, с его помощью осуществляется информационный интерфейс (взаимодействие) различных подсистем САПР. Построение БнД САПР - сложная задача, что обусловлено следующими особенностями САПР:

1. Разнообразие проектных данных, фигурирующих в процессах обмена как по своей семантике (многоаспектность), так и по формам представления. В частности, значительна доля графических данных.

2. Нередко обмены должны производиться с высокой частотой, что предъявляет жесткие требования к быстродействию средств обмена (полагают, что СУБД должна работать со скоростью обработки тысяч сущностей в секунду).

3. В САПР проблема целостности данных оказывается более трудной для решения, чем в большинстве других систем, поскольку проектирование является процессом взаимодействия многих проектировщиков, которые не только считывают данные, но и изменяют их, причем в значительной мере работают параллельно. Из этого факта вытекают следствия: во-первых, итерационный характер проектирования обычно приводит к наличию по каждой части проекта нескольких версий, любая из них может быть принята в дальнейшем в качестве основной, поэтому нужно хранить все версии с возможностью возврата к любой из них; во-вторых, нельзя допускать использования неутвержденных данных, поэтому проектировщики должны иметь свое рабочее пространство в памяти и работать в нем автономно, а моменты внесения изменений в общую БД должны быть согласованными и не порождать для других пользователей неопределенности данных.

4. Транзакции могут быть длительными и трудоемкими. *Транзакцией* называют последовательность операций по удовлетворению запроса. В САПР внесение изменений в некоторую часть проекта может вызвать довольно длинную и разветвленную сеть изменений в других его частях из-за существенной

взаимозависимости компонентов проекта (многошаговость реализации запросов). В частности, транзакции могут включать в себя такие трудоемкие операции, как верификация проектного решения с помощью математического моделирования. В результате транзакции могут длиться весьма продолжительное время. Одна из трудностей заключается в отображении взаимозависимости (ассоциативности) данных. При хранении компонентов проекта во внешней памяти затраты времени на обработку запросов оказываются значительно выше, чем в большинстве других автоматизированных систем, с менее выраженными взаимозависимостями данных.

5. Иерархическая структура проектных данных и, следовательно, отражение наследования в целях сокращения объема базы данных.

В определенной мере названные особенности учитываются в СУБД третьего поколения, в которых стали применяться черты объектно-ориентированных (объектных) СУБД. В них наборы данных, характеризующих состояние предметной области (состояние проекта в случае САПР), помещаются в отдельные файлы. Интерпретация семантики данных осуществляется с помощью специальных процедур (методов), сопровождающих наборы. Наследование свойств объектов предметной области выражается с помощью введения категорий класса, надкласса, подкласса. Информационные модели приложений для таких СУБД разрабатываются на основе методик с использованием языка UML.

Объектные БД выгодны, во-первых, тем, что данные по конкретным объектам проектирования не разбросаны по множеству таблиц, как это имеет место в реляционных БД, а сосредоточены в определенных местах. Во-вторых, для каждого объекта могут быть назначены свои типы данных. В результате проще решаются задачи управления и удовлетворения запросов.

Наряду с чисто объектными СУБД применяют СУБД объектно-реляционные. В последних происходит объединение свойств реляционных и объектно-ориентированных СУБД: объектно-ориентированная СУБД снабжается непроцедурным языком запросов или в реляционную СУБД вводятся наследование свойств и классы. Непроцедурность входного языка обеспечивается использованием языка SQL. Его операторы непосредственно включаются в программы на языке С. Возможно написание дополнительных программ, интерпретирующих SQL-запросы.

Отличительные особенности СУБД третьего поколения: расширенный набор возможных типов данных (это абстрактные типы, массивы, множества, записи, композиции разных типов, отображение величин с значениями разных типов), открытость (доступность из разных языков программирования, возможность обращения к прикладным системам из СУБД), непроцедурность языка (общепринятым становится язык запросов SQL), управление асинхронными параллельными процессами, состояние которых отражает БД.

Рассмотренные особенности банков данных в САПР позволяют квалифицировать их как системы Data Warehouse (DW), т.е. хранилища данных. Для хранилищ данных характерен ряд особенностей, совпадающих с названными выше особенностями банков данных САПР: 1) длительное хранение информации, отражающей историю разработок; 2) частота операций чтения данных выше частоты операций обновления данных; 3) использование единых форматов для однотипных данных, полученных из различных источников (например, от разных программно-методических комплексов). Эти особенности позволяют управлять конфигурацией проектов, что, в частности, означает хранение в САПР всех версий проекта и, возможно, данных по проектам

предыдущих разработок, удовлетворение сложных запросов, для ответа на которые требуется извлечение и обработка данных из различных частей хранилища (так называемая многомерная обработка). Модели данных в DW 痕гличаются от реляционных моделей (RM): в RM использованием нормальных форм стремятся максимально уменьшить избыточность данных, что приводит к увеличению числа таблиц, но уменьшенных размеров, при этом многомерный поиск в множестве таблиц затруднен. Поэтому в DW чаще используется модель данных "звезда", в которой имеется общая таблица фактов (Fact Table) и каждому факту ставится в соответствие несколько таблиц с необходимыми атрибутами. Целостность данных в DW обеспечивается проверкой и трансформацией данных, вводимых из внешних источников, наличием дисциплины обновления данных, централизованным хранением основной базы, при этом достаточное быстродействие поддерживается передачей копий определенных частей базы в локальные базы, называемые киосками данных (Data Mart) и ориентированные на отдельные группы пользователей.

Интеллектуальные средства поддержки принятия решений

В общем случае полная формализация управления проектированием не может быть достигнута, поэтому полезную роль играют СППР - системы поддержки решений, принимаемых людьми (обозначаемые также DSS - Decision Support Systems). Такие системы часто используются совместно с хранилищами данных и OLAP-системами (On-Line Analytical Processing).

OLAP-системы должны обеспечивать оперативный доступ к данным, на основе которого выявляются зависимости между параметрами (измерениями в многомерной модели приложения). В OLAP-системах на реляционных СУБД аналитическая обработка, или, другими словами, многомерный динамический анализ данных, требует просмотра большого числа записей из разных таблиц. Поэтому производительность оказывается невысокой. В специализированных OLAP-системах, обеспечивающих более быстрый многомерный анализ, но с более существенными ограничениями на объем БД, данные хранятся в виде гиперкубов или поликубов - многомерных таблиц с постоянным или переменным числом ячеек соответственно. Пример OLAP системы - Oracle Express, помогающей менеджерам и аналитикам получать данные в виде разрезов таких многомерных таблиц, готовить отчеты, обосновывать решения.

В составе подсистем управления методологией проектирования полезно иметь средства консультирования по принятию проектных решений. Они могут быть представлены в виде множества модулей, объединяемых гипертекстовой оболочкой. Каждый модуль содержит некоторый совет по выбору решения, преодолению противоречий, возникающих в процессе проектирования. Здесь уместно использование методов и приемов решения изобретательских задач.

Интеграция программного обеспечения в САПР

Сложные информационные системы обычно представляют собой некоторое объединение более простых подсистем, созданных независимо друг от друга и возможно в разное время. Очевидно, что такие подсистемы могут быть

ориентированными на разные платформы - использовать разные СУБД, языки программирования и т.п., а иногда даже исполняться в разной операционной среде. Поэтому интеграция подсистем, первоначально являвшихся самостоятельными независимыми системами, - одна из наиболее актуальных и сложных задач в области информационных технологий.

Интеграция возможна на одном из двух подходов:

- на основе единой БД; примером может служить система с трехзвенной арихитектурой, в которой имеется несколько прикладных подсистем и сервер баз данных на основе одной СУБД;
- на основе распределенной БД и/или межси□темного обмена сообщениями.

Модификации систем с единой БД вызывают затруднения, так как дополнение системы новыми подсистемами может потребовать существенной перестройки БД. Вследствие этого, подход на основе единой БД имеет заметные ограничения по сложности интегрируемой системы и возможностям ее развития. Поэтому для интеграции сложных промышленных автоматизированных систем преимущественно используют второй подход. В системах с распределенной БД модификации осуществляются проще, но появляется проблема синхронизации данных.

В свою очередь, системы с распределенной БД и обменом сообщениями различают по способам построения интегрирующей среды.

В наиболее простом варианте, называемом "точка-точка", взаимодействие подсистем осуществляется по схеме полного графа, т.е. для каждой пары взаимодействующих подсистем создается специфическая для них интерфейсная связь в виде конверторов данных с языка одной подсистемы на язык другой. Поскольку число таких дуплексных связей может доходить до $N(N-1)/2$, где N - число подсистем, то вариант "точка-точка" оказывается приемлемым только для малых N. Подключение к системе каждой новой подсистемы оказывается весьма трудоемким.

Число связей уменьшается до $N+1$ в варианте интеграции на основе общего для подсистем языка, поддерживаемого промежуточной метасредой. Теперь достаточно в каждой подсистеме иметь конвертор только на промежуточный язык. Примерами таких языков могут служить XML, Express в стандарте STEP или SQL в технологии ODBC. Управление потоком данных осуществляется на основе явного указания в сообщении адреса подсистемы - партнера по взаимодействию.

Третий вариант, называемый "корпоративная шина" или "интеграционный сервер", характеризуется наличием не только конвертации языков, но и более сложным управлением потоками данных. В интеграционном сервере возможна та или иная обработка данных, например, определение подсистемы-партнера может быть результатом анализа содержательной части сообщений. Примерами таких серверов являются мониторы транзакций, брокер ORB в технологии CORBA, корпоративная шина ESB и др.

Развитие варианта интеграционного сервера порождает так называемые сервис-ориентированные архитектуры систем. Сервис-ориентированная архитектура SOA (Service Oriented Architecture) – это архитектура программной системы, обеспечивающей динамическое обнаружение и вызов сервисов на основе независящего от платформы интерфейса. Обычно системы с SOA строятся на основе использования XML-интерфейса, протоколов HTTP, SOAP,

спецификаций UDDI, языка WSDL. Сервис-ориентированная архитектура тесно связана с шиной ESB, которая является способом реализации SOA.

Корпоративная сервисная шина ESB (Enterprise Service Bus) - программное обеспечение промежуточного слоя, используемое для передачи данных между приложениями, поддерживающее Web-сервисы на основе протокола SOAP, языка WSDL, спецификации UDDI, а также такие сервисы, как обработка и проверка сообщений, маршрутизация, балансирование нагрузки и др.

Ключом для решения проблемы интеграции автоматизированных систем является наличие языка, ориентированного на описание транзакций, имеющих место при выполнении бизнес-процессов. Примером такого языка может служить Business Process Execution Language (BPEL) - язык описания бизнес-процессов, выполняемых с помощью Web-сервисов, т.е. язык описания, во-первых, динамики событий, во-вторых, протоколов взаимодействия приложений.

BPEL - это язык со свойствами декларативного и процедурного программированя. В частности, в языке определены такие операции, как организация циклов, ветвление, параллельное выполнение, ожидание, генерация ответа и др. В основе BPEL лежит XML. Программа на BPEL интерпретируется во время исполнения, при этом выявляются ключевые слова и выполняется соответствующая обработка.

Описание бизнес-процессов, состоящих из обращений к Web-сервисам, возможно на разных языках программирования. Однако именно BPEL является проблемно-ориентированным языком, удобным для описания асинхронных вызовов различных Web-сервисов, координации параллельных бизнес-процессов, управления транзакциями и т.п.

Процесс, определенный в BPEL, содержит секцию деклараций, определяющую используемые Web-сервисы, и секцию действий, включенных в элемент <process>. К числу действий относятся <invoke> (вызвать), <receive> (получить), <reply> (ответить), <wait> (ждать), <terminate> (завершить), <assign> (назначить), <while> (цикл), <case> (выбор), <flow> (параллельное выполнение) и др.

При построении сложных систем из независимо созданных подсистем существенное значение имеет также семантический аспект интеграции.

Семантическая интеграция подразумевает автоматическое распознавание разными системами смысла передаваемых между ними данных. Теоретической базой для обеспечения семантической интеграции ПО в САПР являются:

1) методология автоматизированного проектирования, в соответствии с которой осуществляются типизация проектных процедур и маршрутов проектирования в различных предметных областях, выявление типичных входных и выходных данных процедур, построение информационных моделей приложений и их обобщение, сравнительный анализ альтернативных методов и алгоритмов выполнения типовых процедур;

2) онтологии приложений и их реализация в семантических языках и программах, примерами которых могут служить языки семантического Web, используемые в Internet для разработки интегрированных баз знаний;

3) объектно-ориентированная методология, в соответствии с которой множества сущностей, фигурирующих в процессах проектирования, подразделяются на классы, в классах появляются свои процедуры и типы данных с отношениями наследования. Эти классы могут быть инвариантными и прикладными. Их обобщение и унификация приводят к появлению таких понятий и средств, как интегрированные ресурсы и прикладные протоколы,

фигурирующие в стандартах STEP, или унифицированные программные компоненты типа графических ядер конструкторских САПР. Именно наличие типовых процедур и единообразное толкование атрибутов объектов в рамках разрабатываемых онтологий приложений и конкретных протоколов позволяют разным программным системам "понимать" друг друга при взаимодействии.

Наряду с типовыми графическими ядрами, известны типовые ПМК имитационного моделирования, конструирования деталей и механизмов, технологической подготовки производства и др. Возможность использования типовых программ в составе программных комплексов обусловлена именно унификацией интерфейсов при обменах данными.

В некоторых маршрутах проектирования обмены данными должны происходить с высокой частотой, что обусловливает специфические требования к интерфейсам. Примером могут служить задачи имитационного моделирования, в которых требуется имитировать взаимодействие процессов, описываемых с помощью различного МО (например, на сосредоточенном и распределенном иерархических уровнях, или с помощью аналоговых и дискретных моделей). Для таких задач при моделировании характерно воспроизведение временной последовательности событий, происходящих в анализируемых взаимодействующих системах. Соответственно взаимодействие программ моделирования может происходить через фиксированное число временных шагов или по мере совершения тех или иных событий в моделируемых системах.

Так, в программах смешанного аналого-дискретного моделирования электронных устройств аналоговая часть моделируется с помощью программы анализа электронных схем, а дискретная часть - с помощью программы логического моделирования. Влияние аналоговой части на дискретную отображается в математических моделях путем преобразования непрерывных фазовых переменных в логические переменные в местах сопряжения частей модели, обратное влияние выражается в преобразовании идеализированных логических сигналов в заданные функции времени, соответствующие электрическим сигналам заданной формы. Очевидно, что в содержательной части сообщений, передаваемых из одной части в другую, должны быть сведения либо о состояниях, выражаемых значениями фазовых переменных в интерфейсных узлах, либо о событиях - изменениях фазовых переменных. Обмен сообщениями может происходить многократно в течение акта одновариантного анализа.

В программно-методических комплексах конструирования происходит обработка графической информации. Содержательная часть сообщений относится к геометрическим элементам, их размерам и положению в пространстве. В программах технологической подготовки механической обработки деталей наряду с геометрической информацией о конструкциях заготовок в передаваемые сообщения могут входить сведения об инструменте, технологической оснастке, оборудовании, режимах обработки, нормах времени, траекториях движения инструмента и рабочих органов оборудования и т.п.

Другими словами, в каждом приложении совокупность используемых при обменах понятий, предметных переменных и числовых параметров существенно ограничена и достаточно определена для того, чтобы можно было ставить вопрос о типизации моделей и языка взаимодействия. Такие вопросы решаются в рамках технологий STEP/CALS. Число приложений, нашедших свое описание в прикладных протоколах STEP ограничено, но совокупность таких протоколов может расширяться.

Наряду с прикладными протоколами STEP, к числу средств семантической интеграции приложений относятся средства семантического Web. Семантический Web представляет собой иерархическую структуру, включающую несколько слоев моделей и языков описания информации (рис. 13).

В основании иерархической пирамиды находятся способы кодирования (форматы) данных, такие как, например, ASCII, UNICODE, графические форматы, а также ссылочные идентификаторы URI (Universal Resource Identifier).

Выше расположен уровень языков разметки HTML и XML (вместе со схемами HTML и XML или таблицами определения типов DTD).

Далее следует уровень с моделью RDF и языком RDFS (RDF Schema), служащими для описания метаданных, их интерпретации и обмена данными между приложениями. Если на уровне языков HTML и XML рассматриваются вопросы, связанные только со структурой документов, то на уровне RDF/RDFS решаются проблемы обеспечения семантической интероперабельности документов.

Верхний уровень занимают модели и языки онтологии, создаваемые на базе RDF/RDFS-средств. К языкам онтологии относятся DAML (DARPA Agent Markup Language), OIL (Ontology Interchange Language), OWL (Web Ontology Language) и др. С их помощью можно устанавливать эквивалентность классов, представлять теоретико-множественные операции над классами и т.п.

Применение семантического Web направлено на повышение эффективности решения следующих проблем:

- расширенная навигация в информационном Web-пространстве и многомерный поиск:
- семантическая интероперабельность порталов и других источников и хранилищ информации;
- реструктуризация информации в порталах.

Следует отметить, что для всех языков разметки на верхних иерархических уровнях, представленных на рис. 13, в качестве основы принят язык XML.

Рис. 13. Семантический Web

Программное обеспечение CALS-технологий

Программное обеспечение CALS-технологий должно выполнять те функции, которые обеспечивают создание и поддержку интегрирующей информационной среды для промышленных автоматизированных систем.

Во-первых, это функции управления данными, разделяемыми разными автоматизированными системами и подсистемами на этапах жизненного цикла изделий. Эти функции в настоящее время *выполняют системы управления жизненным циклом изделий* PLM. Внутри этапа проектирования управление

проектными данными, разделяемыми разными подсистемами САПР, выполняют *системы управления проектными данными* PDM.

Во-вторых, это функции управления данными и программами в распределенной сетевой среде, включая функции защиты информации. Эти функции реализуются в технологиях распределенных вычислений таких, как рассмотренные удаленный вызов процедур RPC, архитектура на основе посредников объектных запросов CORBA, объектная модель COM/DCOM, технология SOAP и др. Использование имеющихся систем распределенных вычислений позволяет разработчикам CALS-средств сконцентрировать усилия на решении специфичных задач и не тратить время на реализацию низкоуровневых функций взаимодействия в сетевой среде.

В-третьих, это программные средства логистической поддержки изделий, обслуживания сложной техники и обучения обслуживающего персонала правилам эксплуатации и ремонта изделий, представленные, в частности, интерактивными электронными техническими руководствами (ИЭТР), создаваемыми в CALS-системах с помощью специальных инструментальных средств. Развитые ИЭТР служат не только целям обучения пользователей, но выполняют также функции автоматизированного заказа материалов и запасных частей, планирования и учета проведения регламентных работ, обмена данными между потребителем и поставщиком, диагностики оборудования и поиска неисправностей. Примерами инструментальных систем создания ИЭТР могут служить TG Builder (компания "Прикладная логистика") или Adobe frameMaker+SGML (Adobe).

В-четвертых, к программному обеспечению CALS-технологий следует отнести многочисленные средства поддержки моделирования и обмена данными с использованием языка Express, которые можно объединить под названием STEP-средств (STEP Tools). К STEP-средствам относятся редакторы, компиляторы, визуализаторы, анализаторы, конверторы и т.п., связанные с языком Express. Редакторы помогают синтезировать и корректировать Express-модели. Анализаторы служат для синтаксического анализа и выявления ошибок, допущенных при написании модели. Анализатор входит в состав компилятора, который после анализа осуществляет трансляцию Express-моделей в ту или иную требуемую языковую форму. Визуализаторы генерируют графические представления моделей на языке Express-G. Конверторы используются для преобразования Express-моделей на основе языка Express-X.

В-пятых, к программному обеспечению CALS-технологий можно отнести средства поддержки языков SGML, XML, EDIFACT.

Примерами STEP-средств могут служить продукты компаний STEP Tools, EPM Technology AS, TNO и др.

Например с помощью программ ST-Developer компании STEP Tools реализуют SDAI-интерфейс на языках C, C++, Java, IDL/Corba, интерфейс Express-моделей к SQL базам данных и графическим ядрам ACIS и Parasolid машиностроительных CAD-систем, осуществляют тестирование Express-моделей, генерируют модели на языке Express-G.

Ряд STEP-средств предлагает Национальный институт стандартов и технологий США (NIST). Это средства оперирования обменными файлами и Express-моделями, трансляции моделей в C++ и IDL представления.

Компания Rational Rose предлагает транслятор Express-моделей в UML-представление.

Программные средства компании EPM Technology AS характеризуются разнообразием выполняемых функций. Так, программа EDMdeveloperSeat поддерживает базу данных с Express-моделями, EDMvisualExpress осуществялет визуализацию моделей с помощью расширения языка Express-G, EDMmodelChecker служит для диагностики допущенных нарушений правил языка Express.

Технологии распределенных вычислений и их программное обеспечение используются, но не являются специфичными в CALS-приложениях. Поэтому основными компонентами ПО CALS являются системы PLM, PDM и интерактивные электронные технические руководства (IETM).

Системы PDM предназначены преимущественно для информационного обеспечения проектирования - упорядочения информации о проекте, управления соответствующими документами, включая спецификации и другие виды представления данных, обеспечения доступа к данным по различным атрибутам, навигации по иерархической структуре проекта. Системы, аналогичные PDM, но в большей мере ориентированные на управление информацией в системах ERP, часто называют системами EDM (Enterprise Data Management). В системах PLM поддерживаются информационные связи не только внутри САПР, с их помощью обеспечивается взаимодействие различных АС на протяжении всего жизненного цикла изделий.

В последнее время усилия многих компаний, производящих программно-аппаратные средства автоматизированных систем, направлены на создание систем электронного бизнеса (E-commerce). Основу развитых систем E-commerce составляют средства PLM и CPC.

Среди систем E-commerce различают системы B2C и B2B.

Система B2C (Business-to-Customer) предназначена для автоматизации процедур взаимоотношений предприятия с конечными потребителями его продукции, чаще всего это взаимоотношения юридического лица с физическими лицами (покупателями товаров).

Но значимость систем E-commerce отнюдь не определяется организацией электронной торговли путем размещения на сайтах Internet витрин товаров и услуг. Цель электронного бизнеса заключается в объединении в едином информационном пространстве информации, во-первых, о возможностях множества организаций, специализирующихся на предоставлении различных услуг и на выполнении тех или иных процедур и операций по проектированию и изготовлению заказанных изделий, во-вторых, о запросах на использование этих услуг и заказах на поставки изделий и полуфабрикатов. В отличие от B2C такие E-commerce системы называют системами B2B (Business-to-Business). Эти системы автоматизируют процедуры взаимодействия юридических лиц друг с другом, более конкретно, системы B2B автоматизируют процессы обмена информацией между компаниями-партнерами.

Возникает задача создания единого информационного пространства, в котором функционируют автоматизированные системы управления взаимодействующих предприятий.

Технология интегрированного информационного пространства и управления данными – технология взаимодействия производителей, поставщиков и покупателей на различных этапах жизненного цикла изделий, направленная на оптимальное удовлетворение потребностей заказчиков в продукции и услугах. Благодаря более высокой степени специализации предприятий, проектированию под заказ, комплексному учету затрат на проектирование, изготовление, доставку

продуктов можно минимизировать временные и финансовые затраты при высоком качестве изделий. Чтобы использовать эти возможности, требуются системы PLM, главное назначение которых – обеспечивать информационную согласованность действий всех участников процесса создания продукции. В PLM учитывается, что число участников в цепи поставок может быть весьма значительным, причем состав участников непостоянен, а определяется исходя из конкретных задач и условий. Для эффективного управления процессами на протяжении всего жизненного цикла продукции все участники должны пользоваться доступными для правильного восприятия, интерпретации и исчерпывающе полными данными.

Именно системы PLM интегрируют данные, вырабатываемые и используемые системами CAD/CAM, ERP, SCM, CRM

В большинстве автоматизированных систем для обменов данными внутри системы используют те или иные форматы, или не являющиеся унифицированными, или признанные в ряде систем лишь как стандарты де-факто. Языки типа Express используют для межсистемных обменов и представления многократно используемых данных в общих базах данных, для выполнения роли внутренних форматов они неудобны. Поэтому в прикладные автоматизированные системы для связей с общей информационной CALS-средой должны быть включены конверторы для взаимных преобразований внутренних форматов данных в STEP-форматы. Такие конверторы также относят к программному обеспечению CALS-технологий.

В PDM разнообразие типов проектных данных поддерживается их классификацией и соответствующим выделением групп с характерными множествами атрибутов. Такими группами данных являются аспекты описания, т.е. описания изделий с различных точек зрения. Для большинства САПР машиностроения характерными аспектами являются свойства компонентов и сборок (эти сведения называют Bill of materials - BOM), модели и их документальное выражение (основными примерами могут служить чертежи, 3D модели визуализации, сеточные представления для конечно-элементого анализа, текстовые описания), структура изделий, отражающая взаимосвязи между компонентами и сборками и их описаниями в разных группах.

Вследствие большого объема проектных данных и наличия ряда версий проектов в системах PLM и PDM должна быть развитая система поиска нужных данных по различным критериям.

Защита информации в корпоративных системах

Понятие информационной безопасности относится к широкому кругу проблем обеспечения целостности информации и предотвращения несанкционированного доступа к информации в различных системах. При этом под целостностью информации понимают ее свойство оставаться семантически неизменной в условиях случайных или преднамеренных искажающих воздействий. Доступом к информации (к информационной системе) называют извлечение информации, ее копирование, модификацию или уничтожение. Несанкционированный доступ происходит при нарушениях установленных правил доступа.

Проблемы информационной безопасности, которые относятся к компьютерным системам и вычислительным сетям и которые решаются методами и средствами

информационных технологий, принято относить к дисциплине, называемой безопасностью информации. Ее предметом является защищенность информационных систем от внешних несанкционированных преднамеренных или случайных воздействий. Меры и средства для обеспечения безопасности информации составляют предмет защиты информации (или защиты данных).

Очевидно, что в САПР необходимо иметь средства, обеспечивающие безопасность информации.

Системы защиты информации должны обеспечивать как защиту от несанкционированного доступа и от несанкционированной модификации информации, так и восстановление информации после ее разрушения.

Меры обеспечения информационной безопасности делятся на административные, правовые, морально-этические, физические, программно-аппаратные.

К функциям программно-аппаратных средств систем защиты информации относятся:

- аутентификация пользователей и разграничение доступа к сетевым ресурсам, направленные на предупреждение несанкционированного доступа к защищаемым данным; в частности. эти функции включают использование паролей, администрирование привилегий пользователей;
- шифрование данных для обеспечения конфиденциальности и целостности информации;
- защита информации в сетях, включая мониторинг сетевого трафика, контроль доступа к сетевым ресурсам, фильтрацию пакетов, обнаружение попыток несанкционированного доступа и др.

Под угрозой безопасности в информационных системах понимают возможное воздействие на систему, которое может привести к несанкционированному доступу к информации, нанести ущерб безопасности информации в системе.

Различают три типа угроз безопасности: направленные на нарушение конфиденциальности информации, нарушение ее целостности и нарушение работоспособности информационной системы.

Реализация угрозы безопасности называется атакой на информационную систему. Объектами атак могут быть как непосредственно данные в системе, так и ее аппаратные средства, программные средства и собственно лица, работающие в системе. Атака на данные имеет целью копирование и/или искажение хранимой или передаваемой в системе информации. Атаки на программные средства чаще всего преследуют те же цели и осуществляются с помощью компьютерных вирусов. Особенностью атак на аппаратные средства может быть ухудшение или нарушение работоспособности аппаратуры. Атака на систему через персонал заключается в несоответствующем исполнении должностных обязанностей лицами, работающими в информационной системе, и осуществляется при нарушениях политики безопасности предприятия. Политика безопасности - это комплекс норм, правил и рекомендаций, принятых на предприятии и регламентирующих работу средств защиты информации от определенного множества угроз безопасности. Политика безопасности реализуется путем административно-организационных мер, физических и программно-технических средств и определяет структуру системы защиты информации.

Разновидность атаки, при которой несанкционированное воздействие кодируется во внешне безвредных данных, называется атакой, управляемой данными. Атака, при которой система намеренно выдает себя за другую систему, используя ее сетевой IP-адрес, носит название подмены IP-адресов. Атака, при которой действующий, установленный сеанс перехватывается и контролируется

атакующим, называется перехватом IP-подключения. Атаки типа перехвата IP-подключения могут происходить после успешной аутентификации, что позволяет атакующему работать от имени уже авторизованного пользователя. Основной способ защиты от перехвата IP-подключений состоит в шифровании на уровне сеанса или сети.

Рассмотрим способы защиты информации.

Аутентификация — проверка подлинности информационного объекта или субъекта при его попытке доступа к информации. Аутентификация пользователя информационной системы.чаще всего выполняется через пароли или персональные карточки с номерами идентификации. Однако пароли и идентификационные номера могут быть забыты, утеряны или похищены. Поэтому более надежными, но и сложными считаются методы аутентификации, основанные на достижениях биометрики. В этих методах идентификация личности основана на уникальных для каждого человека признаках, таких как отпечатки пальцев, окраска радужной оболочки глаз, геометрические особенности лица, спектральные составляющие речи, динамические характеристики ударов пальцев по клавишам.

Разграничение доступа (управление доступом) заключается в установлении и соблюдении правил доступа к информации.

Управление доступом к информации выполняется на основе дискреционного (избирательного) или мандатного способов. В этих способах для каждой группы пользователей устанавливаются свои права доступа. В дискреционном способе права пользования объектом (ресурсом) устанавливает субъект — владелец ресурса. Значениями прав при дискреционном доступе могут быть "чтение", "запись", "модификация" данных. Например, права доступа часто выражаются трехразрядным восьмеричным кодом ABC, в котором A — права владельца данных, B — членов некоторой выделенной группы, C — остальных пользователей, а три бита выражают право чтения, записи и исполнения соответственно. При мандатном способе информационные ресурсы различаются метками конфиденциальности, а пользователи — значением статуса. Для каждого статуса устанавливается порог метки конфиденциальности, выше которого доступ к ресурсам невозможен. Значениями меток конфиденциальности регламентируются доступ к сетевым информационным ресурсам, ресурсам ОС и к пользовательским данным или вход в систему, доступ к БД, доступ к приложениям и т.п.

Обнаружение атак на информационную систему составляет сущность защиты информации, поскольку обнаруженное воздействие легко нейтрализовать, например, с помощью специального программного обеспечения в маршрутизаторах или межсетевых экранах.

Чтобы обнаружить атаку, нужно знать ее особенности. Существуют два подхода к обнаружению:

- на основе априорно известных признаков атак (обнаружение злоумышленного поведения);

- на основе отличий поведения контролируемого объекта (узла или сети) от номинального (обнаружение аномального поведения). Номинальное поведение характеризуется нахождением параметров поведения в определенных пределах. Эти пределы устанавливаются в течение некоторого предварительного периода работы системы. Примерами контролируемых параметров могут быть число неудачных попыток входа в систему, число файлов, к которым обращается пользователь, загрузка процессора, продолжительность (в течение суток) работы

пользователя, применение пользователем непривычных для него сетевых сервисов или ресурсов и др.

Преимущество поведенческого подхода заключается в возможности обнаружения новых ранее неизвестных видов атак, хотя возможны и ложные оценки ситуаций. Поведенческий подход реализуют с помощью статистического анализа, экспертных систем или нейросетей. При статистическом анализе исследуются статистические характеристики параметров поведения пользователей. В экспертных системах используются совокупности правил, описывающих сценарии атак. Нейросети обнаруживают атаки после их обучения на предварительно отобранных примерах атак.

Для обнаружения вторжений создаются системы обнаружения вторжений (СОВ). В структуре СОВ выделяют компоненты:

- сенсоры — программные или программно-аппаратные агенты, устанавливаемые в разных местах компьютера или сети для сбора информации о процессах. протекающих в контролируемой системе;
- анализаторы — средства обработки информации, получаемой от сенсоров;
- компоненты реагирования — средства реагирования на атаку.

Различают хостовые и сетевые СОВ.

Хостовые СОВ анализируют отчеты о работе операционной системы хоста, файлы регистрации доступа и работы приложений, т.е. следят за соблюдением принятой политики безопасности.

Сетевые СОВ анализируют сетевой трафик для выявления атак, при обнаружении которых предупреждают администратора, завершают сеанс связи и возможно блокируют межсетевой экран. Сетевые СОВ сравнивают параметры пользовательского сеанса связи с эталонными описаниями способов несанкционированного доступа к информации, называемыми сигнатурами атак. Различают СОВ со статическими и динамическими сигнатурами. В первых используется база данных сигнатур, а во вторых, кроме этого, база знаний сигнатур, позволяющая обрабатывать оперативно появляющиеся новые сигнатуры.

Скрытым каналом называют канал связи, скрытно существующий дополнительно к основному каналу. Скрытые каналы могут как создавать угрозу безопасности вследствие неконтролируемой утечки данных, так и использоваться полезно для передачи информации незаметно для злоумышленников.

Несанкционированный доступ к информации при ее передаче в сетях общего пользования предотвращается с помощью шифрования данных. Однако утечка данных возможна из-за проникоовения в корпоративную сеть вредоносных программ (вирусов или программ-шпионов), которые получают секретную информацию и могут передать ее по скрытым каналам.

Скрытые каналы характеризуются информационной емкостью, интервалом модуляции и вероятностью коллизий. При этом используется понятие модуляции ресурса, понимаемой как изменение состояния ресурса при условии, что каждому из возможных состояний ресурса однозначно сопоставлен некоторый информационный символ.

Информационная емкость — это число состояний ресурса, которые могут быть использованы для кодирования информации.

Интервал модуляции — интервал времени, затрачиваемого на изменение состояния ресурса, соотвествующий одному символу кодирования.

Вероятность коллизий — вероятность изменения состояния ресурса на интервале модуляции.

В качестве ресурсов при атаках могут использоваться параметры трафика, такие как длина пакета или время между передачей соседних пакетов. Полезное использование скрытых каналов имеет место в стеганографии.

Компьютерным вирусом называют программу, которая может искажать другие программы, модифицируя их посредством включения в них своей возможно измененной копии, причем последняя сохраняет способность к дальнейшему размножению.

Название "троянский конь" закрепилось за программами, в которых наряду с объявленными действиями предусмотрены некоторые скрытые действия, ведущие к нарушению безопасности информации в информационной системе.

В отличие от вирусов компьютерный шпион внедряется в вычислительную систему с целью передачи хозяину шпиона информации, считываемой с диска атакованной системы. Программа-шпион ничего не разрушает, но представляет существенную угрозу безопасности, передавая злоумышленнику конфиденциальную информацию.

Отдельная программа, при запуске копирующая себя с одного хоста на другой, а затем запускающаяся на каждом вновь инфицированном хосте, называется червем (worm).

Активизация программ вирусов и шпионов возможна автономно или принудительно. Автономная активизация осуществляется после запуска программы, в которую внедрен вирус, при выполнении некоторых предусмотренных условий. Принудительная активизация происходит после внешнего сигнала от нарушителя или при определенных входных данных основной программы.

Одна из мер защиты информации в сетях заключается в установке между общедоступными и секретными объектами (между общедоступными сетями и корпоративной сетью) специального программного обеспечения, называемого межсетевым экраном (другие названия — брандмауэр или firewall). Межсетевой экран либо запрещает выполнение определенных действий на сервере, либо фильтрует пакеты, приходящие извне или исходящие, т.е. принимает решение о пропуске или нет в отношении каждого пакета. Фильтрация выполняется в соответствии с набором правил, в которых могут учитываться адреса, порты, заголовки и т.п. пакетов.

Различают межсетевые экраны (МЭ) сегментные, разделяющие две или более сетей; встраиваемые, защищающие определенные ресурсы серверов; персональные, защищающие отдельные компьютеры.

Межсетевые экраны классифицируют также по уровню ЭМВОС, на котором реализуется защита. Например, коммутаторы виртуальных ЛВС фактически являются МЭ, поскольку осуществляют фильтрацию пакетов между разными подсетями, выделенными группами портов или группами MAC-адресов. Существуют МЭ, анализирующие сетевой трафик на сетевом и транспортном уровнях и удаляющие запрещенные пакеты из потока; экраны сеансового или прикладного уровня, способные фильтровать пакеты по признакам, имеющимся в данных соответствующего уровня; и др.

В процессе фильтрации пакетов брандмауэр перед отправкой пакета во внешнюю сеть изменяет IP-адрес источника на свой, а при приходе ответов из внешней сети производит обратное преобразование адресов. Тем самым вне защищаемой сети ее внутренняя структура с IP-адресами узлов не видна.

Одновременно со сменой адресов фильтр принимает решение о том, передавать пакет или его отвергнуть.

Информационные системы классифицируются по степени достигаемой в них безопасности информации. В США Министерство обороны выпустило так называемую "Оранжевую книгу", в которой введены уровни и классы безопасности информации. Введено 4 уровня безопасности D, C, B, A, и 6 классов безопасности.

Неудовлетворительные с позиций безопасности системы относятся к уровню D.

Классы C1 и C2 уровня C характеризуют разграничение доступа к системе. В классе C1 доступ должен быть разрешен только именованным пользователям к именованным объектам, для аутентификации используется защитный механизм, например, пароли. В классе C2 дополнительно права доступа должны устанавливаться с точностью до пользователя, необходимо вести журнал регистрации доступа к базовым объектам.

В классе B1 дополнительно к требованиям уровня C вводятся метки безопасности для субъектов и объектов системы, как это принято для мандатного управления доступом. Кроме того, должно быть разделение адресных пространств процессов для их взаимной изоляции, необходима модель политики безопасности. В классе B2 ужесточены требования, предъявляемые к системам класса B1. Например, нужно регистрировать все попытки атак, оценивать пропускную способность скрытых каналов передачи данных, модель политики безопасности должна быть формализована. В дополнение к этим требованиям в классе B3 особо тщательно должна проектироваться базовая часть системы, из которой должны быть вынесены модули, не являющиеся критически важными с точки зрения защиты. Четко специфицируются функции администратора безопасности системы. Необходимо наличие пароцедуры восстановления системы после нарушений работоспособности.

Класс A1 характеризует системы, наиболее совершенные с точки зрения защиты информации. Так, дополнительно к свойствам предыдущих классов предусмотрено управление конфигурацией на всех этапах жизненного цикла по отношению к компонентам системы, сваязанным с обеспечением безопасности.

В основе многих методов шифрования (алгоритмов шифрования или криптоалгоритмов) лежат следующие приемы:

- перестановка символов (например, исходный текст размещается в таблице при ее заполнении по столбцам, столбцы переставляются в соответствии с ключевым словом, передаваемая последовательность символов образуется путем чтения полученной таблицы по строкам);
- замены символов одного алфавита символами того же или другого алфавита в соответствии с ключом;
- гаммирование (сложение кода исходного текста с псевдослучайной последовательностью цифр как в коде Вернама, но в условиях, когда способ выработки псевдослучайных цифр известен как отправителю, так и получателю).

Наиболее часто используют комбинации названных методов. При этом к каждому блоку текста перестановки и замены применяют по несколько раз с использованием перемешивания и рассеяния, чтобы скрыть статистические свойства алфавита.

Протоколы шифрования относят к представительному уровню. Пример протокола шифрования — SSL (Secure Sockets Layer).

Один из основных способов обеспечения безопасности информации в каналах передачи данных является применение шифрования. Вопросы шифрования являются предметом криптографии — науки о шифрах, при этом под шифром понимают совокупность методов и способов взаимного преобразования исходной (открытой) информации в недоступный (закрытый) для непосвященного человека вид, а под шифрованием понимают сам процесс такого преобразования.

Используют ряд подходов к шифрованию. Различают симметричную и асимметричную схемы шифрования.

В симметричных схемах (другое название — схемы с закрытым ключом) секретный ключ должен быть известен как отправителю, так и получателю. Ключ — это дополнение к правилу шифрования, представленное некоторым набором символов (например, двоичным кодом), управляющее преобразованием сообщения из исходного в зашифрованный вид. Ключ может быть операндом в действиях, выполняемых алгоритмом шифрования.

В асимметричных схемах (схемах с открытым ключом) шифрование производится открытым ключом, а дешифрование — секретным ключом, известным только получателю. Возможность асимметричного шифрования вытекает из наличия так называемых односторонних функций $Y = f(X)$, для которых обратное преобразование $X = f -1(Y)$ относится к трудным задачам, требующим полного перебора вариантов. Использование в обратном преобразовании ключа, который и является секретным, делает вычисление X простой процедурой. Но этот ключ известен только получателю, передавать его по ненадежной сети не требуется. Случайно подобрать секретный ключ злоумышленник не может, так как полный перебор при достаточной длине ключа за приемлемое время практически не осуществим.

В настоящее время все большее распространение получает комбинация симметричных и асимметричных схем. При этом сообщение кодируется закрытым ключом K по симметричной схеме, но сам ключ K для каждого сообщения новый и передается в закодированном по асимметричной схеме виде вместе с сообщением. Получатель декодирует сначала ключ K своим закрытым ключом D, а затем и все сообщение ключом K. Такая комбинация выгодна, во-первых, тем, что труднее взломать защиту, во-вторых, затраты времени на преобразования текстов становятся заметно меньше. Последнее связано с тем, что операции шифрования/дешифрования по асимметричной схеме значительно более трудоемки, чем по симметричной схеме. В результате короткий текст с ключом K шифруется медленно, а длинный основной текст сообщения существенно быстрее.

В зависимости от размера блока, на которые делится сообщение при шифровании, криптоалгоритмы к одному из следующих типов:

1. Потоковые шифры. Единицей кодирования является один бит (как в шифре Вернама). Результат кодирования не зависит от прошедшего ранее входного потока. Схема применяется в системах передачи потоков информации, то есть в тех случаях, когда передача информации начинается и заканчивается в произвольные моменты времени и может случайно прерываться. Наиболее распространенными представителями поточных шифров являются скремблеры. Суть скремблирования заключается в побитном изменении проходящего через систему потока данных. Практически единственной операцией, используемой в скремблерах является XOR – "побитное исключающее ИЛИ". Параллельно прохождению информационного потока в скремблере по определенному правилу генерируется поток бит – кодирующий поток из первоначального кода ключа. Как

прямое, так и обратное шифрование осуществляется наложением по XOR кодирующей последовательности на исходную.

2. Блочные шифры. Единицей кодирования является блок из нескольких байтов. Результат кодирования зависит от всех исходных байтов этого блока. Схема применяется при пакетной передаче информации и кодировании файлов.

Чем чаще обновляются ключи, чем они длиннее, тем труднее злоумышленнику их рассекречивание. Поэтому очевидна полезность периодической смены ключей, например, передачи вновь вводимого секретного ключа K по каналу связи, но предварительно зашифрованного с помощью некоторого другого секретного ключа D. Подобная передача ключей выполняется в асимметричных схемах.

В практических методах обычно используют блочное шифрование (один и тот же алгоритм с одним и тем же ключом применяется к каждому из блоков исходного текста).

Блочные шифры являются основой, на которой реализованы практически все криптосистемы. Методика создания цепочек из зашифрованных блочными алгоритмами байт позволяет шифровать ими пакеты информации неограниченной длины. Такое свойство блочных шифров, как быстрота работы, используется асимметричными криптоалгоритмами, медлительными по своей природе. Отсутствие статистической корреляции между битами выходного потока блочного шифра используется для вычисления контрольных сумм пакетов данных и в хешировании паролей.

Следующие разработки всемирно признаны стойкими алгоритмами и публикаций о универсальных методах их взлома в средствах массовой информации на момент создания материала не встречалось.

Таблица 1

Название алгоритма	Автор	Размер блока,бит	Длина ключа, бит
IDEA	Xuejia Lia and James Massey	64	128
CAST128		64	128
BlowFish	Bruce Schneier	64	128-448
ГОСТ 28147-89		64	256
TwoFish	Bruce Schneier	128	128-256
MARS	IBM	128	128-1048

Стойкий блочный шифр должен удовлетворять следующим условиям:

- функция шифрования должна быть обратимой;

- не должно существовать иных методов прочтения сообщения X по известному блоку Z закодированного сообщения, кроме как полным перебором ключей;

- не должно существовать иных методов определения, каким ключом было произведено преобразование известного сообщения X в сообщение Z, кроме как полным перебором ключей.

Наиболее надежно шифрование сообщения X, выраженного двоичным кодом, выполняется с помощью шифра Вернама. В этом шифре шифрование сводится к поразрядной операции логического сложения по модулю 2:

$C = X \veebar K$,

где K — ключ, C — результат шифрования, передаваемый по каналу связи. Пример для шифра Вернама:

X = 1011 0001 0111 0101

K = 1101 1010 0100 0011

C = 0110 1011 0011 0110

На приемном конце аналогичное действие выполняется над C с помощью K, известного получателю. В нашем примере:

C = 0110 1011 0011 0110

K = 1101 1010 0100 0011

X = 1011 0001 0111 0101

В шифре Вернама ключ K — случайный код, он имеет ту же длину, что и сообщение X. Кроме того, для каждого нового сообщения используется новый ключ. Эти условия обеспечивают абсолютную стойкость зашифрованного текста к взламыванию (несанкционированному распознаванию).

Однако соблюдение условий абсолютной стойкости на практике неосуществимо, так как требует передачи нового ключа (например, от отправителя к получателю) с каждым новым сообщением. Поэтому в симметричных схемах передают ключи некоторым надежным способом сравнительно редко и используют каждый ключ многократно. Вероятность взламывания зашифрованных текстов при этом отлична от нуля, но она может быть сделана крайне малой при достаточно изощренных шифрах и больших длинах ключей.

Одним из применений шифрования является электронная подпись, предназначенная для удостоверения подлинности документа, пересылаемого по сети. Здесь применяют асимметричную схему шифрования: документ (чаще его хеш-функция) перед отправкой шифруется закрытым ключом отправителя, а дешифрируется открытым ключом получателя.

Таким образом, получателю передается документ и его хеш-функция, получатель имеет возможность сравнить значения хеш-функции, присланное и вычисленное им. Если хотя бы один бит в вычисленной не совпадет, то, , следовательно, текст по ходу пересылки подвергся несанкционированному изменению.

Примером алгоритма шифрования с закрытым ключом может служить алгоритм DES, утвержденный в качестве стандарта США в 1980 г. Шифрование происходит блоками по 64 бита. Сначала блок делится пополам, затем правая часть ставится на место левой, а на место правой — результат поразрядной операции отрицания равнозначности над двумя частями блока. Далее выполняются перестановки и замены с помощью закрытого ключа.

Пример алгоритма с открытым ключом — алгоритм RSA. В нем два числа E и n представляют вместе открытый ключ, а совокупность чисел K и n — закрытый ключ. Выполняются следующие операции:

1. Выбор двух больших простых чисел p и q.
2. Вычислениеn = pq и m = (p−1)(q−1).
3. Выбор случайного целого числа E, не имеющего общих множителей с m.
4. Вычисление K из уравнения (KE) mod m = 1. Отметим, что (KE) mod m = ((K mod m)(E mod m)) mod m.
5. Если X — блок исходного текста, то результат шифрования этого блока: C = XE mod n.
6. Результат дешифрования: X = CK mod n.

Чтобы злоумышленник мог найти K по E, нужно подобрать соответствующее m, т.е. найти простые множители p и q, а это трудная переборная задача при больших n. Обычно в RSA число n — 200-значное десятичное.

Для шифрования в России рекомендуется алгоритм шифрования по ГОСТ 28147-89, в нем используются блоки по 64 бита, ключи длиной 256. Сначала блок разбивается на два подблока по 32 бита. Затем выполняется следующий цикл:

правый подблок суммируется с первыми 32 битами ключа, производятся замены тетрад и циклический сдвиг в сторону старших разрядов, результат суммируется по модулю 2 с левым подблоком и подблоки меняются местами. Описанный цикл повторяется 32 раза с использованием в каждом цикле следующих 32 битов ключа (тем самым каждые 32 бита ключа используются четырежды).

Основные функции систем PDM

В информационных моделях приложений фигурируют сущности (типы данных) и связи между ними. Установление сущностей, их атрибутов, связей и атрибутов связей означает *структурирование проектных данных*. Структура изделий обычно может быть представлена иерархически, в виде дерева. Иерархическая форма удобна при внесении и отслеживании изменений в модели, например, при добавлении и удалении сущностей, изменениях их атрибутов, введении новых связей. Поэтому одной из первоочередных функций систем PDM является поддержка интерактивной работы пользователя при создании моделей изделий (процессов), структурирование описаний проектируемых объектов, предъявление пользователю этой иерархической структуры вместе с возможностями навигации по дереву и получению нужной информации по каждой указанной пользователем структурной компоненте.

Например, в системе PDM STEP Suite элементы дерева, представляющего структуру изделия, могут соответствовать сборочным узлам, агрегатам, блокам, отдельным деталям. Навигация по дереву позволяет просматривать относящиеся к структурным единицам документы, геометрические модели, чертежи и другие атрибуты.

В системе BaanPDM основными типами данных являются документы и изделия. Экземпляры сущностей идентифицируются и описываются с помощью набора атрибутов, среди которых имеется уникальный индентификатор (ключ) объекта и ряд дополнительных атрибутов, например, тип документа, автор, количество входящих в документ страниц. Стандартные функции поддержки объекта включают возможность добавлять объекты (при этом добавляется уникальный идентификатор и другие атрибуты), модифицировать атрибуты объекта и удалять объекты.

В системе PDM, разработанной фирмой Cadence для своей САПР, предусмотрена иерархическая организация проектных данных, описывающих проектируемые СБИС (сверхбольшие интегральные схемы), с выделением уровней библиотек, категорий, ячеек, видов. Ячейка - базовый объект, который может иметь несколько различных представлений (видов). Ячейки объединяются в родственные группы - категории, а категории - в библиотеки. Разработчик с помощью системной среды имеет доступ к проектным данным, может создавать свои библиотеки, ячейки, виды.

Интерфейс с пользователем поддерживается *визуализацией данных проекта* одновременно в нескольких окнах. Для визуализации данных разных аспектов в PDM имеется ряд браузеров. Типичные изображения, создаваемые браузерами, - дерево проекта или его фрагментов; различные виды, такие как 2D чертеж или 3D изображение; описания моделей; принципиальные схемы; атрибуты объекта (исполнитель, номер версии, дата утверждения и т.п.). Иногда для визуализации и редактирования данных в PDM конкретной фирмы привлекаются браузеры и редакторы других изготовителей.

Для примера на рис. 14 показан фрагмент дерева изделия. Обычно на экране дисплея рядом с названием компонента структуры высвечивается также присвоенный ему код. Выбор любого компонента (узла дерева) позволяет, во-первых, получить в появляющихся окнах требуемую информацию о компоненте. во-вторых, для компонента, являющегося сборкой, раскрыть следующий по иерархии фрагмент, в котором данный компонент будет представлен уже корневым узлом.

Рис. 14. Фрагмент дерева изделия

Управление версиями проекта и внесением изменений в проект должно обеспечивать целостность проектных данных. Если в проект нужно внести изменения, то создается новая версия проекта, основанная на первоначальном проекте, и изменения вносятся уже в эту новую версию. Исходный вариант проекта при этом сохраняется в прежнем виде. Одна версия каждого объекта является текущей или активной версией. Если имеется несколько версий объекта, то текущей является та, которая последней подвергалась изменениям.

В системе BaanPDM принята следующая схема управления версиями. Если версия объекта создана впервые, ей назначается статус "неопределенная". Если версия объекта готова для общего доступа, ее следует занести в сборник, и затем BaanPDM назначает ей статус "готово к выпуску". Выпуск объекта делает его описание доступным для использования в других подразделениях и производства. Если кто-либо желает сделать изменения в готовой к выпуску версии объекта, он должен извлечь ее из сборника. Этой версии присваивается статус "находящаяся в процессе изменения", который показывает, что готовится новая версия, а новой версии - неопределенный статус.

В системе NELSIS CAD Framework предусмотрены следующие статусы для версий документов: рабочий – версия с таким статусом находится в работе, ее можно модифицировать; принятый – именно версия с этим статусом является основой для взаимодействия частей проекта, она служит для обмена данными между объектами, ее модификации осуществляются через рабочий статус; архивный - статус, присваиваемый предыдущим сохраняемым версиям; порождаемый – статус зарезервирован для вновь создаваемых объектов, например, при синтезе проектных решений. Разработчик сам изменяет статус объектов.

Аналогично в системе PDM STEP Suite одна из версий проекта является рабочей (активной), с ней работают пользователи. Но можно обращаться и к любой другой версии. В процессе коллективной работы хранимый в БД документ, чертеж или модель могут быть взяты для дальнейшей проработки. Тогда исходная версия документа помечается как находящаяся в процессе редактирования. После редактирования созданная новая версия хранится вместе с

предыдущей. При этом для каждой версии документа можно определить породившую ее исходную версию.

Целостность данных поддерживается также тем, что нельзя одновременно изменять один и тот же объект разным разработчикам, каждый из них должен работать со своей рабочей версией. Другими словами, необходимо обеспечение синхронизации изменения данных, разделяемых многими пользователями.

Для этого выполняется авторизация пользователей и разрабатываются средства ведения многих версий проекта. Во-первых, пользователи подразделяются на классы (администрация системы, руководство проектом и частями проекта, группы исполнителей-проектировщиков) и для каждого класса вводят определенные ограничения, связанные с доступом к разделяемым данным; во-вторых, доступ регламентируется по типам разделяемых данных. Данным могут присваиваться различные значения статуса, например, "правильно", "необходимо перевычисление", "утверждено в качестве окончательного решения" и т.п. Собственно синхронизация выполняется с помощью механизмов типа рандеву или семафоров, рассматриваемых в пособиях по параллельным вычислениям.

В системе BaanPDM каждому пользователю в зависимости от его роли назначается уровень прав доступа - один из восьми возможных. На низшем уровне пользователь может только просматривать данные. На высшем уровне, присваиваемом старшему администратору, допускаются любые модификации данных любого проекта и архивов. В функции лица, являющегося системным администратором, входят упорядочение данных с их распределением по дискам, контроль за правами доступа пользователей, связь с внешними системами (управление импортом-экспортом данных) и др.

В современных условиях обострившейся конкуренции на рынках продукции выигрывают те предприятия, которые проектируют изделия высокого качества и меньшее время тратят на проектирование и производство. Использование систем PLM и PDM должно способствовать повышению конкурентоспособности предприятий. Поэтому в число функций систем PLM и PDM входят управление качеством продукции и поддержка совмещенного (параллельного) проектирования.

Совмещенное проектирование используют в целях сокращения временных затрат, однако совмещение процедур, взаимосвязанных по входным и выходным данным, возможно не всегда и лишь при наличии специальных методик и программных средств. Примеры совмещения проектных процедур были приведены в гл. 5 применительно к проектированию СБИС. Совмещение некоторых процедур имеет место и в САПР машиностроения, например, в ряде случаев проектирование технологических процессов можно начинать до полного окончания конструирования.

Совмещенное проектирование требует организации групповой работы проектировщиков. В качестве примера на рис. 15 показана типичная схема разделения рабочего пространства между параллельно работающими пользователями.

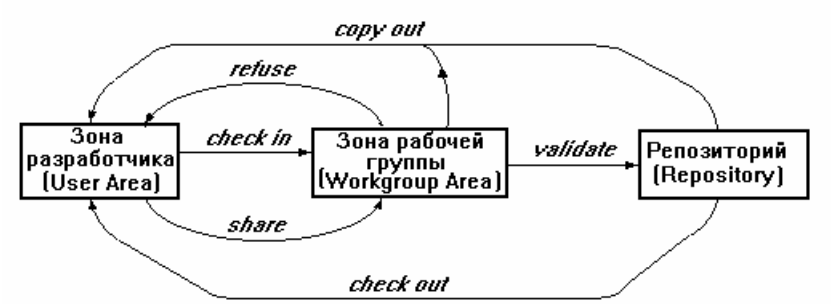

Рис. 15. Информационные связи разработчиков с зонами базы данных

Следующими важными функциями PDM являются *управление документами и документооборотом*. Проектная документация характеризуется разноплановостью и большими объемами. В процессе проектирования используют чертежи, конструкторские спецификации или BOM, пояснительные записки, ведомости применяемости изделий, различного рода отчеты и др. Кроме того, в интегрированных автоматизированных системах проектирования и управления в документооборот входит большое число документов, связанных с процедурами маркетинга, снабжения, планирования, администрирования и т.п. .

Необходимо обеспечить автоматический учет влияния и распространения вносимых в проект изменений на другие части проектной документации.

Для подготовки, хранения и сопровождения необходимых документов, в том числе чертежей и схем, в PDM включают специализированные системы управления документами и документооборотом или адаптируют полнофункциональные системы делопроизводства, разработанные независимо от конкретных PDM.

Часто используют программы Lotus Notes и Lotus Domino компании Lotus Development. Возможности управления чертежно-конструкторской документацией, подготовленной в AutoCAD и Microstation, имеются в продуктах DOCS Open (компания Hummingbird), CADLink, входящем в систему управления документами и бизнес-процессами Documentum, Search (белорусская компания Интермех) и ряде других.

В системе Search осуществляются хранение и поиск данных, доступ к ним, документооборот, разработка спецификаций, внесение изменений и др. Для этого имеются редактор извещений об изменениях в проекте, средства обеспечения групповой работы над проектом, модуль доступа к документам, расположенным на других узлах сети. Редактирование и просмотр выполяются с помощью внешних редакторов.

Следует отметить, что организация совмещенного проектирования, интеграция автоматизированных систем проектирования и управления на современных предприятиях возможны только в распределенной среде. *Распределенные хранение и обработка информации* в большинстве случаев осуществляются на базе применения технологий SOAP, CORBA или DCOM, языков Java и XML. Данные проекта при этом находятся в хранилищах данных, т.е. в нескольких базах распределенного банка данных. Находят применение трехзвенные распределенные системы с уровнями сервер баз данных – сервер приложений – клиенты.

Управление проектами (процессом проектирования) также входит в число функций PDM. Проектирование состоит из многих шагов, объединяемых в потоки работ (workflow). Управление потоком работ включает в себя большое

число действий и условий, поддерживающих параллельную работу многих пользователей над общим проектом. Необходимо распределить работы как между исполнителями, так и во времени, а также обеспечить контроль выполнения работ.

Шаги заданного или динамически определяемого маршрута работ могут представлять собой выполнение проектных операций и процедур, пересылку документов и файлов другим пользователям, изменение статуса объекта, просмотр, контроль и утверждение инженерных проектов и внесения в них изменений и т.п. Между шагами перемещается пакет документов. На шагах маршрута документы проекта обрабатываются, видоизменяются, оцениваются, пакет автоматически пополняется и, в конечном счете, проектная документация выпускаются в производство.

Управление потоком работ выполняется на основе моделей вычислительных процессов. Используются спецификации моделей, принятые в CASE-системах, например, диаграммы потоков данных, ориентированные графы, UML-диаграммы. Сначала модели составляют в терминах проектных заданий, а затем система осуществляет их покрытие имеющимися проектирующими программами и программными модулями. Применяют также описания на языках расширения или 4GL.

Часто управление крупными проектами, включающее распределение большого числа работ во времени и между исполнителями, выполняется программами, относящимися к специальной группе систем управления проектами. В эту группу входят программы верхнего уровня, такие, как Artemis Project (фирма Metier), Primavera Project Planner (Primavera Systems), Open Plan (Welcom Software), среднего уровня - Time-Line (Symantec), Microsoft Project (Microsoft) и др.

Например, система Project Manager Workbench служит для одновременного управления различными проектами с оптимальным распределением ресурсов, помогает построить иерархическую структуру плана, сформировать несколько видов отчетов, описывающих расписания, расходы, контроль качества. С ее помощью контролируют общее использование ресурсов, составляют расписания разнохарактерных работ. В качестве ресурсов могут рассматриваться люди, финансовые средства, устройства.

Интеграция данных на ранних этапах развития PDM связывалась только с организацией сквозного проектирования изделий в рамках конкретной САПР. В настоящее время в связи с развитием CALS-технологий основным содержанием проблемы интеграции стало обеспечение *интерфейса САПР с другими автоматизированными системами*. Проблема решается с помощью поддержки типовых форматов, например, путем конвертирования данных из общепринятых форматов во внутренние представления конкретных САПР.

В CALS-технологиях взаимодействие систем основано на стандартах STEP, поэтому в ряде PDM имеются конверторы из предложенного в STEP языка Express. В STEP введен прикладной протокол AP208, представляющий собой информационную модель, относящуюся к управлению процессами изменений в жизненном цикле изделий. В соответствии с AP208 внесению изменений предшествуют идентификация фактов (недостатков), требующих внесения изменений, установление вызвавших их причин и определение лиц, вносящих изменения. Среди других форматов данных обычно используются IGES, DXF, VRML, SAL, EDIF, текстовые и 2D графические форматы и др.

Адаптация САПР к условиям конкретных предприятий может быть осуществлена с помощью языков расширения. *Язык расширения* - язык

программирования, позволяющий адаптировать и настраивать системную среду на выполнение новых проектов. Язык расширения должен обеспечивать доступ к различным компонентам системной среды, объединять возможности базового языка программирования и командного языка, включать средства процедурного программирования. Для большинства языков расширения базовыми являются Lisp или C.

Примерами таких языков могут служить язык Skill из Design Framework-2 фирмы Cadence или язык CCL (CASE Comment Language) фирмы Matra Datavision, являющиеся Lisp-подобными, или язык AMPLE из PDM Falcon Framework фирмы Mentor Graphics, базирующийся на языках C и ПАСКАЛЬ.

Примеры систем PDM

Многие известные PDM создавались фирмами, специализировавшимися на разработке САПР или АСУ. Примерами таких PDM, разработанных в свое время с ориентацией на конкретные САПР или АСУ, были системы iMAN (Unigraphics Solutions), Optegra (Computervision), ProPDM (PTC), Euclid Design Manager (Matra Datavision), BaanPDM (BAAN) и др. Ряд других систем PDM разрабатывался независимо от конкретных САПР или АСУ, это, например, системы ENOVIA (IBM/Dessault), PDM StepSuite (НПО "Прикладная логистика"), Party Plus (Лоция Софт).

В настоящее время основные разработчики САПР в машиностроении считают целесообразным предлагать комплексные системы PLM, в состав которых входят как модули CAD/CAM/CAE, так и PDM.

Так, компания Dessault Systemes создает систему ENOVIA на базе приобретенной PDM ProductManager. ENOVIA предназначена для моделирования и управления данными об изделиях, процессах и ресурсах на различных этапах жизненного цикла промышленной продукции от концептуального проектирования до эксплуатационного обслуживания. Это распределенная на базе Web-технологий система управления данными, способствующая интеграции систем проектирования, производства и управления внутри предприятия и позволяющая отдельным фирмам объединяться в виртуальные предприятия. Управление проектами и изменениями данных, их распределение, интерфейс с системами ERP - далеко не полный перечень функций этой системы.

Кроме ENOVIA, IBM/Dessault развивают систему SmartTeam. В базовый комплект системы SmarTeam входят модуль создания и редактирования моделей, СУБД (Interbase или Oracle), визуализатор, модуль сопряжения с различными САПР (в список входят SolidWorks, MDT, Inventor, Microstation, Solid Edge, AutoCAD). Базовый комплект может расширяться путем добавления модулей документооборота, интеграции с ERP, SCM и CRM-системами, взаимодействия с партнерами через Internet и др. Состав системы SmarTeam и ее связи с CAD и ERP-системами иллюстрирует рис. 16.

Создаваемая в среде SmarTeam информационная модель объекта состоит из двух частей. Одна часть служит для описания состава изделия (в виде дерева), его структуры (в виде файлов с данными о сборках), геометрии и материала деталей. Другая часть содержит данные о технологических процессах изготовления объекта в виде дерева операций и переходов и автоматически формируемой технологической документации.

Рис. 16. Компоненты и связи системы SmarTeam

Компания Unigraphics Solution осуществляет преобразование систем iMAN и Metaphase в новую PDM Teamcenter. В этой PDM имеются подсистемы управления данными на стадиях проектирования и производства.

Компания PTC располагает двумя системами PDM - это Pro/Intralink и более современная Windchill. Система Windchill основана на использовании Internet и Web-технологий для информационного взаимодействия многих предприятий и потому может позиционироваться как система CPC. Windchill охватывает все этапы проектирования, выполняет функции, которые присущи системам документооборота, управления проектами, конфигурацией и изменениями проектных данных. Системы CPC функционируют в гетерогенной среде, охватывающей пространство, не ограниченное рамками отдельных предприятий и корпораций. Система CPC, отвечая на запросы пользователей, может собирать необходимые данные из web-сайтов, баз данных ERP или PDM систем и, преобразуя в единый формат, предоставляет их пользователю. Имеются возможности планирования и моделирования производственных и логистических процессов.

В SolidWorks используется PDM/Works, в SolidEdge - заимствованная система управления документами SharePoint Portal Server.

Российская компания Аскон предлагает оригинальную систему Лоцман PLM, реализуемую в виде трехзвенной системы распределенных вычислений.

Компания Consistent Software разрабатывает оригинальную PDM-систему OutdoCS PDM и предлагает комплексную систему PartY Plus, разработанную фирмой Лоция Софт. Система PartY Plus предназначена для управления информацией об изделиях, проектах, сооружениях на протяжении всех этапов их жизненного цикла. PartY Plus включает в себя три основных продукта, которые

могут использоваться как автономно, так и совместно - это PDM PartY, система управления документами DOCS Open, управления документооборотом и бизнес-процессами LS Flow. В качестве СУБД нужно использовать одну из систем Sybase Adaptive Server, MS SQL Server или Oracle.

На роль PDM претендует система ведения архива технической документации и управления проектными данными Search белорусской фирмы Интермех. Search выполняет функции: хранение документов (чертежи, спецификации, руководства и др.), поиск и доступ к ним, управление версиями документов и изменениями в них, визуализация структуры изделий в виде дерева связей, поддержка групповой работы над проектом (редактирование, маршрутизация документов), формирования различного рода справок и отчетов, регулирование прав доступа к архиву, импорта данных из внешних баз. Архив создается на базе СУБД Oracle или InterBASE (компания Borland). Обеспечивается удаленный доступ к архиву с помощью Web-браузеров. В системе имеются редактор спецификаций, редактор извещений об изменениях в проекте, модуль доступа к документам, расположенным на других узлах сети, база данных (электронный архив), текстовый редактор, объединенные с чертежной системой. Search "понимает" внутренний язык AutoCADa. Для ее использования необходима СУБД Interbase (компания Borlabd).

Белорусская компания Омегасофтвер разработала систему Omega Production, в которой предусмотрены структурирование данных об изделиях, технологических процессах, оснастке и оборудовании, управление документами и документооборотом, управление конфигурацией изделий, контроль изменений, вносимых в проект, интерфейс с другими САПР. Кроме того, в Omega Production имеются модули оперативного управления производством, контроля качества продукции, управления запасами и поставками материалов и комплектующих, что характерно для логистических систем. Следовательно, Omega Production может служить основой для интеграции систем проектирования и управления предприятием.

Упражнения и вопросы для самоконтроля

1. Назовите причины появления стандартов STEP.
2. В чем заключается отличие технологий САПР и CALS?
3. Что является предметом стандартизации в CALS-технологиях?
4. Что называют прикладным протоколом в STEP-технологиях? Что такое интегрированные ресурсы?
5. На какие классы подразделяются геометрические модели в протоколе AP203?
6. Представьте на языке Express IDEF1X-диаграмму, построенную для сущностей "студенческая группа", "студент", "преподаватель", "дисциплина",.
7. Опишите на языке Express фигуру квадрат.
8. Опишите назначение и структуру обменного файла в языке Express.
9. Для чего нужны разновидности языка Express, такие как Express-X и Express-V?
10. Чем обеспечивается целостность данных в системах PDM?
11. Назовите основные характеристики ИЭТР.
12. Поясните назначение языков разметки. Что такое декларация DTD?
13. Дайте сравнительную характеристику языков HTML и XML.
14. Назовите основные функции систем PDM.
15. Дайте характеристику подхода к контролю качества продукции, принятому в стандартах ISO 9000.

Список литературы

1. Гаврилов Д.А. Управление производством на базе стандарта MRP II. СПб: Питер, 2003.
2. Грушвицкий Р.И., Мурсаев А.Х., Угрюмов Е.П. Проектирование систем на микросхемах программируемой логики. - СПб.: БХВ-Петербург, 2002.
3. Колчин А.Ф., Овсянников М.В., Стрекалов А.Ф., Сумароков С.В. Управление жизненным циклом продукции. М.: Анахарсис, 2002.
4. Ли К. Основы САПР (CAD/CAM/CAE). СПб.: Питер, 2004.
5. Маклаков С.В. BPwin, ERwin. CASE-средства разработки информационных систем. М.: Диалог-МИФИ, 1999.
6. Норенков И.П., Кузьмик П.К. Информационная поддержка наукоемких изделий (CALS-технологии). М.: МГТУ им. Н.Э.Баумана, 2002.
7. Норенков И.П., Трудоношин В.А. Телекоммуникационные технологии и сети. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2000.
8. Образцов И.Ф., Савельев Л.М., Хазанов Х.С. Метод конечных элементов в задачах строительной механики летательных аппаратов. М.: Высш. шк., 1985.
9. Острейковский В.А. Теория систем. М.: Высш. шк., 1997.
10. Питтс Н. XML за рекордное время. М.: Мир, 2000.
11. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. М.: Радио и связь, 2001.
12. Системы автоматизированного проектирования: Учеб. пособие для втузов: В 9 кн./ Под ред. И.П.Норенкова. М.: Высш. шк., 1986.
13. Роджерс Д., Адамс Дж. Математические основы машинной графики. М.: Мир, 2001.
14. Томашевский В., Жданова Е. Имитационное моделирование в среде GPSS. - М.: Бестселлер, 2003.
15. Черненький В.М. Имитационное моделирование. М.: Высш. шк., 1990.
16. Пупков К.А. Автоматизированная разработка систем управления. М.: Изд. МГТУ им. Н.Э.Баумана, 1993.
17. Норенков И.П. Основы автоматизированного проектирования. Учебник для вузов. – 3-е изд., перераб. и доп. – М.: Изд-во МГТУ им. Н.Э.Баумана, 2006.

ОПИСАНИЕ КУРСА И ПРОГРАММА

Цель и задача курса

Целью курса является освоение информационных технологий, обеспечивающих автоматизацию разработки, создания, испытания и эксплуатации технических изделий на всех этапах жизненного цикла;

Задачами курса являются:

- освоение структуры и содержания процесса разработки изделий и систем различного назначения.
- изучение типов математических моделей изделий и их компонентов, используемых на различных этапах разработки: концептуальной модели изделия, регулирования технического задания (ТЗ) и исходных данных для проектирования, создание опытного образца, обработки и испытаний, эксплуатации.
- изучение программных систем для автоматизированного проектирования CAE, CAD и CAM.
- изучение CALS – технологий и применение их для реализации процесса сквозного проектирования и эксплуатации изделий и систем на всем жизненном цикле:

Область знаний: математическое моделирование, оптимизация параметров и структуры изделий и систем САПР.

Уровень обучения – дополнительное образование.

Специальность и направление: “Управление и информатика в технических системах”, “Автоматизация и управление”.

Курс: теоретический и практический.

Инновационность курса по:

- содержанию

Курс включает в себя новейшие достижения в сфере автоматизации управления и автоматизации процессов проектирования сложных технических изделий и систем. Используются научные и методические материалы в области автоматизированных систем автора УМК. Обосновано, что математическое, физическое и натурно-математическое моделирование является научной основой CALS – технологий, а автоматизация проектирования является инструментальной базой. Рассматривается математическое и программное обеспечение процессов анализа и синтеза принятия решений при проектировании, а также состав технического сетевого обеспечения систем автоматизированного проектирования и комплексирования САПР с другими автоматизированными системами (CALS - технологии).

- методике преподавания: лекция, практические занятия, курсовой проект.

- литературе:

В качестве базовой литературы используется цикл учебных пособий: Пупкова К.А. “Автоматизированная разработка систем управления”, учебник Норенкова И.П**.** “Основы автоматизированного проектирования”, учебник “Классические и современные методы теории автоматического управления” в 5-ти томах, под редакцией К.А. Пупкова.

- организации учебного процесса:

Введение курсового проекта обеспечивает освоение теоретического материала в процессе проектирования реального изделия.

Структура курса (36 часов лекций, 18+18 часов практические занятия + курсовой проект).

Темы лекций:

Лекция 1.

Введение. Понятие процесса разработки изделий и систем. Моделирование и автоматизация проектирования. Системы автоматизированного проектирования – интеллектуальные системы.

Самостоятельная работа студента: 2 часа.

Лекция 2.

Структура процесса разработки изделий и систем. Стадии разработки. Понятие конструктивной модели изделия или системы. Формирование технического задания и исходных данных. Классификация изделий и параметров, используемых при проектировании.

Самостоятельная работа студента: 2 часа.

Лекция 3.

Типы автоматизированных систем управления и их связь с системами автоматизированного проектирования. Этапы жизненного цикла произведенных изделий и систем. Структура САПР и разновидности САПР. CALS – технологии - основа для поддержки информационного и технического обеспечения изделий и систем в течение жизненного цикла.

Самостоятельная работа студента: 2 часа.

Лекция 4.

Математическое обеспечение информационного процесса разработки изделий и систем. Математическое обеспечение анализа и принятия решений на концептуальном уровне. Модели линейной и нелинейной оптимизации. Имитационное моделирование.

Самостоятельная работа студента: 2 часа.

Лекция 5.

Математические модели изделий и систем на этапе эскизного проектирования. Параметрическая и структурная оптимизация. Модели многокритериальной динамической оптимизации. Краткое описание языка GPSS.

Самостоятельная работа студента: 2 часа.

Лекция 6.

Математическое обеспечение подсистем машинной графики. Трехмерная графика. Построение геометрических моделей. Поверхностная модель.

Самостоятельная работа студента: 2 часа.

Лекция 7.

Математическое обеспечение анализа и синтеза проектных решений. Формирование базы знаний интеллектуальной системы принятия проектных решений. Формирование структуры и уровней внутренних и внешних воздействий для исследования при проектировании.

Самостоятельная работа студента: 2 часа.

Лекция 8.

Структурный синтез в задачах проектирования. Процедура синтеза проектных решений. Задача принятия решений.

Самостоятельная работа студента: 2 часа.

Лекция 9.

Методы структурного синтеза в системах автоматизированного проектирования. Элементы теории интеллектуальных систем. Элементы теории сложности. Нелинейная оптимизация. Метод ветвей и границ. Эволюционные алгоритмы. Генетические алгоритмы.

Самостоятельная работа студента: 2 часа.

Лекция 10.

Технические средства систем автоматизированного проектирования. Структура технического обеспечения. Пространственно распределенные системы автоматизированного проектирования. Проблемы взаимодействия САПР различных компонентов изделий и систем.

Самостоятельная работа студента: 2 часа.

Лекция 11.

Типы вычислительных средств в составе САПР. Автоматизированные рабочие места.

Самостоятельная работа студента: 2 часа.

Лекция 12.

Корпоративные сети. Проблемы глобальной стандартизации. Начала CALS – технологий. Локальные вычислительные сети. Стеки протоколов и типы сетей.

Самостоятельная работа студента: 2 часа.

Лекция 13.

Программное обеспечение автоматизированных систем. Программные системы CAE, CAD и CAM.

Самостоятельная работа студента: 2 часа.

Лекция 14.

Программное обеспечение процессов обработки и испытаний изделий и систем.

Самостоятельная работа студента: 2 часа.

Лекция 15.

Информационная и техническая поддержка этапов жизненного цикла изделий и систем. Предпосылки и причины инновации CALS – технологий. Интеграция данных при автоматизированной разработке изделий и систем. Глобальная стандартизация.

Самостоятельная работа студента: 2 часа.

Лекция 16.

Лингвистическое обеспечение CALS – технологий. Состав лингвистического обеспечения. Язык EXPRESS. Примеры моделей.

Самостоятельная работа студента: 2 часа.

Лекция 17.

Системные среды автоматизированных систем. Назначение системных сред. Системы управления базами данных и знаний.

Самостоятельная работа студента: 2 часа.

Лекция 18.

Интеллектуализация процесса поддержки принятия решения. Программное обеспечения CALS – технологий. Защита информации в корпоративных сетях.

Самостоятельная работа студента: 2 часа.

Темы практических занятий:

Практическое занятие 1. – 3 часа

Освоение и исследование функций CAE – систем, проведение имитационного моделирования и определение облика изделия на основе моделей систем массового обслуживания и систем Петри.

Практическое занятие 2. – 3 часа

Моделирование физических величин внешних и внутренних воздействий.

Практическое занятие 3. – 3 часа

Оценка состояний моделируемых изделий и систем, динамических процессов в них на уровне концептуальной модели.

Практическое занятие 4. – 3 часа

Освоение и использование функций CAD – систем для двумерного и трехмерного проектирования. Освоение системы “Компас”.

Практическое занятие 5. – 3 часа

Освоение методов разработки моделей сборки с использованием базы знаний.

Практическое занятие 6. – 3 часа

Освоение и исследование основных функций CAM–систем при разработке технологических процессов.

Практическое занятие 7. – 3 часа

Синтез управляющих программ для станков с ЧПУ.

Практическое занятие 8. – 3 часа

Исследование комплексирования функций CAE/CAD/CAM – систем.

Практическое занятие 9. – 3 часа

Исследование особенностей информационных потоков для реализации CALS – технологий.

Практическое занятие 10. – 3 часа

Изучение системы PLM, как системы, поддерживающей жизненный цикл изделий и систем.

Курсовой проект – 6 часов

Темы курсовых проектов:

Тема 1. Разработка проблемы обеспечения и единообразного описания и интерпретации данных в распределенных системах при использовании CALS – технологий.

Тема 2. Расчет и оптимизация размера окна столкновения в информационной сети с одним сегментом кабеля.

Тема 3. Расчет задержек при передаче сигналов в спутниковых системах связи.

Тема 4. Разработка базы знаний экспертной системы при принятии проектных решений.

Тема 5. Разработка метода оптимизации изделия на основе теории дифференциальной эволюции.

Тема 6. Разработка имитационной модели автоматизированной системы управления с использованием CAE – систем.

Тема 7. Функционирование и исследование протокола при взаимосвязи CAE и CAM – систем.

Тема 8. Расчет оптимальных параметров изделия на основе CAM – систем.

Тема 9. Исследование и формирование потоков информации в автоматизированных системах проектирования на основе CALS – технологий.

Тема 10. Оптимизация проектного решения на базе метода ветвей и границ.

Тема 11.Разработка тестов для проверки САПР на информационную безопасность.

Тема 12. Расчет оптимального коэффициента координации в двухуровневой системе.

Тема 13. Разработать оптимальный план и программу испытаний системы управления динамического объекта.

Тема 14. Разработать макет изделия с помощью системы “Компас” в трехмерной графике.

Тема 15. Разработка структуры и проектов информационной сети при организации CALS – технологий.

Описание системы контроля знаний

Общие правила выполнения контрольных заданий.

В курсе “Применение CALS – технологий для повышения качества изделий” предусматривается цикл лекций, практические занятия с использованием системы автоматизированного проектирования и

курсовой проект. По материалам лекций и практических занятий предусматривается три контрольные работы.

В систему контроля знаний входит: контроль посещения лекций, контроль выполнения практических работ, контроль поэтапного выполнения курсовой работы.

Промежуточная аттестация студентов проводится в конце каждого месяца, и результаты размещаются на учебном портале.

Правила выполнения письменных работ (курсовых, лабораторных):

Список тем курсовых работ предлагается студентам в начале учебного семестра. Студент вправе выбрать тему из данного списка или предложить свою (согласовав с преподавателем). Курсовая работа выполняется и сдается в срок, указанный в календарном плане.

Требования к оформлению работ: полуторный интервал, кегль — 13, цитирование и сноски в соответствии с принятыми стандартами, выверенность грамматики, орфографии, синтаксиса.

Курсовая работа должна содержать обзорную часть проблемы, иметь четкую постановку задачи, содержать теоретические исследования проблемы и материалы математического моделирования.

Текст отчета о лабораторной работе должен содержать краткую теоретическую и развернутую практическую части, с подробными комментариями ко всем этапам моделирования, объем не менее 4—6 страниц.

Балльная структура оценки:

Посещение лекций: 0- 20 баллов

Практические занятия (лабораторные работы): 0-35 баллов

Курсовая работа: 0 - 25 баллов

Итоговое испытание: 0 - 20 баллов

Всего - 100 баллов

Шкала оценок:

Баллы за семестр	Автоматическая оценка
91-100	5
76-90	4
56-75	3
35-55	-
<35	-

Студенты, получившие положительные оценки по результатам работы в семестре, но претендующие на получение более высокой оценки, могут участвовать в сдаче экзаменов в период сессии. Количество баллов за экзамен от 0 до 25 баллов.

Студенты, набравшие в течение семестра 35-55 баллов, обязаны пройти итоговую семестровую аттестацию в установленном порядке.

Студенты, не выполнившие программу изучаемой дисциплины и не набравшие 35 баллов, не допускаются до прохождения итоговой семестровой аттестации.

Академическая этика

В содержание курса включаются оригинальные научные и практические разработки автора УМК. Привлеченные материалы в содержание курса будут иметь ссылку на соответствующие источники.

Программа курса УМК

“Применение CALS – технологий

для повышения качества изделий”

Лекции – 36 часов – 1 зачетная единица

Раздел I.

Введение в автоматизированное проектирование и CALS – технологии. В разделе дается понятие процесса разработки изделий и систем, начиная от концептуальной модели, далее этапы формирования технического задания и исходных данных для проектирования, эскизно-технического и рабочего проектирования, создания основного образца, испытаний и отработки, сдачи заказчику, серийной эксплуатации и модификации.

Рассматривается проблема обеспечения информационной и технической поддержки изделий и систем на всем жизненном цикле. Вводится понятие CALS – технологий. Показывается, что системы автоматизированного проектирования (САПР) являются интеллектуальными системами, целью которых является разработка проекта и создание образца изделий или систем.

Рассматриваются системы автоматизированного управления технологическими процессами и определяется их связь и взаимодействие с САПР на всех стадиях жизненного цикла изделий и систем.

Раздел II.

Математическое обеспечение информационного процесса разработки изделий и систем.

Рассматривается математическое обеспечение информационного процесса разработки на концептуальном уровне. Рассматриваются такие задачи нелинейной оптимизации структуры и параметров модели на основе применения методов дифференциальной

эволюции, генетических алгоритмов, методе ветвей и границ, а также методы многокритериальной оптимизации в условиях противоречивых требований к изделию или системе.

Обсуждается математическое обеспечение подсистем машинной графики, включая трехмерную графику, построения геометрических и поверхностных моделей. Рассматривается также применение имитационного моделирования изделий и систем на концептуальном уровне.

Раздел III.

Математическое обеспечение анализа и синтеза проектных решений. Показано, каким образом можно формально описать возможные внутренние и внешние воздействия, знание которых необходимо при проектировании. Рассматривается структурный анализ и синтез, процедуры анализа и синтеза проектных решений и задача принятия решений в проектировании. Даны элементы теории интеллектуальных систем. Показаны способы формирования базы знаний и данных в экспертных системах. Даны элементы теории сложности. Рассматриваются стандартные программные системы, позволяющие автоматизировать процесс проектирования.

Раздел IV.

Технические средства комплексов и систем автоматизированного проектирования. Показывается общая структура технического обеспечения САПР. Обсуждается пространственно-временное распределение комплексов автоматизированного проектирования, включая реализацию CALS – технологий. Приведены типы вычислительных средств в составе САПР. Рассмотрены корпоративные сети, проблемы глобальной стабилизации, локальные вычислительные сети, стеки протоколов и типы сетей.

Раздел V.

Программное обеспечение автоматизированных систем. Рассматриваются программные системы CAE/CAD/CAM, их функциональные возможности и сфера применений. Показаны основы организации работы с программными системами.

Раздел VI.

Информационная и техническая поддержка этапов жизненного цикла изделий и систем. Предпосылки и причины появления CALS – технологий. Рассматривается интеграция данных при автоматизированной разработке изделий и систем. Рассматривается также лингвистическое обеспечение CALS – технологий, его состав и язык EXPRESS. Показываются системные среды автоматизированных систем и их назначение. Рассматриваются системы управления базами данных и знаний. Рассматривается также интеллектуализация процесса поддержки принятия решений и программное обеспечение CALS – технологий. Показаны возможности защиты информации в корпоративных сетях. Рассмотрено также программное обеспечение процессов обработки и испытаний изделий и систем.

Заключение.

Список обязательной и дополнительной литературы

I.Обязательная литература.

1. Пупков К.А. “Автоматизированная разработка систем управления”, учебное пособие в 3 томах. – М.: МГТУ им. Н.Э. Баумана.

2. Норенков И.П. “Основы автоматизированного проектирования”, учебник. - М.: МГТУ им. Н.Э. Баумана, 2006.

II. Дополнительная литература.

1. Норенков И.П., Кузьмин П.К. “Информационная поддержка наукоемких изделий (CALS - технологии) ” – М.: МГТУ им. Н.Э. Баумана, 2002.

2. Черненский В.М. “Имитационное моделирование”. - М.: Высшая школа, 1990.

3. Роджерс Д., Адамс Дж, “Математические основы машинной графики.” – М.: Мир, 2001.

4. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф., “Защита информации в компьютерных системах и сетях”. - М.: Радио и связь, 2001.

5. Пупков К.А., Феоктистов В.А., “Исследование проблемы нелинейной оптимизации в задачах технического проектирования”, Вестник МГТУ им. Н.Э. Баумана, сер. “Приборостроение”, 3(56), 2004, - с. 115-127.

Темы рефератов.

1. Анализ и оценка современного состояния CALS – технологий. Перспективы применения.
2. Анализ проблем стандартизации программного обеспечения в корпоративных автоматизированных системах.
3. Анализ причин утечки информации в автоматизированных системах и способы защиты информации.
4. Интеллектуализация процессов проектирования.
5. Сравнительный анализ состава без знаний в системах, основанных на CALS – технологиях.
6. Анализ современных языков программирования и программных систем, используемых в САПР.

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
КАФЕДРА КИБЕРНЕТИКИ И МЕХАТРОНИКИ

КАЛЕНДАРНЫЙ ПЛАН

учебных занятий по дисциплине Применение CALS-технологий для повышения качества изделий
Индекс специальности 550200 группы ИУБ-401, 402, 403 курс 4, семестр 7 200 _/200_ уч. года

д.т.н., проф. Пупков К.А.
(звание, степень, фамилия и инициалы ведущего дисциплину)

Число недель 18
Лекций 36 час
Практ. занятий 30 час
Курс. проект 6 час
Экзамены час
Зачеты час
Консульт. час
Всего 72 часа

Недели	Лекции	Число часов	Лабораторные / практические занятия, срок выполнения	Число часов
1 неделя	Введение. Понятие процесса разработки изделий и систем. Моделирование и автоматизация проектирования. Системы автоматизированного проектирования – интеллектуальные системы.	2	Освоение и исследование функций CAE – систем, проведение имитационного моделирования и определение облика изделия на основе моделей систем массового обслуживания и систем Петри.	3
2 неделя	Структура процесса разработки изделий и систем. Стадии разработки. Понятие конструктивной модели изделия или системы. Формирование технического задания и исходных данных. Классификация изделий и параметров, используемых при проектировании.	2		

3 неделя	Типы автоматизированных систем управления и их связь с системами автоматизированного проектирования. Этапы жизненного цикла произведенных изделий и систем. Структура САПР и разновидности САПР. CALS – технологии - основа для поддержки информационного и технического обеспечения изделий и систем в течение жизненного цикла.	2	Моделирование физических величин внешних и внутренних воздействий.	3
4 неделя	Математическое обеспечение информационного процесса разработки изделий и систем. Математическое обеспечение анализа и принятия решений на концептуальном уровне. Модели линейной и нелинейной оптимизации. Имитационное моделирование.	2		
5 неделя	Математические модели изделий и систем на этапе эскизного проектирования. Параметрическая и структурная оптимизация. Модели многокритериальной динамической оптимизации. Краткое описание языка GPSS.	2	Оценка состояний моделируемых изделий и систем, динамических процессов в них на уровне концептуальной модели.	3
6 неделя	Математическое обеспечение подсистем машинной графики. Трехмерная графика. Построение геометрических моделей. Поверхностная модель.	2		
7 неделя	Математическое обеспечение анализа и синтеза проектных решений. Формирование базы знаний интеллектуальной системы принятия проектных решений. Формирование структуры и уровней внутренних и внешних воздействий для исследования при проектировании.	2	Освоение и использование функций CAD – систем для двумерного и трехмерного проектирования. Освоение системы “Компас”.	3

8 неделя	Структурный синтез в задачах проектирования. Процедура синтеза проектных решений. Задача принятия решений.	2		
9 неделя	Методы структурного синтеза в системах автоматизированного проектирования. Элементы теории интеллектуальных систем. Элементы теории сложности. Нелинейная оптимизация. Метод ветвей и границ. Эволюционные алгоритмы. Генетические алгоритмы.	2	Освоение методов разработки моделей сборки с использованием базы знаний.	3
10 неделя	Технические средства систем автоматизированного проектирования. Структура технического обеспечения. Пространственно распределенные системы автоматизированного проектирования. Проблемы взаимодействия САПР различных компонентов изделий и систем.	2		
11 неделя	Типы вычислительных средств в составе САПР. Автоматизированные рабочие места.	2	Освоение и исследование основных функций CAM–систем при разработке технологических процессов.	3
12 неделя	Корпоративные сети. Проблемы глобальной стандартизации. Начала CALS – технологий. Локальные вычислительные сети. Стеки протоколов и типы сетей.	2		
13 неделя	Программное обеспечение автоматизированных систем. Программные системы CAE, CAD и CAM.	2	Синтез управляющих программ для станков с ЧПУ.	3
14 неделя	Программное обеспечение процессов обработки и испытаний изделий и систем.	2		

15 неделя	Информационная и техническая поддержка этапов жизненного цикла изделий и систем. Предпосылки и причины инновации CALS – технологий. Интеграция данных при автоматизированной разработке изделий и систем. Глобальная стандартизация.	2	Исследование комплексирования функций CAE/CAD/CAM – систем.	3
16 неделя	Лингвистическое обеспечение CALS – технологий. Состав лингвистического обеспечения. Язык EXPRESS. Примеры моделей.	2		
17 неделя	Системные среды автоматизированных систем. Назначение системных сред. Системы управления базами данных и знаний.	2	Исследование особенностей информационных потоков для реализации CALS – технологий.	3
18 неделя	Интеллектуализация процесса поддержки принятия решения. Программное обеспечения CALS – технологий. Защита информации в корпоративных сетях.	2	Изучение системы PLM, как системы, поддерживающей жизненный цикл изделий и систем.	3

Ведущий дисциплину: _____/ К.А. Пупков

Декан инженерного факультета: _____ / Н.К. Пономарев

Дата