

Представление данных КИП

Шильников П. С.

к.т.н., доцент кафедры «Компьютерные системы
автоматизации производства» (РК-9)

МГТУ им. Н. Э. Баумана

Оглавление

1. Унификация представления данных – основа обеспечения взаимодействия между программными приложениями	4
2. Моделирование	18
Контрольные вопросы.....	27
3. Стандарт ISO 10303 STEP. Общее определение и состав стандарта.....	28
Контрольные вопросы.....	32
4. Базовые инвариантные инструменты ISO 10303 STEP	33
Методы описания. Язык концептуальных схем данных EXPRESS. Общие положения.	36
Методы реализации. Символьный обменный файл STEP. Общие положения.	39
Элементы языка EXPRESS и представление соответствующих экземпляров данных в символьном обменном файле STEP.....	44
Контрольные вопросы.....	48
Контрольные вопросы.....	62
Исполняемые операторы языка EXPRESS.....	76
Спецификация интерфейса между схемами.	78
Ссылки между отдельными листами схем EXPRESS-G	82
Заголовочная секция обменного файла	83
Контрольные вопросы.....	88
5. Стандартный интерфейс доступа к данным SDAI (Standard data access interface)	88
Раннее и позднее связывания	90
Контрольные вопросы.....	93
6. Методика разработки Прикладных протоколов STEP.....	93
Контрольные вопросы.....	101
7. Основы представления данных об изделии в ISO 10303 STEP	101
8. Интегрированные информационные ресурсы STEP	105
Определение изделия в STEP	105
Организация геометрических моделей в STEP.	111
Проволочные (каркасные) модели.	112
Поверхностные модели.	112
Твердотельные модели.....	113
advanced_boundary_representation аналитическая граничная модель.....	117
faceted_boundary_representation плоскогранная граничная модель	118
wireframe_with_topology проволочная модель с топологией	119
manifold_surface_with_topology манифолд-поверхность с топологией.....	120
non_topological_surface_and_wireframe поверхности без топологии и проволочная модель.....	120

csf_model конструктивная твердотельная геометрия.....	121
Том 43 Организация связи изделия и его представлений.	121
Том 44 Состав и конфигурация изделия	122
Связь элементов технологического процесса с оборудованием, объектом воздействия, методом.....	129
Контрольные вопросы.....	132
9. Прикладные протоколы STEP	132
Некоторые Концепции Высокого уровня	138
Контрольные вопросы.....	145
10. Заключение.....	145
11. Библиография.....	147

1. Унификация представления данных – основа обеспечения взаимодействия между программными приложениями

Часто людям приходится выполнять какие-то крупные работы. Конечно, самый лучший вариант – если человек захотел построить, например, дом, самолет, ракету или подводную лодку, он знает, что ему нужно, все продумывает, все держит у себя в голове и сам все делает от начала до конца. Тогда он может обойтись и без всякого обмена информацией и без каких-либо записей о своей работе. Мы понимаем, что на практике это невозможно. Существует огромное количество задач, непосильных для одного человека, и люди должны работать вместе. Для того, чтобы работать вместе над общей задачей, необходимо обмениваться информацией.

Обмен информацией имеет большое значение. Информацией должны обмениваться люди, совместно участвующие в выполнении некоторой работы (например, строительство домов, разработка и создание каких-то изделий: автомобилей, самолетов, подводных лодок, ракет, телевизоров, компьютеров и т.д.). Так было всегда. Неудачно организованный обмен информацией может стать основной причиной срыва выполнения некоторой работы.

Из глубины веков до нас дошли сведения о том, что один крупный, как теперь принято говорить, «амбициозный» строительный проект не удался именно из-за того, что его участники не смогли наладить в ходе совместной деятельности обмен информацией. Имеется в виду строительство Вавилонской башни.

Таким образом, вопрос обмена информацией возник отнюдь не в XX веке. Информационными технологиями люди занимаются давно. Задолго до появления компьютеров возникли понятия «запись», «данные», «смысл», «концепция», «отображение», но до появления компьютеров, поскольку большинству людей обычно удавалось обмениваться информацией, не задумываясь о том, как они это делают, все понятия, связанные с информационными технологиями, были абстракциями, интересными только узкому кругу профессиональных философов, математиков и лингвистов.

Возможны два сценария обмена информацией (независимо от того, используется при этом компьютер или нет):

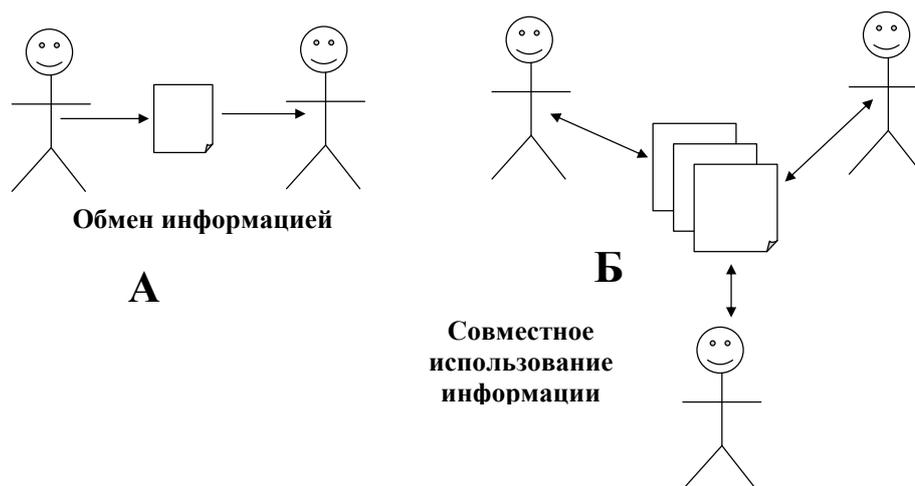


Рис. 1. Сценарии обмена информацией

Здесь все видно из схемы. В первом случае (А) каждый из участников обмена имеет свое подмножество информации. Частью своей информации он обменивается с некоторыми из партнеров, и может пополнять свою информацию информацией от других партнеров. Во втором случае (Б) все участники работы пользуются общим хранилищем информации. Можно два этих варианта проанализировать, выявить преимущества и недостатки каждого из них, сравнить. То, о чем мы говорили, касается обмена информацией вообще (независимо от сценария).

С появлением и с широким внедрением компьютеров все прошлые наработки по информационным технологиям приобрели практическое значение. Поэтому уточним: в данном материале рассматриваются информационные технологии применительно к компьютерам.

Начиная примерно с 60-х годов XX века, компьютеры все больше и больше используются в разных областях человеческой деятельности. Работа многих специалистов в настоящее время немыслима без применения компьютеров.

Применение компьютерных программ для решения самых разных производственных задач позволяет значительно повысить эффективность труда тех, кто решает эти задачи. Еще в середине 80-х годов в нашей стране такое повышение эффективности было сомнительным. Достаточно привести простой расчет: хороший, достаточно производительный компьютер стоил около миллиона рублей, хороший, достаточно квалифицированный инженер имел зарплату около 250 рублей, и был счастлив от такой зарплаты. Следовательно, выгоднее было вместо покупки одного компьютера нанять отдел инженеров, которые вполне могли справиться с задачами, решаемыми компьютером. Конечно, за исключением задач типа расчета по методу конечных элементов, которые без компьютера не решаемы в принципе.

Сейчас эти сомнения исчезли сами собой, и тот же хороший, достаточно квалифицированный инженер может купить хороший, достаточно производительный компьютер за одну месячную зарплату. К тому же, возможности

такого компьютера неизмеримо больше, чем у компьютера 80-х годов. Казалось бы, сомнения и препятствия исчезли, можно покупать все новые и новые компьютеры и бесконечно повышать эффективность решения производственных задач.

На самом деле реальная польза от компьютерной автоматизации не столь впечатляюща. Очевидно, массовое внедрение компьютеров, которое должно было бы повысить эффективность работы, привело к появлению новых проблем, которые препятствуют этому повышению эффективности. Для того, чтобы устранить эти проблемы, нужно их выявить. Для того, чтобы выявить эти проблемы, рассмотрим, чем работа специалиста с компьютером отличается от работы специалиста без компьютера.

Принято все, что относится к компьютерной технике, делить на аппаратное обеспечение (*hardware*) и программное обеспечение (*software*). При рассмотрении особенностей решения задач деловой деятельности основное внимание будет уделяться программной части инструментов, необходимых для решения данной задачи. В большинстве случаев аппаратное обеспечение рассматривается только с точки зрения повышения стоимости решения задачи, т.к. его надежность принимается за 100%, что, в принципе, на сегодняшний день недалеко от истины.

В той области человеческой деятельности, к которой относится рассматриваемый материал, большинство терминов, особенно – на русском языке, не устоялось. Многие русские термины имеют по два и более значений. У термина «электронный документ» есть, по меньшей мере, два значения:

1. Информационный объект, который с помощью некоторых аппаратных и/или программных средств доступен, без преобразования, для восприятия человеком,
2. Информационный объект, который в ходе процессов деловой деятельности проходит некоторое регламентированное множество состояний.

В таблице под «электронным документом» понимается значение 2.

Задача		Без компьютеров	С компьютерами
Взаимодействие	Доступность	Два исполнителя, говорящих на одном языке, всегда могут обмениваться информацией, даже если они работают в совершенно разных областях.	Для того, чтобы два компьютерных приложения могли обмениваться информацией, необходимо это специально обеспечить. Доходит до того, что возникают трудности при обмене информацией между разными версиями одного приложения.
	Надежность	Зависит только от квалификации взаимодействующих сотрудников	Зависит как от квалификации сотрудников, так и от надежности программного обеспечения. Практически надежность передачи данных между различающимися программными продуктами составляет 90-95%.
	Ресурсоемкость	Затраты труда. Кроме того, для передачи требуется копировальная техника. При достаточно больших объемах информации объем физических (бумажных) носителей может быть достаточно велик.	Затраты труда + стоимость программного обеспечения + стоимость аппаратного обеспечения. Затраты на саму передачу, осуществляемую по компьютерным сетям или на компактных носителях, незначительны.
Утверждение	Доступность	Для утверждения (в частности, - сертификации) необходимо предоставить комплект документации на бумаге, который всегда может быть просмотрен и понят утверждающим лицом (организацией).	Для возможности просмотра документа при утверждении необходимо программное обеспечение или идентичное тому, в котором документ был создан, или совместимое с ним

Задача		Без компьютеров	С компьютерами
	Надежность	Зависит только от квалификации утверждающего лица	Зависит как от квалификации утверждающего лица, так и от надежности программного обеспечения. Практически надежность передачи данных между различающимися программными продуктами составляет 90-95%.
	Ресурсо-емкость	Только затраты труда	Затраты труда + стоимость программного обеспечения (в т.ч. обеспечивающее ЭЦП – электронную цифровую подпись) + стоимость аппаратного обеспечения.
Хранение	Доступность	Любой документ, если обеспечена его физическая сохранность (см. пункт «Надежность») может быть извлечен из архива и использован.	В связи с постоянными обновлениями: - аппаратных платформ; - программных платформ; - программных приложений обеспечение доступности требует особых усилий
	Надежность	Если приняты меры по защите от влаги и солнца, мышей (поедающих бумагу) и мух (поедающих сухой остаток туши), надежность хранения – 100% на протяжении сотен лет.	Единственный случай из рассматриваемых, когда играет роль надежность аппаратных средств и физических носителей.
	Ресурсо-емкость	Требуются большие помещения, в которых обеспечиваются необходимые для хранения условия.	Затраты труда + стоимость программного обеспечения + стоимость аппаратного обеспечения.

Задача		Без компьютеров	С компьютерами
Создание документов	Доступность	Любой документ, если обеспечена его физическая сохранность (см. пункт «Надежность») может быть извлечен из архива и скопирован.	В связи с постоянными обновлениями: - аппаратных платформ; - программных платформ; - программных приложений обеспечение доступности документа для его воспроизводства требует особых усилий
	Ресурсо-емкость	Затраты труда + стоимость (копировальной) аппаратуры + расходные материалы	Затраты труда + стоимость программного обеспечения + стоимость аппаратного обеспечения.
Получение справочной информации	Ресурсоемкость	В значительной степени зависит от организации учета и каталогизации документов.	Затраты труда + стоимость (поискового) программного обеспечения + стоимость аппаратного обеспечения.
Внесение изменений	Ресурсо-емкость		Затраты труда + стоимость программного обеспечения + стоимость аппаратного обеспечения.
Проверка	Надежность	Зависит только от квалификации проверяющего лица	Зависит от квалификации проверяющего лица, но в гораздо большей степени зависит от специализированных программных инструментов проверки – «чекеров»
	Ресурсо-емкость	Только затраты труда	Затраты труда + стоимость программного обеспечения (специализированных инструментов проверки - «чекеров») + стоимость аппаратного обеспечения.

Как видно из приведенного краткого анализа, многие задачи которые при прежней, докомпьютерной технологии решались человеком без каких-либо дополнительных инструментов, теперь для своего решения требуют специализированных программных и технических средств.

Экземпляр информационного изделия может иметь следующие формы, различающиеся средствами, необходимыми для извлечения информации из изделия (интерпретации):

	Форма	Средства интерпретации
1	Традиционная бумажная	Не требуется, понимается непосредственно человеком
2	Медиа (микрофильмы, пластины, магнитные ленты, аудио- и видеодиски)	Требуются устройства воспроизведения
3	Компьютерная	Требуются компьютеры и соответствующее программное обеспечение

Следовательно, существует необходимость обеспечить обмен информацией не только между людьми, но и между компьютерами, с помощью которых эти люди работают, а точнее – между используемыми людьми компьютерными приложениями.

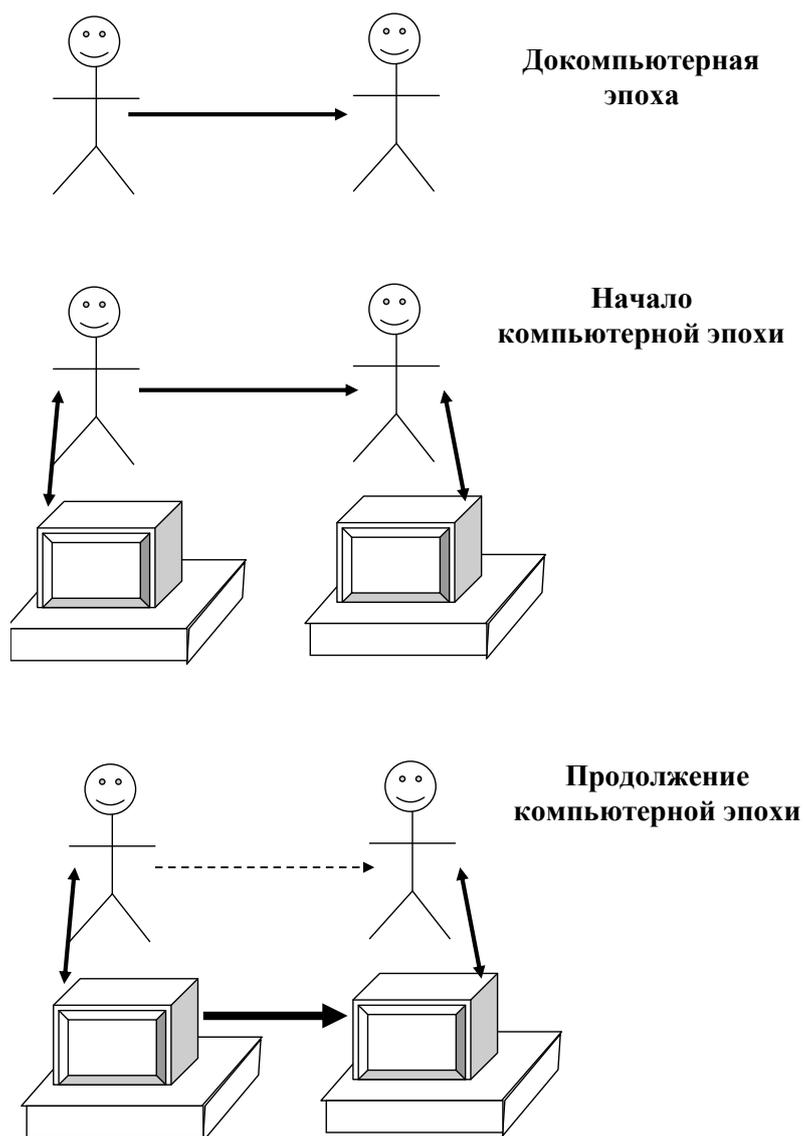


Рис. 2. Изменения принципов обмена информацией, связанные с наступлением компьютерной эпохи

Обеспечить обмен данными между компьютерными приложениями сложнее, чем между людьми. Если человек знает язык, на котором с ним общаются (устно или письменно) и разбирается в своей работе, он в состоянии воспринимать практически любой текст. С компьютерным приложением гораздо сложнее. Компьютерные приложения, в отличие от людей, могут воспринимать только очень формализованные данные.

Представим, например, работу конструктора, разрабатывающего соединительный шов (ряд заклепок, соединяющих пакет листовых деталей). Если конструктор, готовя спецификацию, для заклепки в графе «количество» вместо «4500» запишет «4500?», то любым другим человеком (другим конструктором, технологом, плановиком, диспетчером и т.д.) эта запись будет принята вполне адекватно. Человек поймет, что данное количество, возможно, будет меняться.

Совершенно по-другому воспримет эту запись компьютерная программа (компьютерное приложение), если при ее написании не предусматривалось появление в графе «количество» вопросительного знака:

- в наилучшем случае программа обнаружит нарушение синтаксиса и обратит на это место наше внимание;
- в лучшем случае программа проигнорирует неожиданный символ и воспримет правильное значение: 4500;
- в худшем случае программа «упадет» или «зависнет»;
- в наихудшем случае программа попытается прочитать и обработать непонятный символ как число, получит некоторое безумное значение, например, «45 млн заклепок» и продолжит с ним работать. Вообразите, что произойдет, если это значение впоследствии будет использовано при подготовке задания для вспомогательного производства или при формировании заявки на закупку комплектующих!!!

Таким образом, компьютерное приложение способно прочесть и интерпретировать (т.е. понять и обработать) данные только в том случае, когда приложение понимает формат данных (физический уровень представления данных), структуру данных (логический уровень представления данных) и смысл данных (концептуальный уровень представления данных).

Способность программных приложений к взаимодействию называется интероперабельностью. Уровни интероперабельности программных приложений – это интероперабельность аппаратной части, интероперабельность программной части и интероперабельность данных. Самой сложной, самой универсальной и самой актуальной является интероперабельность данных. Интероперабельность данных двух приложений можно определить как способность приложения работать с данными другого приложения.

Интероперабельность тесно связана с понятием интеграции, т.е. можно сказать, что интеграция – это технология достижения интероперабельности.

К методам интеграции данных относятся [4]:

Инкапсуляция – при этом методе PDM-система распознает файлы программного приложения и может запустить соответствующее приложение;

Интеграция данных на уровне инкапсуляции осуществляется с помощью прикладных программных систем, относящихся к классу PDM (Product Data Management). Система PDM – это инструментальное средство, которое помогает участвующим в разработке изделия исполнителям управлять как данными об изделии, так и процессом разработки, производства, эксплуатации и утилизации изделия. Система PDM при этом контролирует информацию об изделии, о состоянии объектов данных, об утверждении вносимых изменений. Система PDM следит за большими, постоянно меняющимися, массивами данных. Системы PDM отличаются от баз данных тем, что интегрируют информацию разных форматов и типов.

Ниже приведена схема, показывающая место систем PDM в общей производственной цепочке.



Рис. 3. Взаимосвязь систем автоматизации производственных процессов

Системы PDM играют роль связующего звена между этапом инженерно-конструкторской подготовки нового изделия и системами MRP (Manufacturing Resource Planning), решающими задачи автоматизации управления финансами, складским хозяйством, снабжением и сбытом, техническим обслуживанием.

Интерфейс – PDM-система и прочие приложения могут автоматически обмениваться файлами и некоторыми метаданными (без вмешательства пользователя);

Стоимость трансляторов данных для коммерческого прикладного программного обеспечения может достигать 40% от стоимости самого программного обеспечения. При большом числе различных трансляторов данных стоимость трансляторов данных значительно превысила бы стоимость самого программного обеспечения.

Следует отметить, что интерфейс между программными приложениями в общем случае не обеспечивает 100% полноты передачи данных со 100% надежностью. Приведем результаты трансляции данных между различными форматами, публикуемые фирмой STEP Tools, inc. Служба трансляции данных существует с 1991 года, и приводятся данные, осредненные за несколько последних лет. График построен на основе данных, содержащихся в электронной газете фирмы STEP Tools, Inc. Электронная газета рассылалась по подписке.

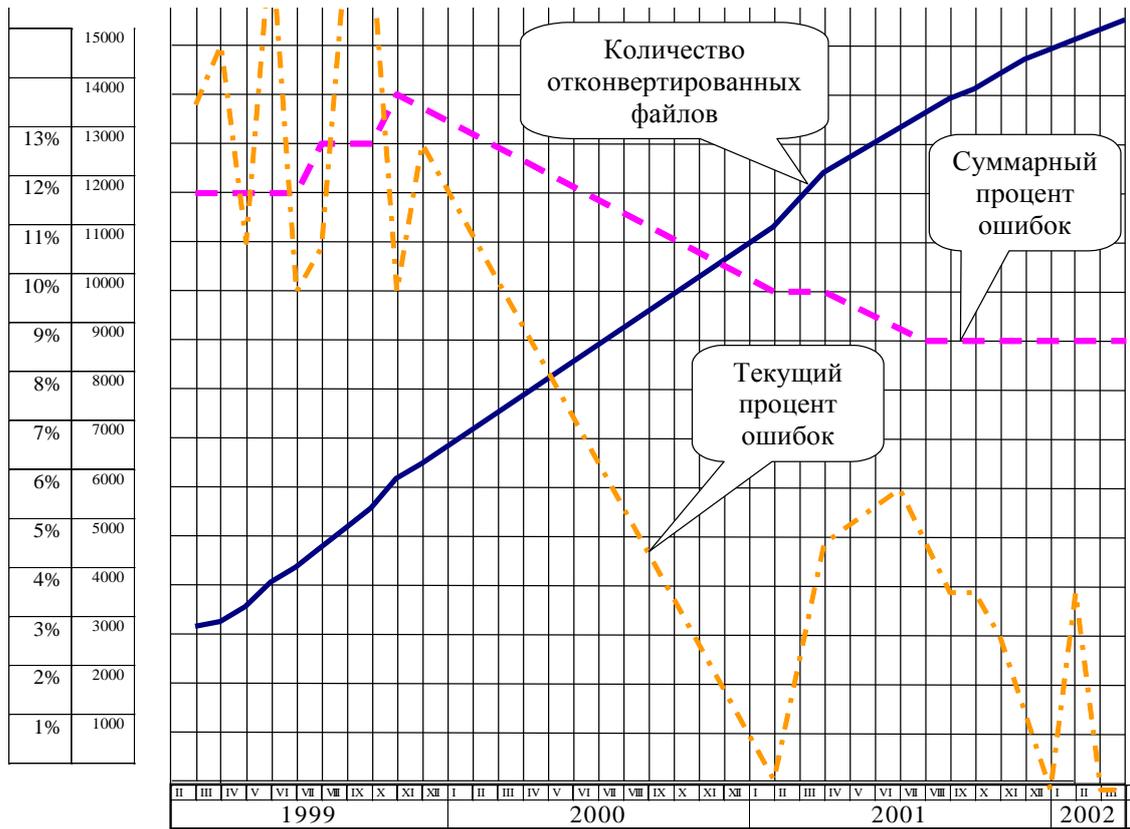


Рис. 4. Результаты конвертации файлов за 1999-2002 гг по материалам STEP Tools, Inc.

Нужно учитывать, что модель, приходящая на вход системы импорта, может быть сгенерирована любым программным комплексом с любым типом представления геометрии и требованиям к ее качеству. Кроме того, в ряде случаев стандарт STEP допускает несколько различающиеся трактовки и различные способы достижения одной цели. Например, одних способов представления сборок в процессе работы мы насчитали два основных, с под-видами, и несколько смешанных, использовавшихся не очень распространенными системами типа I-DEAS и т.п. Кроме того, разнообразие протоколов STEP вносит дополнительные требования к поддержке интероперабельности с одной стороны и полезных особенностей (таких как передача цветов в протоколе 214) с другой. Следует добавить встречающуюся на практике вовсе не гипотетическую ситуацию, когда приходящий файл вовсе некорректен или содержит некорректные данные. Даже в этом случае транслятор должен работать устойчиво, позволяя при этом пользователю получить максимум из доступной информации.

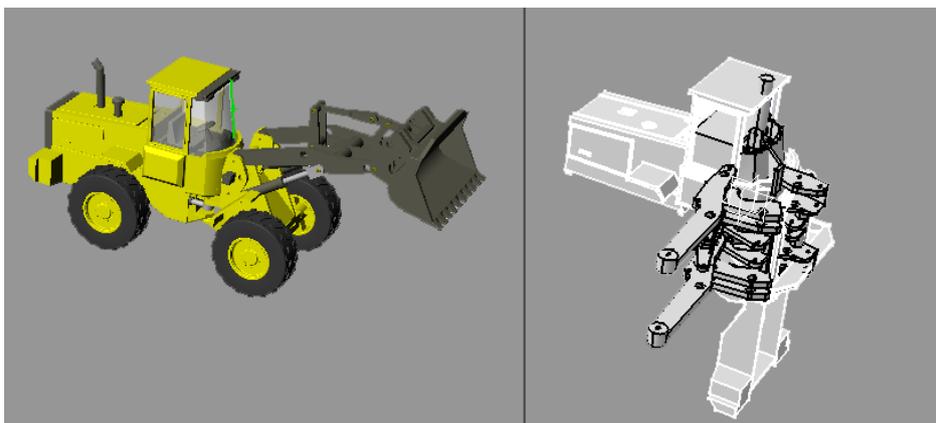


Рис. 5. Помимо проблем с геометрией очевидны проблемы с интерпретацией сборочной информации

На Рис. 5 представлены последствия импорта из одной известной системы в другую, не менее известную. Странные метаморфозы обусловлены тем, что системы использовали разные способы представления сборок и тем, что разработчики не удосужились проверить результаты работы на файлах коллег-конкурентов.

Именно на подсистему анализа и построения структуры модели возлагается достаточно нелегкая задача получения корректно построенных сборок. Особенностью реализации данной подсистемы является сложность определения возможных конфигураций представления информации. То есть ее развитие обеспечивается только огромным количеством примеров, сгенерированных разными системами различных версий, использующихся для тестирования и отладки модуля.

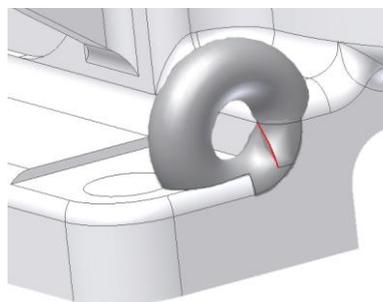


Рис. 6. Типичная ошибка - неправильная ориентация поверхности грани

Однако, наверное, самый важный аспект при обеспечении обмена с внешними программными комплексами – обеспечение корректного восстановления геометрической информации. Причем, и в процессе импорта – за это отвечает подсистема корректирования и восстановления геометрии и топологии, и в процессе экспорта – это подсистема преобразования данных для экспорта. Некорректная работа этих подсистем, как уже отмечалось в начале, может даже поставить под угрозу выполнение проекта, не говоря уже о необходимости искать обходные пути. Например, на Рис. 6 изображена «простая» ошибка. Либо при сохранении, либо при построении модели была «перепутана» ориентация грани, что привело к тому, что принимающая система неправильно построила сглаженный угол, поверхность грани для которого описывалась тором. Конечно, некоторая часть современных систем уже позволяет лечить подобные ошибки автоматически или интерактивно, но, все

же, если бы этой проблемы не возникло, пользователю было бы значительно легче.

Еще один аспект – существование стандартов качества, таких как VDA4955, AIAG-D15, JAMA, содержащих в себе ряд требований к геометрическому представлению модели. Однако, при использовании современных «готовых» геометрических ядер типа Parasolid, тела, проходящие штатные проверки, как правило, уже попадают в подмножество корректных тел по этим протоколам.

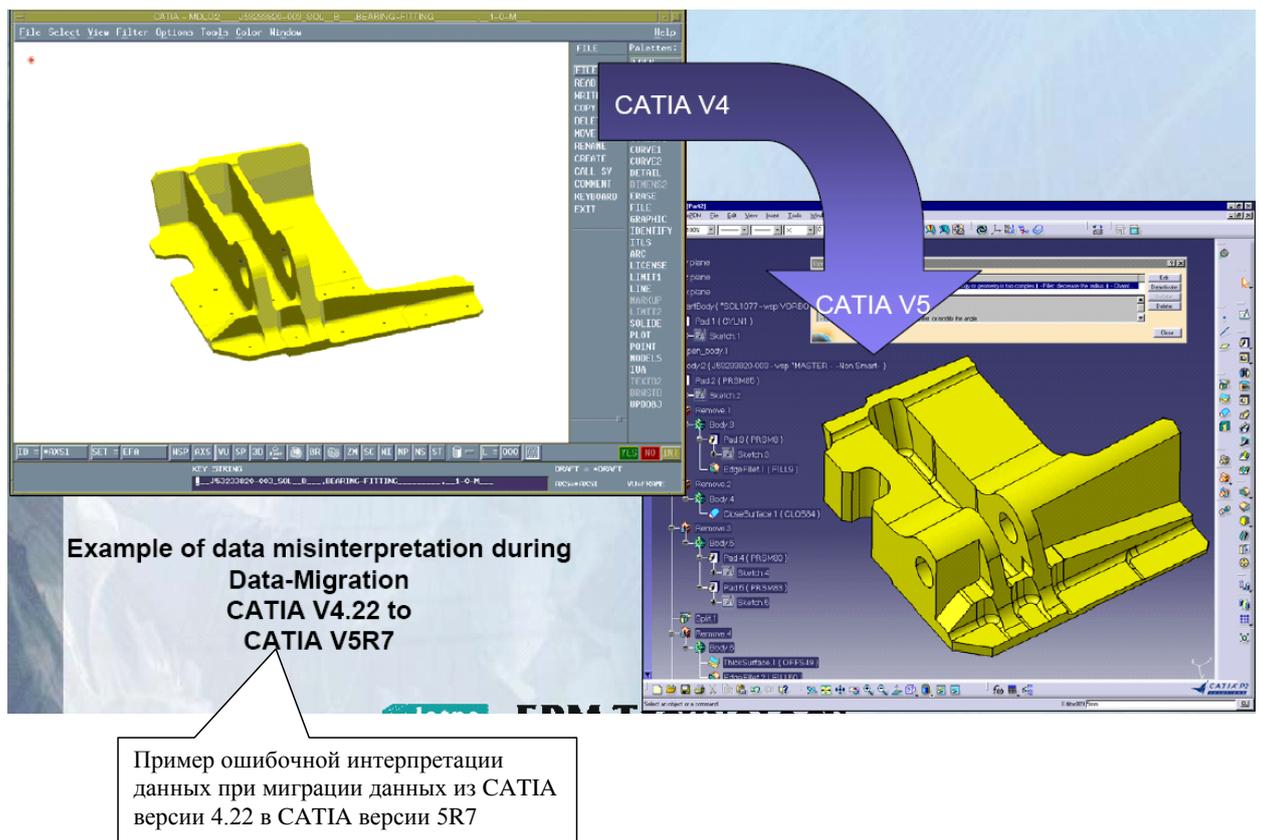


Рис. 7. Пример ошибочной интерпретации данных

Интеграция – Обеспечивается полный автоматический обмен данными об изделии всех типов и метаданными.

Используется унифицированное стабильное представление данных. Поскольку речь идет не только об обмене между множеством программных приложений в данный момент времени, но и о долгосрочном хранении, когда данные, созданные одним программным продуктом, должны через некоторое время читаться другим программным продуктом, пусть даже решающим те же самые задачи, без конвертации все равно не обойтись. Но необходимы всего две конвертации: из внутреннего формата программного приложения в нейтральный формат и наоборот. Учитывая низкую надежность процесса конвертации (90-95%), каждая лишняя конвертация значительно снижает пригодность компьютерных моделей для использования.

Существует ли возможность обойтись вообще без конвертации и добиться того, чтобы все программные приложения работали только с нейтральными форматами. Теоретически – да, но практически в обозримом будущем вопрос об этом не стоит, и здесь он рассматриваться не будет.

К интеграции данных существуют два подхода – CALS и PLM. Данные понятия часто путают, некоторые авторы даже наивно полагают, что это – одно и то же. В действительности же, несмотря на некоторое сходство, CALS и PLM – это два противоположных подхода к достижению одной цели. Цель эта состоит в полном объединении всех задач, решаемых с помощью компьютера, на всех этапах Жизненного Цикла Изделия: маркетинг, подготовка производства (проектирование, конструкторская и технологическая подготовка производства), материально-техническое снабжение, производство, контроль, упаковка и хранение, распределение, эксплуатация и утилизация (см. стандарты серии ISO 9000).

Поход PLM, суть которого ясна из приведенного рисунка, состоит в том, чтобы обеспечить решение всех задач с помощью набора взаимоувязанных программных продуктов одного крупного разработчика программного обеспечения. На Рис. 8 такие разработчики показаны в виде «акул империализма», заглатывающих всю цепочку Жизненного Цикла Изделия. И из этого же рисунка видна и основная возникающая при этом проблема. Проблема заключается в том, что пользователь привязывается к программным продуктам одного разработчика.

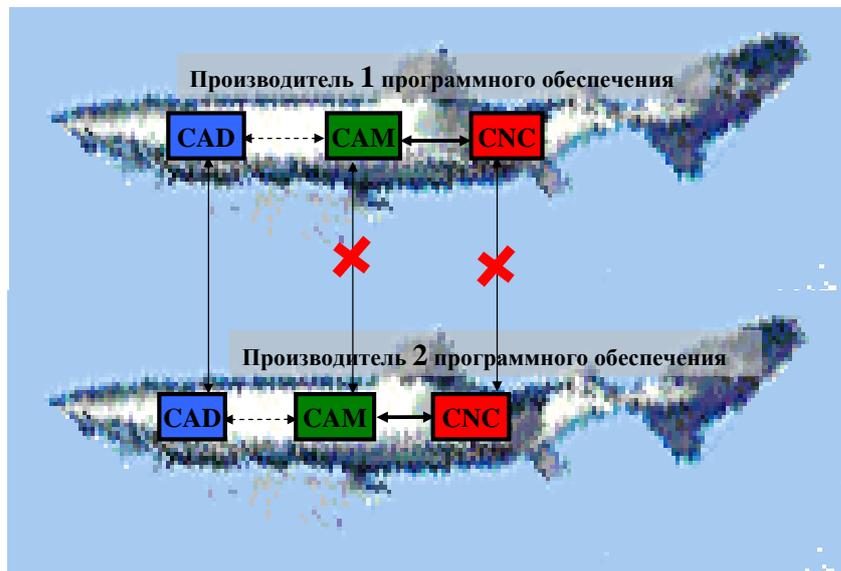


Рис. 8. Проблемы обмена данными между интегрированными системами

Подход CALS, наоборот, состоит в том, чтобы освободить пользователя от зависимости от одного разработчика. Основа подхода – это SDE, или Единое Информационное Пространство, построенное на применении Международных стандартов представления данных. Основным стандартом – это ISO 10303 STEP (STandard for Exchange of Product model data – Стандарт обмена данными модели изделия). Статус Международного стандарта обеспечивает два очень важных свойства STEP – стабильность (стандарт пересматривается раз

примерно в пять лет, и новые версии не изменяют и не отменяют, а дополняют старые) и общедоступность (необходимые для практической работы материалы по стандарту или находятся в свободном доступе в Интернете или могут быть куплены за несколько сотен долларов в официальных органах стандартизации, например, [ВНИИКИ](#)).

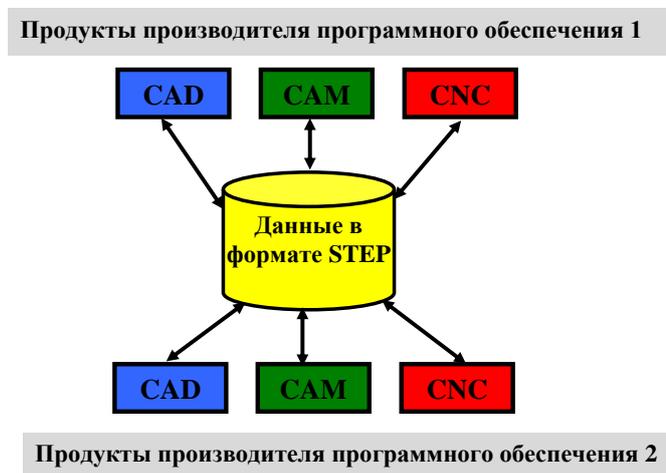


Рис. 9. Схема взаимодействия с использованием STEP

Для России, учитывая отсутствие фирм класса Siemens PLM Solutions (разработчик системы NX), Dassault Systemes (разработчик системы CATIA), и т.д., CALS является единственным приемлемым решением. К тому же, CALS для России – это в настоящее время единственный реальный путь обеспечения информационной безопасности. Если промышленные данные, включая рабочие проекты современных изделий, в том числе – оборонного назначения, хранятся во внутренних форматах программных приложений, то существует вероятность того, что при отзыве лицензий на программные приложения, возможно, по политическим мотивам, российская промышленность останется без современных разработок.

2. Моделирование

Для того чтобы перейти к более подробному изложению материала, необходимо дать определения основных понятий. Несомненно, «модель» - одно из самых важных. Существует множество определений понятия «модель». Одним из самых простых и понятных является определение, данное в [1]:

M есть модель системы S , если M может быть использована для получения ответов на вопросы относительно S с точностью A . Таким образом, целью модели является получение ответов на некоторую совокупность вопросов.

То же самое относится и к модели отдельного объекта, объект – это понятие более обобщенное, чем система. И сам моделируемый объект или система и сама модель – это объекты. Какова природа рассматриваемых объектов – модели и моделируемого объекта, вопрос достаточно сложный, касающийся-

ся соотношения идеального и материального, объективного и субъективного. Для того, чтобы утверждать, что моделируемые объекты – это обязательно объекты реального мира, надо ответить, а что же это такое – объекты реального мира. Ответы могут быть разные – от примитивного материалистического (например, реально только то, что можно пощупать) до той интерпретации, которая дается в фильме «Матрица». Формулировка точного обоснованного ответа выходит за рамки данного материала.

Начнем с более простого – с модели. Почему это проще – хотя бы потому, что мы уже имеем определение того, что такое модель: это – объект, позволяющий получать ответы на вопросы о моделируемом объекте, не взаимодействуя с самим моделируемым объектом.

Модель может быть материальным вещественным объектом. Это – компоновочные модели из дерева, пенопласта, пластилина и т.д., аэродинамические модели, модели для испытаний на прочность – для статических испытаний и динамических испытаний. В быту мы чаще всего встречаем детские игрушки, большинство из которых являются моделями. Все эти модели позволяют получить ответы на некоторые (далеко не все!!!) вопросы о моделируемом объекте. В данном материале вещественные модели не рассматриваются.

Модели могут быть невещественными. Если считать, что материя – это объективная реальность, существующая независимо от нашего представления о ней, можно предположить, что такие модели хотя не вещественны, но, тем не менее, материальны. Но вопрос о том, является ли невещественная модель материальным объектом, выходит за рамки рассмотрения данного материала.

Например, не взаимодействуя с собакой, а иногда даже и не видя ее перед собой, мы знаем, что собака может укусить. Значит, мы располагаем моделью собаки, и эта модель позволяет получить ответы на множество вопросов: от внешнего вида до особенностей поведения. Подавляющее большинство людей имеют модель собаки, но мало кто способен объяснить, как эта модель устроена. К счастью, модели подобного рода также выходят за рамки данного материала.

К невещественным моделям относятся формальные математические модели. Не взаимодействуя со стальным шариком, можно по формуле

$$h = \frac{v^2}{2g}$$

получить ответ на вопрос о том, на какую высоту поднимется шарик, если его бросить вверх с заданной скоростью.

Бывают и более сложные модели, не сводимые к одной зависимости, дающей точный ответ. Рассмотрим пример, который далее будет уточняться и использоваться для иллюстрации других положений.

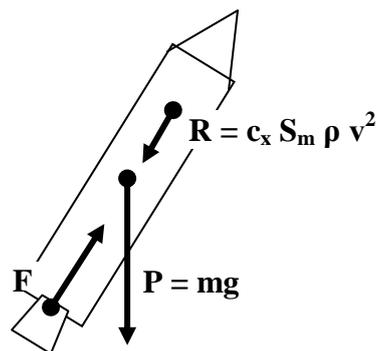


Рис. 10. Силы, действующие на одноступенчатую ракету

На одноступенчатую ракету, показанную на Рис. 10, действуют сила веса, сила тяги двигателя и сила аэродинамического сопротивления.

Рассмотрим этап определения Основных проектных параметров (ОПП), когда еще не ведется конструкторская проработка изделия. Существует некоторый набор свойств, значения которых определяют ракету: вес полезного груза, миделево сечение, коэффициент аэродинамического сопротивления, удельный импульс топливной пары, коэффициент конструктивного совершенства и т.д. Модель, содержащая значения всех этих свойств, позволяет получить ответы на ряд вопросов, например:

- какой полезный груз может быть доставлен на заданное расстояние;
 - на какое расстояние может быть доставлен заданный полезный груз;
 - какой коэффициент совершенства конструкции необходимо обеспечить для того, чтобы доставить заданный груз на заданное расстояние;
- и т.д.

Между некоторыми свойствами существуют зависимости, чаще всего – эмпирические.

И первая и вторая из рассмотренных задач могут решаться с помощью компьютерных программных приложений. Если для первой из задач можно воспользоваться стандартными средствами, например, MS Excel, то решение второй задачи требует специализированного программного приложения.

Что может быть моделируемым объектом? Несомненно, моделироваться могут реально существующие вещественные объекты. Также могут моделироваться некоторые изделия, которые еще не созданы (а, возможно, и не будут созданы). Это относится, в первую очередь, к процессу разработки изделий. Рабочий проект изделия – это модель изделия, которого еще не существует. Еще один класс моделируемых объектов – это различные схемы отношений. Такими схемами могут быть кинематические схемы механизмов, организационные структуры, принципиальные электрические схемы и т.д. Эти схемы не существуют как вещественные объекты, но, тем не менее, они объективно существуют и могут моделироваться с той или иной степенью однозначности и адекватности.

На основании вышесказанного можно сформулировать определение: моделируемый объект – это то, что моделируется или описывается т.е. *денотат*. Такое определение, несмотря на его неинформативность, точнее всего отражает все многообразие моделируемых объектов. Классификацию моделей и моделируемых объектов можно изобразить в следующем виде (Рис. 11):

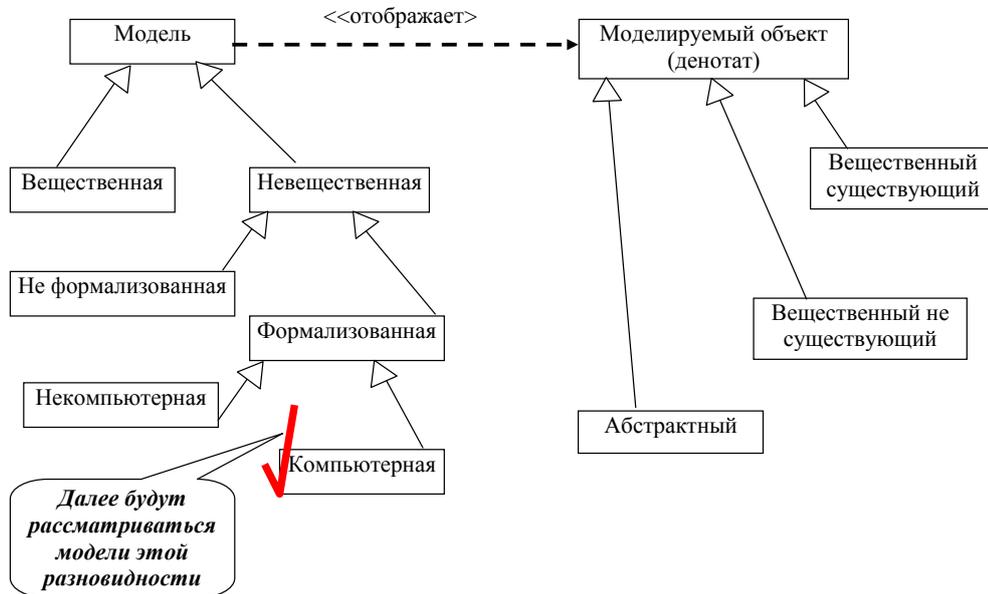


Рис. 11. Классификацию моделей и моделируемых объектов

Соотношение между денотатом и формальной знаковой моделью (Знаком) можно представить следующей схемой (Рис. 12):

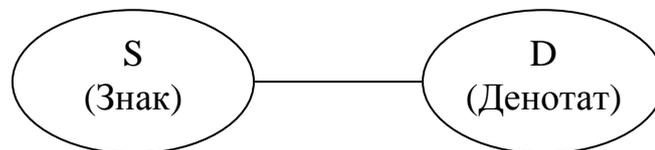


Рис. 12. Соотношение между денотатом и формальной знаковой моделью

Знак является отображением (знаковой моделью) денотата.

Остается вопрос о том, сколько свойств имеет денотат и какие свойства денотата отображает знак. О количестве свойств денотата существуют разные предположения. В некоторых философских системах утверждается, что любой объект неисчерпаем в своем познании, и сколько бы его свойств мы ни открыли, у объекта всегда остается еще некоторое число неоткрытых свойств. Более того, объект, взаимодействуя с другими объектами, постоянно развивается и обретает новые свойства. С другой же стороны, если, согласно другим философским системам, все материальное создано неким высшим существом по вполне определенному плану (замыслу), то можно предположить, что любой материальный объект имеет ровно столько свойств, сколько было заложено в вышеупомянутом замысле.

Но даже если число свойств описываемого или моделируемого объекта, т.е. объекта денотата конечно, этих свойств так много, что при современных

вычислительных ресурсах можно допустить, что число свойств **бесконечно**. Из этого и будем исходить дальше.

Особенности моделирования определяются потребностями предметной области. Существует множество работ, в которых используется термин «предметная область», но мало где можно найти определение этого термина. Одно из немногих определений дано в [3], где под предметной областью понимается «фрагмент реальной (виртуальной) действительности, представляемый некоторой совокупностью принадлежащих ему сущностей». Поскольку далее само понятие «сущность» будет выводиться на основе понятия «предметная область», для данного материала такое определение не подходит, и далее под предметной областью будет пониматься *существенная для некоторой деятельности совокупность объектов, свойств объектов и процессов, в которых эти объекты участвуют*.

Денотат может содержать множество объектов одного типа, каждый из которых обладает некоторым набором свойств. Значения части свойств могут совпадать, а другой части свойств может различаться. Должно существовать определение или идея объектов одного типа. А.Шопенгауэр в работе «Мир как воля и представление» писал: «Львы рождаются и умирают, а львиность, т.е. форма льва и идея льва существует вечно». Т.е. существует одна идея или концепция объектов некоторого типа и существует некоторое количество самих объектов, соответствующих этой идее (Рис. 1).

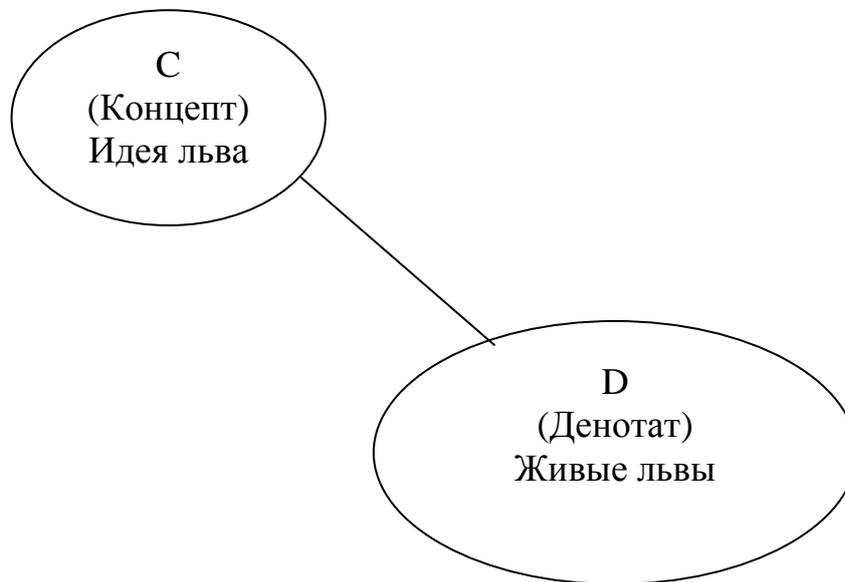


Рис. 13. Концепт и денотат

Объединив три рассмотренных элемента (концепт, знак и денотат), получаем «Концепцию смысла» Готлоба Фреге (Рис. 14).

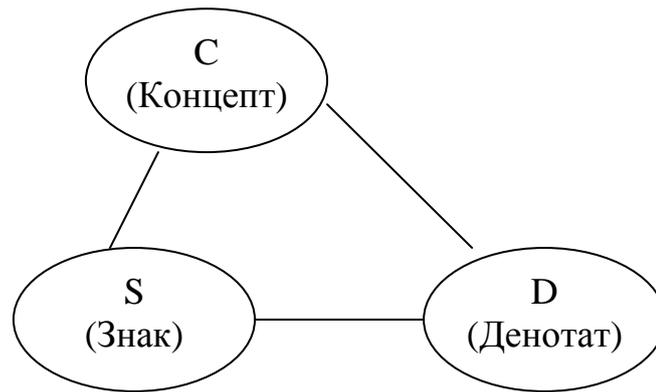
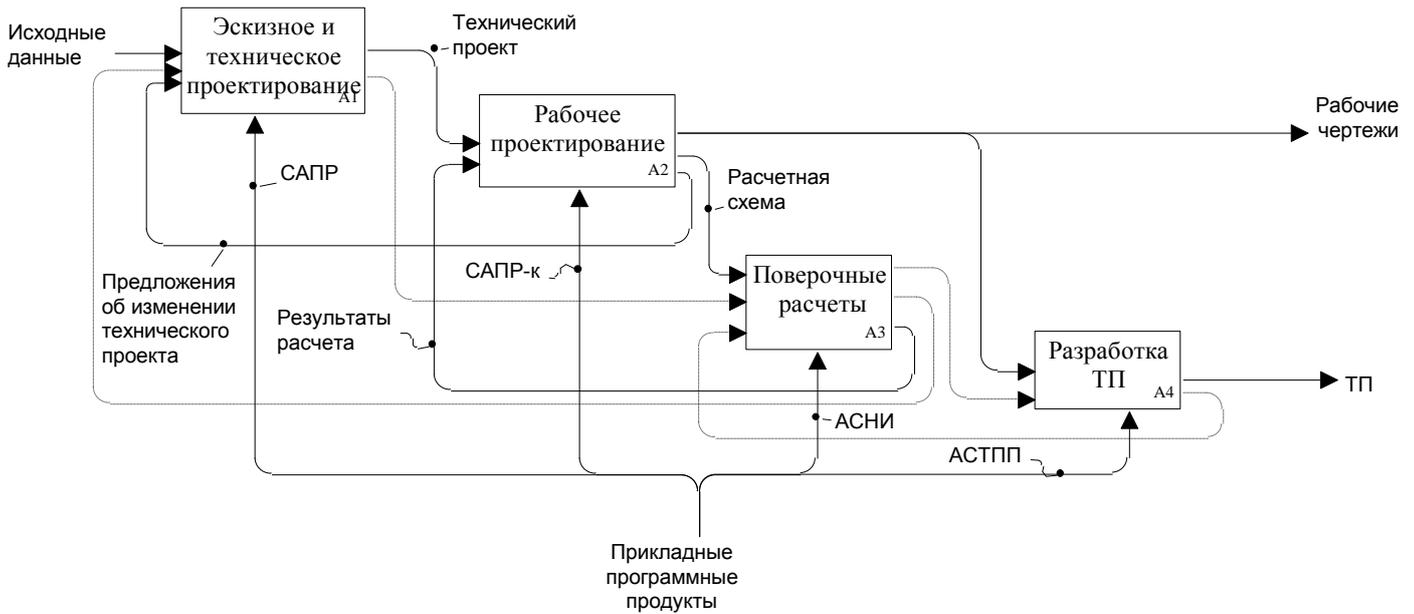


Рис. 14. «Концепция смысла» Готлоба Фреге

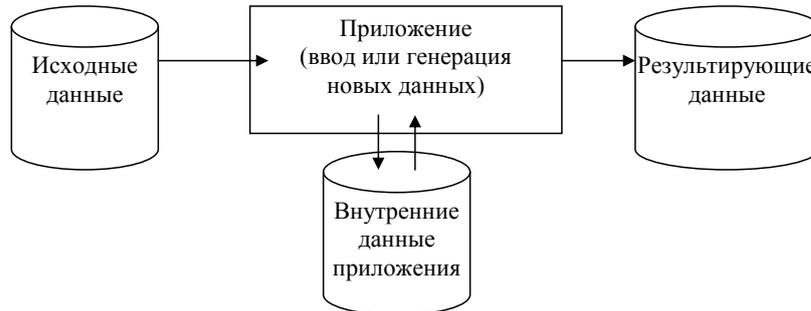
Если вернуться к нашим львам, возникает вопрос: существует ли одна идея льва, или в разных областях человеческой деятельности могут существовать различающиеся идеи, отображающие те или иные свойства одного объекта денотата, обладающего, как было решено выше, бесконечным числом свойств. Можно предположить, что у работника цирка, организатора сафари и ветеринарного врача будут разные идеи льва. Следовательно, концепт неразрывно связан с предметной областью. В качестве концепта будет рассматриваться концептуальная схема данных. Концептуальная схема данных – это описание того, какие сущности существуют в предметной области, как сущности соотносятся между собой, какими свойствами сущности обладают с точки зрения данной предметной области, и какие ограничения накладываются на значения свойств сущностей. Следовательно, концептуальная схема данных – это описание предметной области. Такое описание не является самоцелью, и должно содержать определения тех данных, с которыми работают программные приложения, используемые в данной предметной области.

Исходными данными для построения концептуальной схемы данных является функциональная модель работы приложений, обеспечивающих компьютерную автоматизацию деловой деятельности в данной предметной области. Например, рассмотрим процесс создания механических изделий – элементов ферменной конструкции.

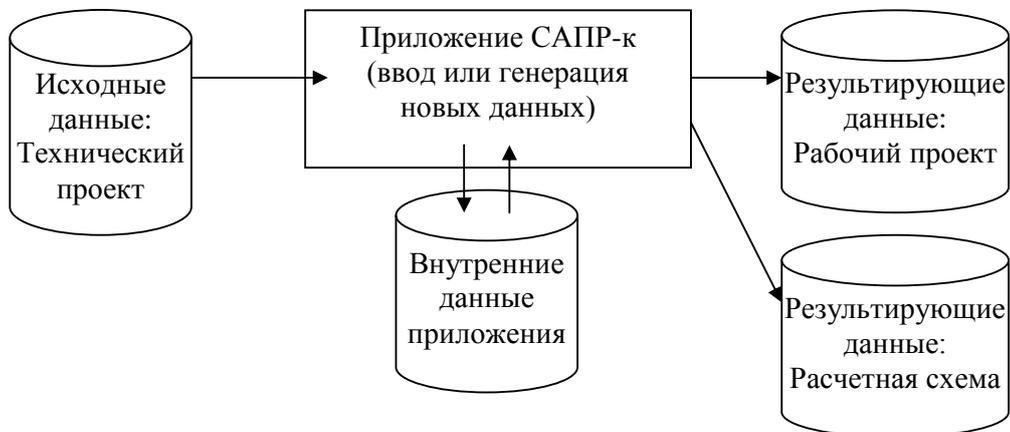
Модель представлена в виде IDEF0-диаграммы. В этой модели каждый функциональный блок соответствует определенному виду деятельности, выполняемому с помощью определенного класса приложений. Приложения на диаграмме представлены механизмами. Входами и выходами функциональных блоков являются информационные объекты, соответствующие пакетам данных, которыми обмениваются между собой приложения.



На основе полученной функциональной модели выполняется детальная проработка данных. Необходимо определить, из каких элементов данных состоит каждый из пакетов данных (информационных объектов), соответствующих стрелке на функциональной диаграмме.



Например, для осуществляемого с помощью САПР-к рабочего проектирования данная схема интерпретируется следующим образом:



Для примера рассмотрим те данные, которые используются для описания стержня ферменной конструкции.

При этом следует понимать, что реальные примеры будут иметь дело с гораздо более широкой номенклатурой изделий (например, «механические изделия» вообще, «изделия автотракторной промышленности», «трубопроводы», «здания», «электротехнические изделия» и т.д.), и в реальных примерах используются более обобщенные объекты и понятия.

В примере, для простоты и наглядности, будет рассмотрено определение только одного класса информации. В действительности, многообразие тех свойств изделия, которые должны отражаться в концептуальной схеме данных, приводит к необходимости применения большого числа классов информации, которые отображают отдельные свойства описываемого объекта.

Рассмотрим выявленные потребности в информации, описывающей единичное изделие – стержень. Жирным шрифтом здесь представлены пакеты данных, соответствующие стрелкам на IDEF0-диаграмме. Для каждого из пакетов данных раскрывается его состав.

Среди приведенного набора элементов данных под *Идентификатором* понимается внешний символьный идентификатор, принятый в разработке и производстве изделий (например, «224.00700.210-57»), называемый в информационных технологиях «внешним идентификатором», а не формальный идентификатор, обозначающий некоторый информационный объект, и имеющий смысл только внутри программного приложения. Внешний идентификатор (обозначение) обеспечивает уникальную идентификацию изделий, ресурсов, операций, свойств и т.д. в процессе деловой деятельности. Обычно для внешних идентификаторов разрабатывается и утверждается система их формирования, часто – на основе стандартов. Для программного приложения этот идентификатор чаще всего воспринимается как строка символов. Внешние идентификаторы могут служить поисковыми признаками в системах управления данными об изделии (Product data management - PDM), в каталогах, классификаторах и т.д.

Технический проект

Идентификатор (уникальный индекс) изделия;

Геометрия.Длина;

Эксплуатационная нагрузка;

Рабочий проект

Идентификатор (уникальный индекс) изделия;

Геометрия.Длина;

Геометрия.Диаметр;

Геометрия.Толщина стенки;

Материал и его свойства;

Допустимые отклонения размеров;

Серийность;

Расчетная схема

Идентификатор (уникальный индекс) изделия;

Геометрия.Длина;

Геометрия.Свойства сечения (производное от диаметра и толщины стенки);

Эксплуатационная нагрузка;

Вид опоры;

Результаты расчета

Идентификатор (уникальный индекс) изделия;

Геометрия. Длина;

Рассчитанные напряжения и частоты;

Технологический процесс

Идентификатор (уникальный индекс) изделия;

Последовательность изготовления.

Геометрия. Длина;

Геометрия. Диаметр;

Геометрия. Толщина стенки;

Материал и его свойства;

Допустимые отклонения размеров;

Серийность;

Эти элементы данных, в зависимости от их мощности (множественности), особенностей областей определения и прочих свойств, могут представляться:

- отдельными членами информационных классов. Например, идентификатор – это единичная символьная строка, и наиболее разумным было бы представить его в виде одного члена информационного класса.

- отдельными информационными классами, имеющими несколько членов. Материал, обладающий, в зависимости от потребностей данной предметной области, некоторым фиксированным набором свойств: наименование, плотность (удельный вес), прочность, теплопроводность и т.д., лучше представить в виде отдельного информационного класса, члены которого представляют эти отдельные свойства.

- совокупностью информационных классов. Последовательность изготовления, т.е. технологический процесс – это сложный информационный объект, отражающий структуру технологического процесса (декомпозицию и последовательности), режимы обработки, возможно – связь с используемым инструментом и многие другие данные, которые столь сложны и разнотипны, что их непросто представить в виде отдельного информационного класса.

Ниже в виде диаграммы классов UML показана схема соотношения между информационными классами, описывающими различные данные об изделии – полом цилиндрическом стержне на различных стадиях его разработки.

Концептуальная схема данных, представленная в формализованном виде, например, как в нотации UML, используемой в рассматриваемом примере, называется онтологией, т.е. «спецификацией концептуализации».

3. Стандарт ISO 10303 STEP. Общее определение и состав стандарта

Информация, об изделии на отдельных стадиях его жизненного цикла (маркетинг, проектирование, производство, эксплуатация и утилизация) широко используется на протяжении всего жизненного цикла с помощью различных компьютерных систем в CALS-технологиях предлагается применение интегрированной информационной модели изделия. Таким образом, возникает потребность в единой, понятной для компьютеров форме представления информации об изделии, которая должна обеспечивать организацию информационного обмена между различными компьютерными системами.

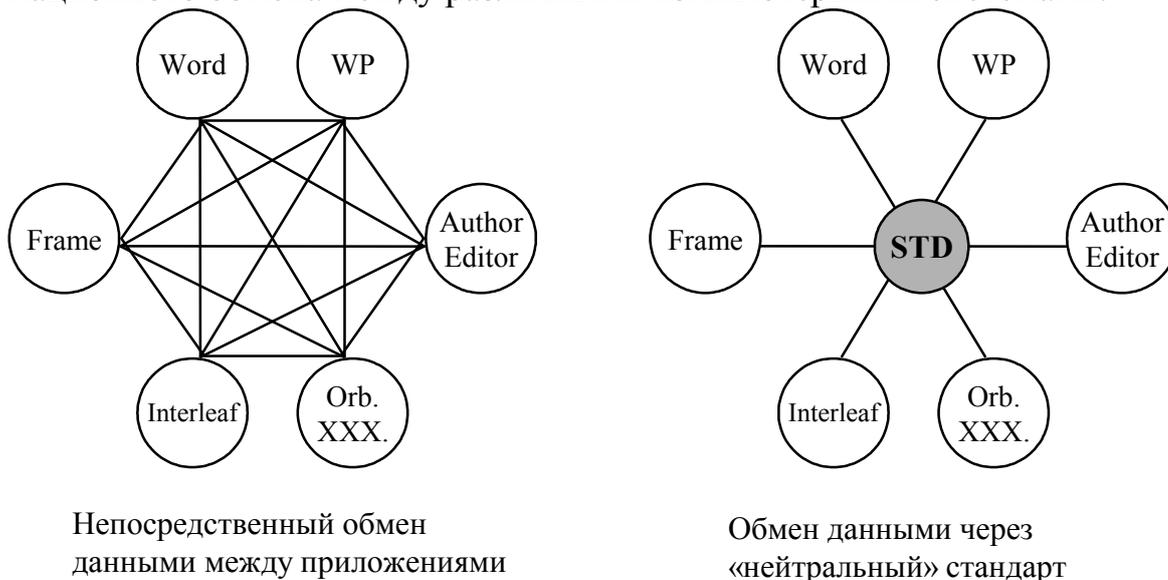


Рис. 15. Информационная среда при наличии и отсутствии нейтрального стандарта для обмена данными

Стандарт STEP задает полную информационную модель изделия на протяжении его жизненного цикла, а также способы реализации обмена данными, представленными согласно его полной модели. Как полная модель изделия, так и способы обмена данными представлены в компьютерном виде, причем они не зависят от программных и аппаратных средств, применяемых участниками ЖЦИ.

Преимущество единого стандарта состоит в возможности легко организовать информационный обмен между всеми компьютерными системами, которые используются в течение жизненного цикла изделия. В противном случае информационный обмен будет вестись между каждой парой компьютерных систем, что имеет существенные недостатки:

невозможность создания интегрированной модели изделия;

необходимость приобретения большого количества конвертеров форматов $N \cdot (N - 1)$ штук, где N – количество используемых компьютерных систем.

В случае применения стандарта STEP количество конвертеров сокращается до $2 \cdot N$ штук, где N – количество используемых компьютерных систем.

тем. Кроме того, STEP имеет статус международного стандарта, что облегчает взаимодействие с зарубежными партнерами.

В 1984 г. было признано, что быстрое увеличение нейтральных стандартов может привести к новому препятствию в эффективном обмене данными. Кроме того, разработка очередной САЕ-технологии требовало расширения существующих стандартов. Научно-исследовательские работы в США и Европе показали возможность новых решений. Широкие круги в промышленности осознали долгосрочную потребность в едином стандарте представления данных об изделии, который будет использоваться в течение полного ЖЦИ и применим для всей информации, поддерживаемой САПР.

Тогда же стали видны и значительные ограничения спецификации IGES. Наряду с глобализацией бизнеса, увеличением сложности изделий, количества поставщиков, разнообразием используемого ПО, необходимостью поддержки ЖЦ это привело к старту работ по программе PDES (Product Data Exchange Specification – спецификация обмена данными об изделии). Ее цель – имея под собой четкую методологическую и научную основы, делать основной упор на логическое представление данных об изделии и схемы обмена.

Основной задачей первого конгресса по проблеме нового стандарта обмена данными (STEP) в Вашингтоне было создание стандарта, который делает возможным получение информации, включающей обработанную на компьютере модель изделия в нейтральной форме без потери полноты и целостности в течение всего ЖЦИ.

Новые особенности предложенного стандарта включают полноту и критерии целостности, а также объем информации по всему ЖЦИ.

Даже на этом раннем этапе было признано, что стандарт должен обладать:

гибкостью для получения возможности расширения существующих частей стандарта без потери им законной силы;

эффективностью обработки, обмена и хранения;

строгой и формальной документацией;

минимально возможным множеством элементов данных;

разделением данных, зависящих от физического формата;

логической классификацией элементов данных;

совместимостью с другими существующими связанными стандартами.

Ядро модели изделия должно включать:

геометрию:

трехмерные твердотельную и поверхностную модели;

граничное представление;

конструктивную стереометрию;

допуски;

конструктивно-технологические элементы (КТЭ);

конструкторские спецификации, включая: материалы, технические требования процесса, обозначение изделия, замечания.

прикладные требования данных или прикладное влияние на ядро модели изделия следующих областей:

Состав изделия;

Моделирование конечных элементов;

Обеспечение качества;

Обработка данных;

Механизм для стандартных изделий;

Параметрические особенности проекта;

Синтаксис данных и структуру файла, независимую от вышеупомянутого содержания.

Представление состава изделия должно опираться на существующие национальные и международные стандартные требования. Где возможно, будет определена общая семантика для использования с флагом, обозначающим желательный стандарт состава для производства определенной отображаемой информации.

Предпочтительны более устойчивые геометрические формы кривых и поверхностей. Руководство по реализации должно включать конверсионные методы. Необходимы конструктивные и оценочные формы (результата) геометрических объектов».

Дата окончания разработки стандарта была назначена на конец 1985 г. Она основывалась на положительных результатах инициатив типа CAD1, PDDI, GMAP и PDES США. Если бы присутствующие на первой встрече полностью осознали величину задачи и поняли, что принимали десятилетнюю программу, возможно результат единодушного голосования четырех стран-участниц (США, Великобритания, Франция и Германия) был бы другим, но потребность в новом стандарте осталась бы бесспорной.

В том же 1984 году прошло первое заседание Подкомитета 4 Технического комитета 184 ISO (ISO Technical committee 184 Industrial automation systems and integration Subcommittee 4 Industrial data – Технический комитет 184 ИСО Промышленные системы автоматизации и интеграция Подкомитет 4 Промышленные данные) - ISO TC184/SC4, положившее начало разработке международного стандарта для обмена данными об изделии ISO 10303 STEP. Его основой стали практически все предыдущие работы: IGES, PDDI, AECMA, VDA-FS, SET, впоследствии произошла интеграция усилий с программой PDES. В процессе создания STEP большую роль сыграла международная электротехническая комиссия (IEC). Учитывая большие объемы предстоящей работы было решено разработку и принятие стандарта ISO 10303 STEP осуществлять по отдельным томам (part). В 1994 году были официально опубликованы первые тома нового международного стандарта ISO 10303. Разработка и успешное внедрение в промышленную практику стандарта ISO 10303 STEP продолжается до настоящего времени.

Структуру STEP можно условно представить схемой, состоящей из трех уровней. Первый уровень является ядром стандарта и содержит инструментарий STEP, с помощью которого задаются остальные компоненты, а также реализуется информационный обмен. На втором уровне находится ба-

зовое представление информации об изделии, инвариантное по отношению к предметной области. Оно включает базовую информационную модель изделия, заданную с помощью инструментария STEP. Наконец, третий уровень содержит представления информации об изделии, специфичные для конкретной предметной области (например, машиностроение, автомобилестроение, судостроение и т.п.). Они характеризуют информационную модель изделия для конкретной предметной области и опираются как на инструментарий STEP (первый уровень), так и на базовую модель изделия (второй уровень).



Рис. 16. Структура стандарта STEP (ISO 10303)

Ниже дана краткая характеристика пяти основных компонентов STEP.

Методы описания предназначены для всех информационных моделей в STEP. Они позволяют задать структуру данных, которой описывается изделие. Основа методов описания – язык концептуальных схем данных EXPRESS (ISO 10303-11).

Методы реализации используются для представления модели изделия в соответствии со STEP. Они позволяют организовать обмен или хранение информации, представленной с помощью методов описания STEP. Экземпляры данных могут быть представлены в символьном обменном файле STEP в соответствии с ISO 10303-21 или в некотором источнике данных (репозитории STEP) в соответствии с ISO 10303-22, чаще всего - в двоичном виде и (с некоторыми ограничениями) в виде файлов XML.

Также в томах, относящихся к методам реализации, содержится стандартизованная спецификация набора программных функций, обеспечивающих работу с данными, содержащимися в репозитории STEP. Специфицирован-

ные функции могут быть реализованы программно, и программная реализация может использоваться для разработки конверторов данных (препроцессора и постпроцессора STEP), системы управления хранилищем данных STEP, системы долгосрочного архивирования данных в формате STEP и некоторых программных приложений, например, вьюеров, навигаторов, систем управления качеством данных и систем лечения.

Особое место среди методов реализации занимает представление данных STEP в формате XML (см. ISO 10303-28). В ISO 10303-28 содержится описание принципов отображения моделей из формата STEP в формат XML. После такого отображения появляется возможность осуществлять просмотр и навигацию моделей STEP с помощью широко распространенных программных средств работы с XML. В то же время из-за того, что конструкции STEP более сложные, чем в XML, полное отображение невозможно. В некоторых случаях при отображении требуется дублирование данных, что приводит к избыточности модели.

Каждая из трех форм представления знака имеет свои преимущества и недостатки:

	Символьный обменный файл	Файл XML	Репозиторий данных STEP
Преимущества	Независимость от программной и аппаратной платформ Доступность для чтения и редактирования вручную	Доступность для программной обработки большим количеством браузеров без использования специализированных инструментов STEP.	Хорошие возможности компьютерной обработки
Недостатки	Сложность программной обработки	Неполное отображение всех особенностей моделей STEP.	Зависимость от программной и аппаратной платформ

Методология тестирования на соответствие определяет основные принципы тестирования различных программных средств на соответствие стандарту STEP.

Интегрированные ресурсы задают базовое представление информации об изделии, инвариантное по отношению к предметной области. Они являются основой при построении протокола применения.

Протокол применения определяет специфичное для конкретной предметной области представление информации об изделии как основу для обмена данными, построенную на базе интегрированных ресурсов STEP.

Контрольные вопросы

4. Чем вызвана необходимость разработки формата STEP?
5. Какими средствами в STEP представляются концепт (концептуальная схема данных) и знак (модель) (см. Рис. 17).

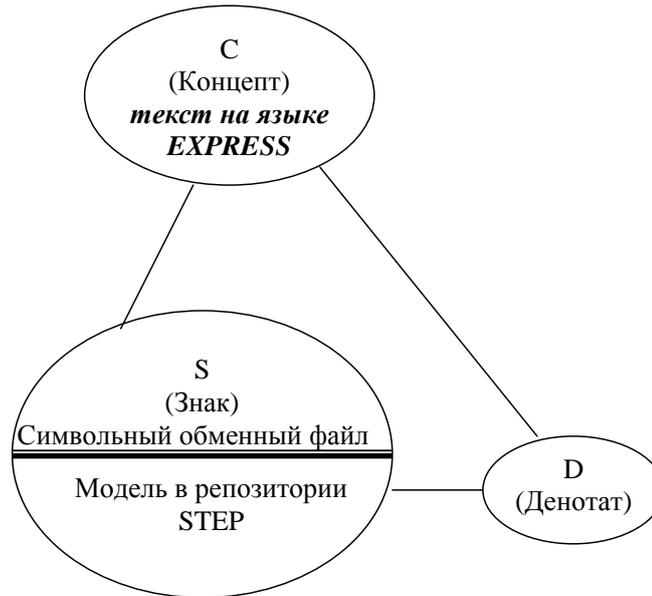


Рис. 17.

4. Базовые инвариантные инструменты ISO 10303 STEP

Все данные (как определения данных – концепта, так и информационные объекты – экземпляры данных – денотат) формата ISO 10303 STEP предназначены для программной обработки и, следовательно, эти данные должны быть формализованы. В данном разделе будут рассмотрены принятые в STEP средства формализации определений данных (методы описания) и средства формализации представлений экземпляров данных (методы реализации). Для формального описания синтаксиса языка EXPRESS и обменного файла STEP будет использоваться синтаксическая нотация Вирта (Wirth Syntax Notation – WSN).

Синтаксическая нотация Вирта обеспечивает однозначное представление формальных грамматик третьего уровня по иерархии Н.Хомского – контекстно-независимых грамматик. В стандарте ISO 10303 STEP Синтаксическая нотация Вирта принята в качестве инструмента, используемого для всех используемых в стандарте формальных грамматик.

Грамматика определяется как четверка (V_t, V_n, P, S) , где V_t - алфавит, символы которого называются **терминальными** (терминалами), из них строятся цепочки, порождаемые грамматикой. V_n - алфавит, символы которого называются **нетерминальными** (нетерминалами), используются при построении цепочек; они могут входить в промежуточные цепочки, но не должны входить в результат порождения. V_t и V_n не имеют общих символов. Полный алфавит (словарь) грамматики V определяется как объединение V_t и V_n . P - набор **порождающих правил (выводов)**, каждый элемент которых состоит из пары (a, b) , где a находится в V^+ , а b в V^* , a - левая часть правила, а b - правая, т.е. цепочки, построенные из символов алфавита V . Правило записывает-

ся $a = b$. S принадлежит V_n и называется **начальным** символом (аксиомой). Этот символ - отправная точка в получении любого предложения языка.

В нотации Вирта приняты следующие обозначения:

Знак равенства “=” означает вывод. Определено, что элемент слева должен быть комбинацией элементов справа. Любые пробелы, появляющиеся между элементами вывода, не являются значащими, если они появляются не внутри литерала. Вывод завершается точкой “.”;

Фигурные скобки “{ }” означают нуль или более повторений;

Квадратные скобки “[]” означают необязательный параметр;

Вертикальная линия “|” означает логическое ИЛИ;

Скобки “(“ ”)” показывают приоритет операций. В частности, там, где скобки включают в себя элементы, разделенные вертикальными линиями, один из элементов должен быть выбран в сочетании с любой другой операцией, например $A(B|C|D) = AB|AC|AD$.

Литерал, т.е. строка, которая определяет саму себя и является терминальным символом, далее не раскрывающимся, заключается в апострофы, например, ‘SCHEMA’.

В качестве примера рассмотрим формальную грамматику, описывающую вещественные число.

Начальным символом S здесь будет *вещественное_число*. Вещественное число записывается как *знак*, за которым следует *значение*. Знак и значение являются нетерминальными символами, т.е. входят во множество V_n . Знак является необязательным элементом, поскольку положительные вещественные числа, как правило, записываются без знака числа.

вещественное_число = [знак] значение. (WSN-1)

Знак раскрывается как один из двух литералов «+» и «-», которые определяют сами себя и далее не раскрываются, т.е. входят в множество терминальных символов V_t .

знак = ‘+’|‘-’. (WSN-2)

Значение вещественного числа состоит, в общем случае из целой части, дробной части и экспоненты. Наличие экспоненты допускается не во всех системах, работающих с вещественными числами, но поскольку мы рассматриваем самый общий случай, охватывающий все возможные варианты, введем экспоненту во множество нетерминальных символов V_n . Тогда вывод для значения имеет следующий вид:

значение = [целая_часть] [дробная_часть] [экспонента]. (WSN-3)

Как видим, из данного вывода следует возможность существования пустых значений, не представленных никакими символами (все символы являются необязательными). Можно сделать более сложный, но более корректный вывод:

значение = целая_часть | дробная_часть | экспонента | (целая_часть дробная_часть) | (дробная_часть экспонента) | (целая_часть экспонента) | (целая_часть дробная_часть экспонента). (WSN-4)

Очевидно, что такой вывод имеет только одно назначение: устранить возможность представления вещественных чисел в виде пустых строк, и на практике такие абсолютно корректные, но слишком сложные выводы не применяются.

Далее по порядку раскроем нетерминальные символы. Целая часть представляется следующим выводом:

целая_часть = {цифра}. (WSN-5)

Если принять, что в целой части обязательно должна быть хотя бы одна цифра, то имеем второй вариант вывода:

целая_часть = цифра {цифра}. (WSN-6)

Первый вариант более универсален, второй более строг и легче реализуется программно. Например, записи

.1 .2E+15

не соответствуют второму варианту.

дробная_часть = '.' {цифра}. (WSN-7)

На случай, если символ экспоненты может быть представлен как буквой нижнего регистра, так и буквой верхнего регистра, приведем для экспоненты наиболее полный вывод. Здесь учитывается также то, что может отсутствовать знак экспоненты, хотя в большинстве грамматик, описывающих вещественные числа, он является обязательным:

экспонента = 'e' | 'E' [знак] цифра {цифра}. (WSN-8)

Остается сделать вывод для символа цифра. Такой вывод очевиден, но для полноты грамматики его стоит привести:

цифра = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'. (WSN-9)

Это – самый общий вариант. В конкретных грамматиках могут быть введены некоторые ограничения, например, как мы увидим ниже, в обменном файле STEP в записи вещественного числа наличие десятичной точки является обязательным.

На основе формальной грамматики можно однозначно построить программный продукт, разбирающий тексты, соответствующие этой грамматике (интерпретатор текстов). Такой продукт строится как конечный автомат с магазинной памятью. Существуют программные средства, автоматизирующие процесс создания интерпретаторов текстов.

В томе ISO 10303-11 «справочное руководство по языку EXPRESS» и в томе ISO 10303-21 «явное текстовое кодирование структур обмена» приведены определения формальных грамматик в Синтаксической нотации Вирта. Эти определения несколько различаются, но в данном разделе они будут рассматриваться параллельно, и поэтому будет проведено их согласование.

Начнем с общих определений, которые будут использоваться в обоих нотациях. Прежде всего, это – цифры:

digit = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'. (WSN-10)

В обоих грамматиках: в тексте языка EXPRESS и в символьном обменном файле под буквами понимаются буквы из базового набора латинского алфавита. При этом в некоторых случаях допускается использование только букв верхнего регистра, поэтому имеет смысл дать отдельные определения для букв верхнего и нижнего регистров и общее определение чувствительной к регистру буквы:

upper = 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z'. (WSN-11)

lower = 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z'. (WSN-12)

letter = upper | lower. (WSN-13)

Методы описания. Язык концептуальных схем данных EXPRESS. Общие положения.

EXPRESS – это используемый в STEP язык определения данных или язык концептуальных схем данных. Описание языка EXPRESS содержится в томе ISO 10303-11. Язык EXPRESS предназначен для формального выражения концепта. Следовательно, описания концептуальных схем данных, выполненные средствами языка EXPRESS, являются онтологиями.

Язык EXPRESS может применяться вне стандарта STEP как самостоятельный язык информационного моделирования и уже используется в других международных стандартах (ISO 13584 PLIB, ISO 15531 MANDATE, ISO 15926 OIL&GAS, IFC), а также для представления информационной модели CALS NATO. Независимость EXPRESS от предметной области достигается за счет того, что для представления объекта применяется не специфическое, а обобщенное понятие «сущность» (**ENTITY**). Каждая сущность обладает атрибутами, выражающими характерные свойства моделируемого объекта. Например, сущность *геометрическая точка* имеет атрибуты, отражающие координаты этой точки в пространстве, сущность *изделие* имеет атрибуты, отображающие обозначение, наименование изделия и т.д.

Сущность – это основной элемент модели STEP. Существует множество определений понятия сущность, от сущностей в философских системах (вспомним одну из парных категорий – *сущность и явление*), сущностей в определении Пин-Шен Чена (нотация ER – сущность-связь), функциональные сущности и множество других определений. В STEP сущность определяется как класс информации, определяемый общими свойствами. Примем несколько более расширенное определение: *сущность – это класс информации, предназначенный для отображения объектов денотата, имеющих, с точки зрения рассматриваемой предметной области, одинаковый набор свойств.*

Язык EXPRESS не является языком программирования, а может быть использован только для задания информационных моделей.

Информационная модель на языке EXPRESS выражается в виде текста, состоящего из синтаксических элементов (компонентов языка), включающих:

алфавит;
 комментарии;
 зарезервированные слова;
 знаки;
 идентификаторы;
 литералы.

Из этих компонентов формируются семантические элементы (понятия языка). Собственно сам текст состоит из строк, а строки – из символов. Комбинации синтаксических элементов дают семантические элементы (понятия языка).

Алфавит языка, предназначенный для формирования всех остальных компонентов языка и содержащий набор символов EXPRESS, включает в себя арабские цифры, прописные и строчные буквы английского алфавита и специальные символы.

Основной набор символов (Character set) включает:

Цифры;

Буквы (регистр имеет значение только внутри символьных строк), в стандартный набор входит только латинский алфавит;

Подчеркивание, которое может использоваться внутри ключевых слов, но не может быть первым символом ключевого слова;

Специальные символы, к которым относятся *, (, \$ и т.д., которые используются для пунктуации и для обозначения операторов;

Пробел и конец строки. Пробелы не могут вставляться внутрь лексем. Пробелы во всех других случаях игнорируются. Конец строки во всех случаях, кроме тех, когда пробел завершает хвостовой комментарий, воспринимается как пробел.

Комментарий (Remark). Комментарии в языке EXPRESS бывают двух типов: встроенные (embedded) и хвостовые (tail).

remark = embedded_remark | tail_remark . (E141)

Встроенный комментарий заключается между парами символов (* и *). Если в тексте комментария встречается одна из пар (* и *), она должна быть заключена в апострофы.

embedded_remark = '(* { not_lparen_star | lparen_not_star | star_not_rparen | embedded_remark } *)' . (WSN-14)

Как видно из определения, допускается вложенность встроенных комментариев.

Хвостовой комментарий начинается с пары символов '--' и заканчивается концом строки.

tail_remark = '--' { \a | \s | \o } \n . (WSN-15)

Символы (Symbols). Символы - это отдельные символы или их группы, имеющие в языке EXPRESS специальное значение. Они используются в качестве операторов или в качестве разделителей (separator).

Зарезервированные слова (Reserved Words). Это - или ключевые слова (keywords), или операторы, или стандартные константы, функции и процедуры.

Начальным символом грамматики EXPRESS является *syntax* (синтаксис):

syntax = schema_decl { schema_decl } . (WSN-16)

Концептуальная схема данных, написанная на языке EXPRESS (декларация схемы), синтаксически определяется следующими выводами:

schema_decl = 'SCHEMA' schema_id ';' schema_body 'END_SCHEMA' ';' . (WSN-17)

Здесь *schema_id* – это обозначение (идентификатор), т.е. уникальное имя концептуальной схемы данных. Синтаксис обозначения концептуальной схемы данных подчиняется общим правилам построения идентификаторов, которые более подробно будут рассмотрены ниже.

Основная содержательная часть – это тело схемы *schema_body*. Например, ниже в качестве примера приведена схема *group_schema*, описывающая концепцию групп и отношений между ними. Схема взята из тома ISO 10303-41.



Рис. 18. Схема *group_schema*

Тело схемы *schema_body* определяется следующими выводами:

schema_body = { interface_specification } [constant_decl] { declaration | rule_decl } . (WSN-18)

declaration = entity_decl | function_decl | procedure_decl | type_decl . (WSN-19)

Здесь:

interface_specification – информация об определениях, которые заимствуются из других схем.

constant_decl – объявление используемых в схеме констант.

entity_decl – объявление сущности.

function_decl – объявление функции. Функции, как и процедуры, играют в EXPRESS вспомогательную роль и предназначены для вычисления производных атрибутов и задания ограничений, накладываемых на свойства и поведение сущностей.

procedure_decl – объявление процедуры.

type_decl – объявление нестандартного типа данных, определение которого дано в пределах схемы.

rule_decl – объявление «глобального» правила, используемого для задания сложных ограничений, накладываемых на свойства и поведение множества разнотипных сущностей.

В языке EXPRESS допускаются ссылки как вперед, так и назад. Это позволяет располагать декларации в тексте в произвольной последовательности. Для удобства работы человека с текстами EXPRESS общепринятым правилом является группирование деклараций по типам (сначала определенные в схеме типы, затем сущности, затем функции, затем процедуры, и в конце – глобальные правила). В пределах каждой группы декларации располагаются по алфавиту. Несоблюдение этого правила не является формальным нарушением, но затрудняет работу человека.

Более подробно все эти понятия языка EXPRESS будут рассмотрены ниже.

Методы реализации. Символьный обменный файл STEP. Общие положения.

Формат обменного файла STEP “мягкий” - в нем существуют только ограничения на максимальную длину физической записи (строки). Ограничение на максимальную длину физической записи сохранилось с того времени, когда одним из основных носителей данных были перфокарты. В настоящее время многие совместимые со STEP приложения генерируют и читают обменные файлы с произвольной длиной записи, хотя формально заложенное в стандарте ограничение длины физической записи остается в силе. Между разделителями допускается любое число пробелов, в любом месте могут быть вставлены комментарии.

Используемые в обменном файле символы:

space = ‘ ‘. (WSN-20)

special = ‘!’ | ‘”’ | ‘*’ | ‘\$’ | ‘%’ | ‘&’ | ‘.’ | ‘#’ | ‘+’ | ‘;’ | ‘-’ | ‘(’ | ‘)’ | ‘?’ | ‘/’ | ‘:’ | ‘;’ | ‘<’ | ‘=’ | ‘>’ | ‘@’ | ‘[’ | ‘]’ | ‘{’ | ‘|’ | ‘}’ | ‘^’ | ‘^’. (WSN-21)

reverse_solidus = ‘\’. (WSN-22)

apostrophe = ‘”’. (WSN-23)

character = space | digit | lower | upper | '_' | special | reverse_solidus | apostrophe.
(WSN-24)

Обменный файл обладает свойством самодостаточности и при своей передаче не сопровождается никакой дополнительной сопроводительной информацией, и поэтому он должен содержать в себе все необходимые для своей интерпретации данные. В обменном файле отсутствуют ссылки на какие-либо данные, не содержащиеся в самом обменном файле (за исключением имени EXPRESS-схемы).

Согласно стандарту, ключевые слова (KEYWORD) в обменном файле пишутся прописными буквами.

keyword = user_defined_keyword | standard_keyword . (WSN-25)

user_defined_keyword = '!' 'upper { upper | digit | '_' } . (WSN-26)

standard_keyword = upper { upper | digit | '_' } . (WSN-27)

К ключевым словам в обменном файле относятся:

- зарезервированные слова:

ISO-10303-21;

HEADER;

ENDSEC;

DATA;

END-ISO-10303-21;

"&SCOPE

ENDSCOPE

- имена EXPRESS-схем.. Содержимое обменного файла должно соответствовать определениям, данным в EXPRESS-схеме;

- определенные в соответствующей EXPRESS-схеме идентификаторы – именованные типы данных и элементы перечисления, а также именованные типы данных, определенные в специальной EXPRESS-схеме «header_section_schema».

Символьный обменный файл STEP имеет следующее синтаксическое определение:

exchange_file = "ISO-10303-21;" header_section data_section "END-ISO-10303-21;" .
(WSN-28)

Секция данных обменного файла состоит из множества записей – информационных объектов или экземпляров сущностей. Под экземпляром понимается идентифицируемое значение.

EI = {обозначение (идентификатор) информационного объекта, обозначение (идентификатор) определения информационного объекта, набор значений}

data_section = 'DATA' entity_instance_list 'ENDSEC' ';' . (WSN-29)

entity_instance_list = entity_instance {entity_instance} . (WSN-30)

Для представления экземпляра сущности в обменном файле используется следующая синтаксическая конструкция:

entity_instance = simple_entity_instance | complex_entity_instance. (WSN-31)

simple_entity_instance = entity_instance_name '=' [scope] simple_record ';'. (WSN-32)

complex_entity_instance = entity_instance_name "=" [scope] subsuper_record ";" . (WSN-33)

subsuper_record = "(" simple_record_list ")" . (WSN-34)

simple_record_list = simple_record { simple_record }. (WSN-35)

Член *simple_record*, представляющий содержательную часть записи, будет рассмотрен ниже.

Член *entity_instance_name* – это идентификатор записи, уникальный в пределах обменного файла:

entity_instance_name = '#' digit { digit }. (WSN-36)

Идентификатор – это то, что делает значение экземпляром. Как видно из определения, в символьном обменном файле STEP используется числовой идентификатор, т.е. это – просто номер. На значения идентификаторов, кроме их уникальности, накладывается еще одно ограничение: первая цифра не может быть нулем, т.е. нумерация записей начинается с единицы и предшествующие нули не используются. Эти ограничения в определении синтаксиса не учтены, и являются семантическими. Можно ли их учесть на уровне синтаксиса? Легко:

entity_instance_name = '#' natural_digit {digit}. (WSN-37)

natural_digit = '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'. (WSN-38)

digit = '0' | natural_digit. (WSN-39)

Почему это не сделано? – Вопрос к разработчикам стандарта. Может быть, из-за того, что это – излишнее усложнение синтаксиса. Подобные неточности могут влиять как на трудоемкость разработки интерпретаторов текстов, так и на надежность и эффективность работы этих интерпретаторов.

```

ISO-10303-21:
HEADER:
FILE_DESCRIPTION('STEP conformance test data',minimal AP203 instantiation',2;1);
FILE_NAME('min_crnA.p21',1993-07-22 T11:23:12;
(M,Green',J,Black)',Obie, Ventures, Inc.',P.O. Box,2222,'Middleton','Michigan',
'50800'),'NIST Data Proce, Release Mar2,1993 'conformance test suite',K.H.White');
FILE_SCHEMA('CONFIG_CONTROL_DESIGN');
ENDSEC:

DATA:
#1=PRODUCT('11111','Solid Cube','Description for part 11111',#2);
#2=MECHANICAL_CONTEXT('detailed design',#3,'mechanical');
#3=APPLICATION_CONTEXT('Control the configuration of three dimensional design');
#300=APPLICATION_PROTOCOL_DEFINITION('AP definition status', 'config_control_design', 1994, #3);
#4=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#5,#8,#1);
#5=PERSON_AND_ORGANIZATION(#6,#7);
#6=PERSON('333-003','White','K.',('H.',$,$));
#7=ORGANIZATION('NVI-Michigan',New Ventures, Inc., 'An engineering and research center);
#8=PERSON_AND_ORGANIZATION_ROLE('design_owner');
#9=PRODUCT_RELATED_PRODUCT_CATEGORY('detail','Part Type for product 11111',#1);
#10=PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE('A','Description of version with specified source for part 11111',#1,BOUGHT.);
#11=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#13,#15,#10);
#13=PERSON_AND_ORGANIZATION(#14,#7);
#14=PERSON('222-002','Pierre','Gabrielle',$,$,$);
#15=PERSON_AND_ORGANIZATION_ROLE('creator');
#44=CC_DESIGN_APPROVAL(#45,#10);
#45=APPROVAL(#46,'Approved as initial STEP test case part');
#46=APPROVAL_STATUS('approved');
#47=APPROVAL_DATE_TIME(#48,#45);
#48=DATE_AND_TIME(#50,#51);
#50=CALENDAR_DATE(1993,17,7);
#51=LOCAL_TIME(13,29,52.0,#29);
#69=APPROVAL_PERSON_ORGANIZATION(#70,#45,#72);
#70=PERSON_AND_ORGANIZATION(#71,#7);
#71=PERSON('777-007','Black','James',$,$,$);
#72=APPROVAL_ROLE('Authorize part release');
#31=CC_DESIGN_SECURITY_CLASSIFICATION(#32,#10);
#32=SECURITY_CLASSIFICATION('1993-C1','open availability of STEP test data',#33);
#33=SECURITY_CLASSIFICATION_LEVEL('unclassified');
#34=CC_DESIGN_DATE_AND_TIME_ASSIGNMENT(#35,#39,#32);
#35=DATE_AND_TIME(#36,#37);
#36=CALENDAR_DATE(1993,17,7);
#37=LOCAL_TIME(13,45,20.0,#29);
#39=DATE_TIME_ROLE('classification_date');
#40=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#41,#43,#32);
#41=PERSON_AND_ORGANIZATION(#42,#7);
#42=PERSON('555-005','Johnson','Stephen',$,(Mr.),$,$);
#43=PERSON_AND_ORGANIZATION_ROLE('classification_officer');
#53=CC_DESIGN_APPROVAL(#54,#32);
#54=APPROVAL(#55,'Approved as unclassified STEP test data');
#55=APPROVAL_STATUS('approved');
#56=APPROVAL_DATE_TIME(#57,#54);
#57=DATE_AND_TIME(#58,#59);
#58=CALENDAR_DATE(1993,17,7);
#59=LOCAL_TIME(13,47,28.0,#29);
#73=APPROVAL_PERSON_ORGANIZATION(#74,#54,#76);
#74=PERSON_AND_ORGANIZATION(#75,#7);
#75=PERSON('666-006','Spock','Robert',$,(S.),$,$);
#76=APPROVAL_ROLE('Authorize security code');
#17=PRODUCT_DEFINITION('D1','detailed drawing as planned for STEP conformance testing',#10,#18);
#18=DESIGN_CONTEXT('detailed design',#3,'design');
#19=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#13,#23,#17);
#23=PERSON_AND_ORGANIZATION_ROLE('creator');
#24=CC_DESIGN_DATE_AND_TIME_ASSIGNMENT(#25,#30,#17);
#25=DATE_AND_TIME(#27,#28);
#27=CALENDAR_DATE(1993,19,7);
#28=LOCAL_TIME(19,46,55.0,#29);
#29=COORDINATED_UNIVERSAL_TIME_OFFSET(8,$,BEHIND.);
#30=DATE_TIME_ROLE('creation_date');
#61=CC_DESIGN_APPROVAL(#62,#17);
#62=APPROVAL(#63,'Approved as STEP conformance testing drawing');
#63=APPROVAL_STATUS('approved');
#64=APPROVAL_DATE_TIME(#65,#62);
#65=DATE_AND_TIME(#66,#67);
#66=CALENDAR_DATE(1993,19,7);
#67=LOCAL_TIME(19,47,51.0,#29);
#77=APPROVAL_PERSON_ORGANIZATION(#70,#62,#79);
#79=APPROVAL_ROLE('Authorize product definition');
#80=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#81,#84,#10);
#81=PERSON_AND_ORGANIZATION(#82,#83);
#82=PERSON('CDI-2','Smith','B.',('K.',$,$));
#83=ORGANIZATION('CDI','Contract Design, Inc.','Contract engineering organization. ');
#84=PERSON_AND_ORGANIZATION_ROLE('design_supplier');
ENDSEC:
END-ISO-10303-21;

```

Заголовочная
секция
header_section

Секция
данных
data_section

Рис. 19. Пример обменного файла STEP

Идентификаторы записей в обменном файле STEP – это, по сути, числа. Единственным ограничением, накладываемым на них, является их уникальность в пределах обменного файла. Из-за того, что структуры данных в STEP достаточно сложные, и рекурсивные ссылки в моделях STEP хотя и нежелательны, но не запрещены, упорядочить записи в обменном файле таким образом, чтобы исключить ссылки вперед, не представляется возможным. Ссылки на записи допускаются как вперед, так и назад.

Никакой смысловой нагрузки идентификаторы не несут. Большинство программ (препроцессоров STEP и т.д.), генерирующих обменные файлы, присваивают записям идентификаторы с возрастающими значениями от начала к концу обменного файла, например, #1 #2 #3 #4 или #10 #20 #30. В то же время некоторые программы, например, конвертор STEP систем CATIA и Pro/Engineer, присваивают идентификаторы произвольным образом (запись с

идентификатором #21 может появиться в конце обменного файла, между записями #50134 и #50135). Это затрудняет чтение обменного файла человеком, но с точки зрения стандарта STEP не является нарушением и не препятствует программной обработке обменного файла.

Одно из основных положений стандарта ISO 10303 STEP – отсутствие доступной для пользователя концепции идентификации данных. Вследствие этого ни в коем случае нельзя полагаться на данные идентификаторы:

1. После прочтения приложением обменного файла идентификаторы чаще всего теряются и во внутренних структурах данных приложения отсутствуют;

2. Если те же самые данные выгрузить из приложения снова в обменный файл, идентификаторы тех же самых элементов данных не обязаны совпадать с идентификаторами в исходном файле.

Некоторые системы могут сохранять идентификаторы обменного файла после загрузки модели STEP, но это является дополнительной нестандартной функциональной возможностью (Рис. 20):

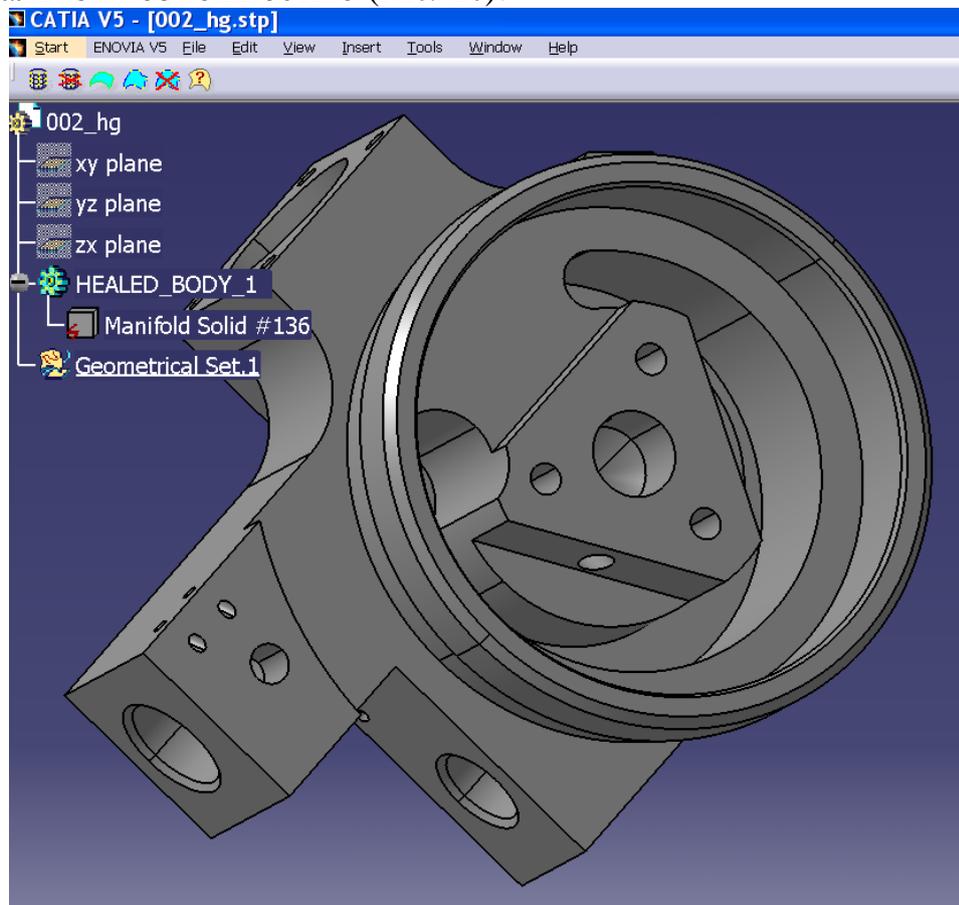


Рис. 20. Модель, загруженная в CAD-систему CATIA из STEP

Здесь в дереве структуры изделия приведен идентификатор #136, который твердое тело, представляющее форму детали, имело в обменном файле:

```
#136=MANIFOLD_SOLID_BREP('HEALED_BODY_1 ',#137);
```

```
#137=CLOSED_SHELL(",(#424,#425,#426,#427,#428,#429,#430,#431,#432,#433,  
#434,#435,#436,#437,#438,#439,#440,#441,#442,#443,#444,#445,#446,#447,#448,
```

#449,#450,#451,#452,#453,#454,#455,#456,#457,#458,#459,#460,#461,#462,#463,
 #464,#465,#466,#467,#468,#469,#470,#471,#472,#473,#474,#475,#476,#477,#478,
 #479,#480,#481,#482,#483,#484,#485,#486,#487,#488,#489,#490,#491,#492,#493,
 #494,#495,#496,#497,#498,#499,#500,#501,#502,#503,#504,#505,#506,#507,#508,
 #509,#510,#511,#512,#513));

Также, в протоколах трансляции некоторые САД-системы могут указывать идентификаторы некорректных записей, которые они имели в исходном обменном файле, что облегчает выявление и устранение ошибок:

```

=====
*** = Processing new independent element
* = Intermediate processing
!! = Independent element K.O.
! = Intermediate error
-----
<I> = Information
<W> = Warning
<E> = Error
-----
[0000] = Message identifier : 0000
[T=xxx] = Entity Type : xxx
[#0000] = Entity identifier number : 0000
=====
Actual display level : Customer

<W> [0165] [T=B_SPLINE_CURVE_WITH_KNOTS] [#421] NURBS curve quality checker Diag: Arc(s) of curve too
small
<W> [0165] [T=B_SPLINE_CURVE_WITH_KNOTS] [#423] NURBS curve quality checker Diag: Arc(s) of curve too
small
<E> [0046] [T=EDGE_LOOP] [#225] Edges of boundary are not created
!! <E> [0002] [T=ADVANCED_FACE] [#184] Element not created
!! <E> [0002] [T=ADVANCED_FACE] [#184] Element not created
<W> [0165] [T=B_SPLINE_CURVE_WITH_KNOTS] [#430] NURBS curve quality checker Diag: Arc(s) of curve too
small
<W> [0223] [T=B_SPLINE_SURFACE_WITH_KNOTS] [#202] NURBS surface quality checker Diag: Patch(es) of
surface too small
<W> [0165] [T=B_SPLINE_CURVE_WITH_KNOTS] [#421] NURBS curve quality checker Diag: Arc(s) of curve too
small
<W> [0165] [T=B_SPLINE_CURVE_WITH_KNOTS] [#423] NURBS curve quality checker Diag: Arc(s) of curve too
small
<W> [0165] [T=B_SPLINE_CURVE_WITH_KNOTS] [#430] NURBS curve quality checker Diag: Arc(s) of curve too
small
<W> [0903] [T=CLOSED_SHELL] [#137] Shell created with errors; trying to repair
<I> [0776] [T=CLOSED_SHELL] [#137] The entity name is CLOSED_SHELL #137
!! <E> [0002] [T=MANIFOLD_SOLID_BREP] [#136] Element not created

```

Здесь #421, #423 и т.д. – это идентификаторы записей в обменном файле.

Элементы языка EXPRESS и представление соответствующих экземпляров данных в символьном обменном файле STEP

Основной элемент модели STEP – это сущность. Обозначим определение сущности через ED (“Entity definition”). Для того, чтобы различать определения сущностей между собой, необходимо присвоить определению сущности некоторое уникальное обозначение (идентификатор). Очевидно, определение сущности не ограничивается только ее обозначением (идентификатором):

$ED = \{\text{Обозначение (идентификатор) сущности, дополнительные определения}\}$

Что подразумевается под «дополнительными определениями», будет раскрыто далее.

В тексте на языке EXPRESS определение сущности задается следующими синтаксическими конструкциями:

entity_decl = entity_head entity_body 'END_ENTITY' ';'. (WSN-40)

entity_head = 'ENTITY' entity_id [sub_super] ';'. (WSN-41)

Здесь *entity_id* – это идентификатор (уникальное обозначение) сущности. В заголовке сущности посредством необязательного члена *sub_super* задаются отношения сущности с ее супертипами и, возможно, с подтипами. Что подразумевается под «уникальностью» обозначения и как это обозначение выглядит? В настоящее время наиболее распространенным подходом является использование символьных обозначений.

Такое обозначение понятно человеку, а возможности современных компьютеров позволяют легко обрабатывать символьные строки произвольной длины. Если учесть, что сущности STEP соответствуют некоторым объектам денотата, то, наверное, самым разумным и удобным было бы давать сущностям обозначения, совпадающие с тем, как называются отображаемые сущностью объекты денотата, например, *лев, заяц, изделие, станок, заготовка, режущий инструмент* и т.д.. Единственное ограничение здесь состоит в том, что в компьютерных технологиях базовым набором символов во всех формальных языках является латиница. Но с этим приходится мириться, для конечного пользователя интерфейс может быть организован на любом языке и с применением любых гарнитур символов, включая арабскую, корейскую, грузинскую и т.д., но в основе всегда лежат программы и определения, заданные латинскими буквами, и чаще всего – на английском языке.

Идентификатор в EXPRESS – это последовательность букв, цифр и знаков подчеркивания, причем первым членом последовательности обязательно должна быть буква. Поскольку идентификаторы есть не только у сущностей, и понятия всех типов в EXPRESS имеют идентификаторы, имеет смысл вывести идентификатор как отдельный нетерминальный символ:

simple_id = letter {letter | digit | '_'}. (WSN-42)

Тогда идентификаторы всех понятий раскрываются следующим образом:

Идентификатор сущности:

entity_id = simple_id. (WSN-43)

Идентификатор атрибута (свойства):

attribute_id = simple_id. (WSN-44)

Идентификатор константы:

constant_id = simple_id. (WSN-45)

Идентификатор элемента перечислимого типа (enumeration):

enumeration_id = simple_id . (WSN-46)

Идентификатор функции:

function_id = simple_id . (WSN-47)

Идентификатор параметра (функции)

parameter_id = simple_id . (WSN-48)

Идентификатор процедуры:

procedure_id = simple_id . (WSN-49)

Идентификатор глобального правила:

rule_id = simple_id . (WSN-50)

Идентификатор схемы:

schema_id = simple_id. (WSN-51)

Идентификатор объявленного в схеме типа данных:

type_id = simple_id. (WSN-52)

Обозначение (идентификатор) любого информационного объекта уникально на некотором множестве схожих информационных объектов. Такое множество называется «информационная база». Если вспомнить, что с точки зрения различных предметных областей (в различных концептуальных схемах данных) один и тот же объект денотата может описываться по-разному, то ясно, что обозначение сущности уникально только в пределах концептуальной схемы данных. Для обозначения (идентификатора) определения сущности информационной базой будет концептуальная схема данных. Например, если вернуться к нашим львам, в разных концептуальных схемах данных, описывающих разные предметные области, один и тот же объект денотата – *лев* будет представлен разными сущностями: *цирк.лев*, *сафари.лев*, *ветеринария.лев* и т.д.

entity_body = {explicit_attr} [derive_clause] [inverse_clause] [unique_clause] [where_clause]. (WSN-53)

Как видно из определений, большинство синтаксических элементов являются необязательными. Следовательно, в качестве примера можно привести простейший вариант определения сущности, в котором присутствуют только обязательные элементы синтаксиса:

**ENTITY integer_type;
END_ENTITY;**

Все прочие элементы являются необязательными и будут подробнее раскрыты ниже. Здесь дадим только их краткую характеристику:

explicit_attr – явные атрибуты (свойства). Это – те свойства, значения которых или полностью независимы друг от друга или их зависимость не может быть выражена непосредственно и задается посредством набора правил (см. ниже). Значения явных атрибутов задаются непосредственно создателем модели.

derived_clause – множество производных (вычисляемых) атрибутов. Значения таких атрибутов могут быть однозначно вычислены на основе значе-

ний явных атрибутов (см. ниже) и не могут быть заданы создателем модели непосредственно.

inverse_clause – множество инверсных атрибутов (обратных ссылок). По своей сути это – специфическая разновидность производных атрибутов.

unique_clause – множество правил, задающих уникальность значения атрибутов или совокупности атрибутов.

where_clause – множество местных правил, накладывающих ограничения на значения атрибутов.

Ниже приведены примеры определения трех сущностей, включающие в той или иной комбинации все синтаксические конструкции.

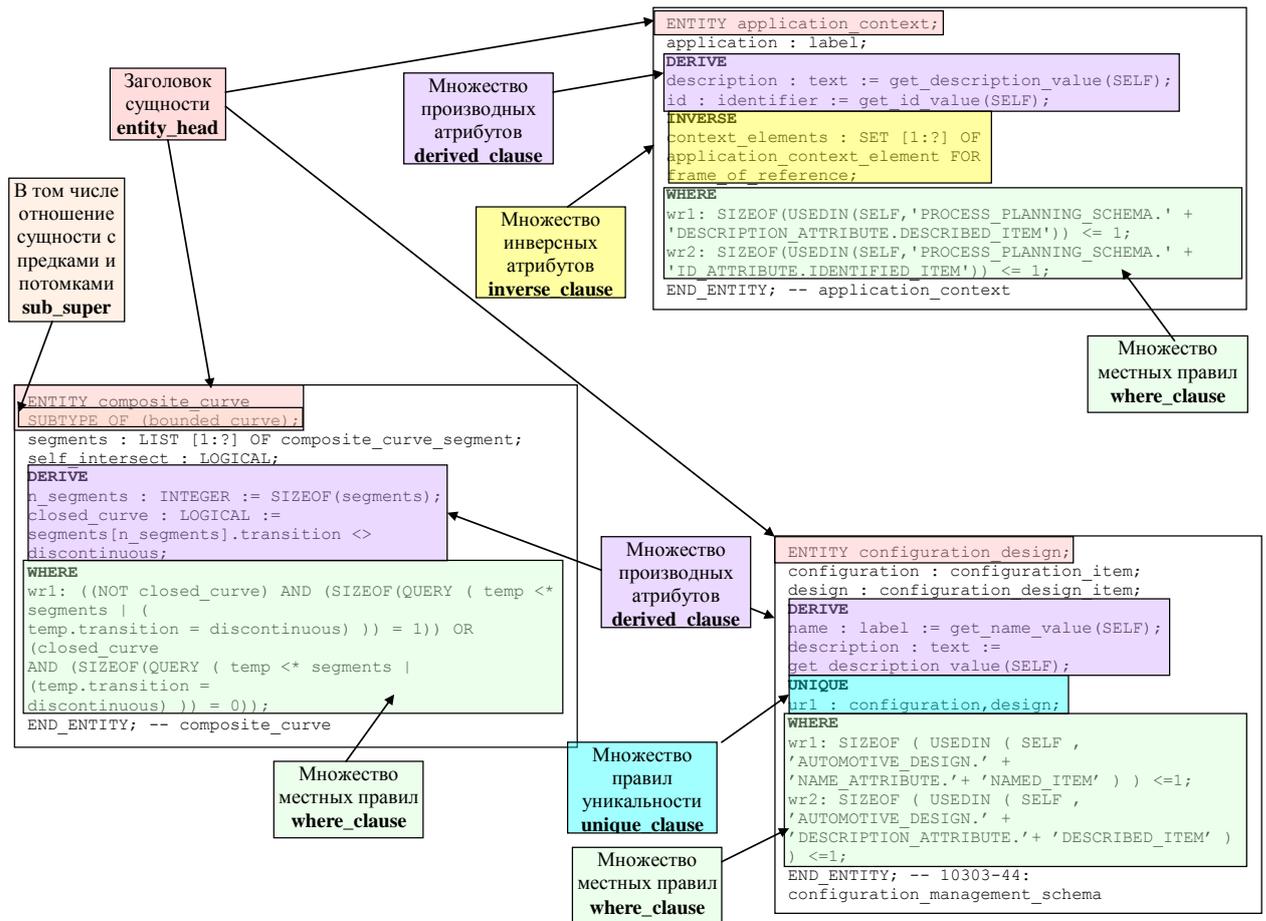


Рис. 21. Примеры определения трех сущностей

Например, показанная на Рис. 21 сущность *composite_curve* (составная кривая) представляет кривую, состоящую из множества последовательно соединенных сегментов кривых. Явные атрибуты здесь – *segments* (список сегментов) и *self_intersect* (флаг самопересекаемости). Производные атрибуты – *n_segments* (количество сегментов), который может быть определен посредством запроса размера списка и флаг *closed_curve* (замкнутая кривая), который может быть определен на основании того, есть ли у последнего сегмента указатель на следующий за ним сегмент. В блоке правил WHERE задано следующее условие: **WR1:** Ни для одного сегмента, кроме последнего сег-

мента открытой кривой, код смещения не должен иметь значения *discontinuous* (разрывный).

Для наглядности представления концептуальных схем данных, написанных на языке EXPRESS, разработана специальная графическая нотация EXPRESS-G. Назначение EXPRESS-G лишь иллюстративное, и его средствами невозможно отобразить полностью все определения, существующие в текстовом варианте EXPRESS. Первый вариант EXPRESS-G был опубликован в 1990 году, за несколько лет до появления языка UML. Между EXPRESS-G и UML существуют различия, иногда – существенные, и последние годы ведутся работы по их согласованию.

В графической нотации EXPRESS-G определение сущности обозначается сплошным прямоугольником, в котором записан идентификатор. Например, сущность *integer_type*, определение которой дано выше, в нотации EXPRESS-G будет представлена:

integer_type

Контрольные вопросы

6. Что такое сущность в STEP. Что входит в определение сущности на языке EXPRESS?
7. Что такое EXPRESS-G? Может ли диаграмма EXPRESS-G полностью заменить текст на языке EXPRESS?

То, что рассматривалось выше, касается определения (интенционала) сущности. Множество экземпляров сущности образуют экстенционал сущности (элементы знака). В данном разделе будет рассматриваться представление экземпляров сущностей в символьном обменном файле в соответствии с ISO 10303-21. Представление экземпляров сущностей в репозитории STEP стандартизовано только в обобщенном виде, подробности такого представления оставлены на усмотрение разработчиков программной реализации STEP. Представление экземпляров сущностей в репозитории STEP будет рассмотрено ниже.

Можно предположить, что та запись, которую мы рассматриваем (экземпляр сущности типа *integer_type*), относится к простым – уж куда еще проще! Поэтому здесь раскроем синтаксис для простого экземпляра сущности (второй вариант будет рассмотрен ниже):

Пока необязательный член выражения *scope* рассматривать не будем, а раскроем дальше два других члена из правой части вывода.

simple_record = keyword '(' [**parameter_list**] ')'. (WSN-54)

Член *parameter_list* в правой части – это список параметров, т.е. значений атрибутов. Поскольку у рассматриваемой сущности *integer_type* атрибутов

нет, раскрытие члена *parameter_list* можно отложить на потом. Тогда остается член *keyword* (ключевое слово).

Под ключевым словом в данном контексте понимается обозначение определения сущности. Поскольку ключевые слова могут включать буквы только верхнего регистра (см. определение члена *standard_keyword*), то буквы нижнего регистра из обозначения отображаются в ключевом слове в буквы верхнего регистра. Запись экземпляра сущности будет, в соответствии с рассмотренными синтаксическими правилами, выглядеть:

#1 = INTEGER_TYPE();

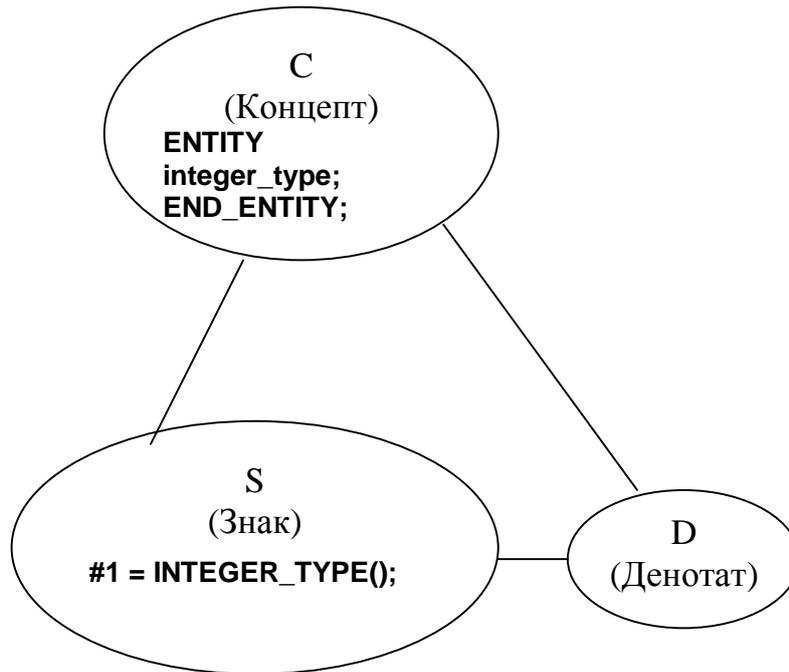


Рис. 22. Запись экземпляра сущности

Поскольку сущность – это класс информации, то очевидно, что для полного определения сущности в большинстве случаев недостаточно просто указать ее обозначение, ведь класс информации, согласно основам объектного подхода, обладает множеством атрибутов и методов. Почему в «большинстве случаев», а не «всегда»? Как можно видеть из вышеприведенного примера, встречаются и такие сущности, которые не имеют никаких свойств, кроме своего обозначения. Такие сущности самодостаточны.

Это – одно из отличий между реляционной и объектной моделями данных. При реляционной модели данных невозможно существование двух полностью идентичных записей, а при объектной модели допускается как существование неограниченного числа полностью идентичных записей, так и пустых записей. Но все-таки большая часть сущностей имеет некоторые свойства. Свойства сущности представлены атрибутами. В определении (декларации) сущности могут присутствовать атрибуты трех разновидностей (см. выше): явные, производные и инверсные. Поскольку наибольший интерес представляют явные атрибуты, с них и начнем.

Определение явного атрибута в синтаксической нотации Вирта представляется следующим выводом:

explicit_attr = attribute_decl { ',' attribute_decl } ':' ['OPTIONAL'] base_type ';'. (WSN-55)

attribute_decl = attribute_id | qualified_attribute. (WSN-56)

Атрибут имеет область допустимых значений (*domain*), которая представлена в выводе членом *base_type*. Дальнейшее рассмотрение начнем с областей допустимых значений, которая раскрывается посредством следующего вывода:

base_type = aggregation_types | simple_types | named_types. (WSN-57)

Из трех разновидностей областей определения привлекает своей простотой вторая разновидность: простейшие типы (*simple_types*). С них и начнем:

simple_types = binary_type | boolean_type | integer_type | logical_type | number_type | real_type | string_type. (WSN-58)

Как видно из данного вывода, в EXPRESS существует семь простейших типов данных. В таблице ниже дано определение их синтаксиса в нотации Вирта.

В символьном обменном файле STEP значения атрибутов представлены в виде вышеупомянутого списка параметров:

parametr_list = parameter { ',' parameter }. (WSN-59)

parameter = typed_parameter | untyped_parameter | omitted_parameter. (WSN-60)

Раскроем нетипизированный параметр

untyped_parameter = undefined_parameter | integer | real | string | entity_instance_name | enumeration | binary | list. (WSN-61)

Ссылка на тип данных в текстах на языке EXPRESS	Представление типа данных в схемах EXPRESS-G	Представление экземпляров данных (параметров) в символьном обменном файле STEP
binary_type = 'BINARY' [width_spec]. (WSN-62)		binary = ''' ('0' '1' '2' '3') { hex } '''. (WSN-63)
boolean_type = 'BOOLEAN'. (WSN-64)		В обменном файле параметр этого типа рассматривается как частный случай перечислимого (enumeration) типа, который может принимать только значения Т (истина) и F (ложь). Специального синтаксического определения у параметра этого типа нет
integer_type = 'INTEGER'. (WSN-65)		integer = [sign] digit {digit}. (WSN-66)

Ссылка на тип данных в текстах на языке EXPRESS	Представление типа данных в схемах EXPRESS-G	Представление экземпляров данных (параметров) в символьном обменном файле STEP
logical_type = 'LOGICAL' . (WSN-66)		В обменном файле параметр этого типа рассматривается как частный случай перечислимого (enumeration) типа, который может принимать только значения Т (истина) F (ложь), и U (неизвестно). Специального синтаксического определения у параметра этого типа нет
number_type = NUMBER . (WSN-67)		В обменном файле параметр этого представляется как целое или вещественное значение. Специального синтаксического определения у параметра этого типа нет
real_type = 'REAL' [(' precision_spec ')]. (WSN-68)		real = [sign] digit {digit} '.' {digit} [,E' [sign] digit {digit}]. (WSN-69)
string_type = 'STRING' [width_spec]. (WSN-70)		string = ''' { non_q_char apostrophe apostrophe reverse_solidus reverse_solidus control_directive }'' . (WSN-71)

Для некоторых определений применяется символ знак.

sign = '+' | '-'. (E159)

В этих определениях для некоторых из типов данных могут быть заданы параметры: точность для вещественных (REAL) значений.

precision_spec = numeric_expression. (WSN-72)

Кроме того, двоичные (BINARY) и строковые (STRING) значения, если для них задана длина, могут быть объявлены как имеющие фиксированную длину:

width_spec = (' width ') ['FIXED']. (WSN-73)

width = numeric_expression. (WSN-74)

numeric_expression = simple_expression. (WSN-75)

simple_expression = term { add_like_op term }. (WSN-76)

В общем виде численное выражение имеет достаточно сложный синтаксис:.

term = factor { multiplication_like_op factor }. (WSN-77)

factor = simple_factor ['' simple_factor]**. (WSN-78)

simple_factor = aggregate_initializer | entity_constructor | enumeration_reference | interval | query_expression | ([unary_op] ('(' expression ')' | primary)).
(WSN-79)

Для целей задания точности вещественных чисел или длин строк будут, скорее всего, использоваться цифровые литералы, являющиеся частным случаем численного выражения, поэтому далее раскроем только часть последовательности выводов:

primary = literal | (qualifiable_factor { qualifier }). (WSN-80)

literal = binary_literal | integer_literal | logical_literal | real_literal | string_literal.
(WSN-81)

integer_literal = digits. (WSN-82)

digits = digit { digit }. (WSN-83)

Теперь, освоив минимальный набор синтаксических правил, можно рассмотреть примеры определений и экземпляров простейших сущностей. В Прикладных протоколах (концептуальных схемах данных) STEP такие простейшие сущности встречаются крайне редко. Одна из немногочисленных сущностей такого типа (простейшая, и к тому же имеющая все атрибуты простейших типов) представлена ниже:

```
ENTITY dimensional_exponents;
length_exponent : REAL;
mass_exponent : REAL;
time_exponent : REAL;
electric_current_exponent : REAL;
thermodynamic_temperature_exponent : REAL;
amount_of_substance_exponent : REAL;
luminous_intensity_exponent : REAL;
END_ENTITY; -- dimensional_exponents
```

Чтобы данный пример не выглядел абстрактно, поясним смысл сущности *dimensional_exponents*. Эта сущность служит для представления размерности единиц измерения. Предполагается, что любая единица измерения может быть выражена в виде комбинации семи базовых физических свойств: длина, масса, время, сила тока, термодинамическая температура, количество вещества и интенсивность света.

Тогда описание размерности единицы длины будет представлено в обменном файле следующей записью:

#1 = DIMENSIONAL_EXPONENTS(1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

Размерность единицы площади:

#10 = DIMENSIONAL_EXPONENTS(2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

Размерность единицы объема:

#20 = DIMENSIONAL_EXPONENTS(3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

Размерность единицы ускорения:

#30 = DIMENSIONAL_EXPONENTS(1.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0);

Размерность единицы силы:

#40 = DIMENSIONAL_EXPONENTS(1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0);

Размерность единицы электрического напряжения:

#50 = DIMENSIONAL_EXPONENTS(2.0, 1.0, -3.0, -1.0, 0.0, 0.0, 0.0);

Размерность единицы электрической емкости:

#60 = DIMENSIONAL_EXPONENTS(-2.0, -1.0, 4.0, 2.0, 0.0, 0.0, 0.0);

Размерность единицы поглощенной дозы ионизирующего излучения:

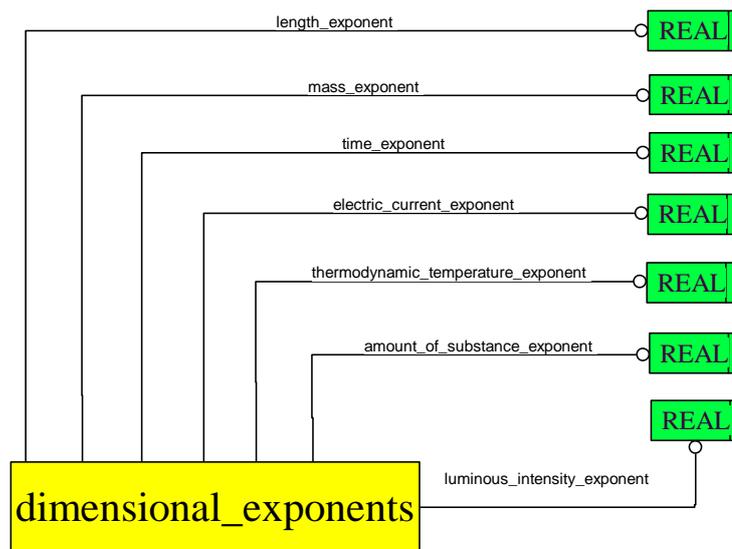
#70 = DIMENSIONAL_EXPONENTS(2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0);

Размерность единицы освещенности:

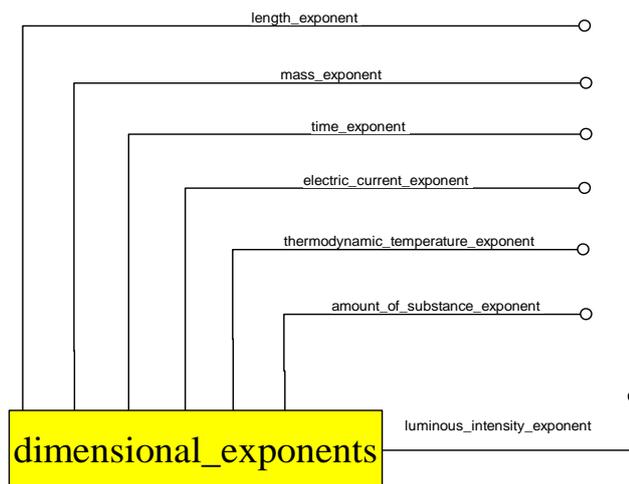
#80 = DIMENSIONAL_EXPONENTS(-2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);

Можно заметить, что поля параметров следуют в той же последовательности, что и объявления атрибутов сущности. Полей ровно столько, сколько у сущности объявлено явных атрибутов.

В нотации EXPRESS-G атрибут обозначается тонкой направленной линией, связывающей сущность, которой принадлежит атрибут, с областью допустимых значений атрибута. На конце линии ставится кружок. Около линии в произвольном месте пишется обозначение (идентификатор) атрибута:



Схемы EXPRESS-G в основном используются в качестве иллюстраций, обеспечивающих представление конструкций на языке EXPRESS, иногда – достаточно сложных. Для того, чтобы избежать излишнего усложнения схем, на этих схемах изображение простейших типов обычно опускается:



Следующая возможная разновидность базовых типов – это именованные типы (*named_types*), к которым относятся ссылки на другие сущности и типы данных, определения которых даны в схеме (*defined_types*).

named_types = entity_ref | type_ref. (WSN-84)

В синтаксисе языка EXPRESS предусмотрено различие между идентификатором и ссылкой. Рассмотрим понятие ссылки на сущность.

entity_ref = simple_id. (WSN-85)

С точки зрения синтаксиса, ссылка и идентификатор идентичны, отличаются они только по своему назначению: идентификатор встречается один раз, при объявлении (декларации) сущности. Когда сущность объявлена, на нее может иметься произвольное число ссылок. Здесь мы приближаемся к не слишком четкой границе между синтаксисом и семантикой.

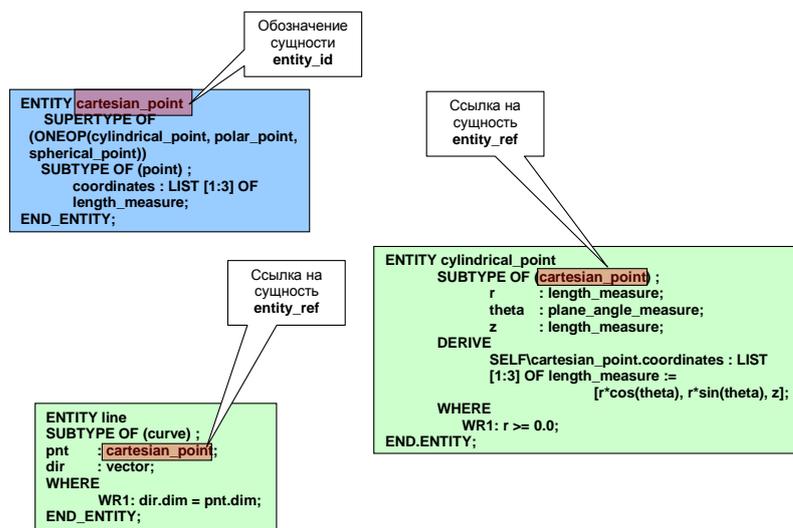


Рис. 23

Например, декартова точка *cartesian_point* объявлена и описана в схеме один раз, а ссылка на нее встречается многократно – как на супертип, например, в объявлении ее подтипа – точки в цилиндрической системе координат *cylindrical_point*, так и в качестве области допустимых значений атрибутов,

например, в объявлении сущности *line* (бесконечная прямая линия), заданная точкой в декартовом пространстве и вектором.

Ссылка в тексте может делаться на все элементы, имеющие идентификатор:

Ссылка на атрибут:

attribute_ref = attribute_id. (WSN-86)

В некоторых случаях обычной ссылки оказывается недостаточно. Например, обозначение (идентификатор) атрибута уникально только в пределах сущности, к которой относится атрибут. Таким образом, в одной схеме может существовать произвольное количество атрибутов, имеющих одно обозначение (идентификатор), но относящихся к различным сущностям. Более того, одна сущность может иметь несколько одноименных атрибутов, из которых не более чем один объявлен в определении самой сущности, и произвольное число атрибутов с тем же именем могут быть унаследованы сущностью от ее супертипов. Следовательно, при ссылке для однозначного обозначения (идентификации) атрибута необходимо указать обозначение сущности и обозначение самого атрибута:

attribute_qualifier = '.' attribute_ref. (WSN-87)

qualified_attribute = 'SELF' group_qualifier attribute_qualifier. (WSN-88)

group_qualifier = '\' entity_ref. (WSN-89)

Ссылка на константу:

constant_ref = constant_id. (WSN-90)

Ссылка на элемент перечислимого типа:

enumeration_ref = enumeration_id. (WSN-91)

enumeration_reference = [type_ref '.'] enumeration_ref. (WSN-92)

Ссылка на функцию:

function_ref = function_id. (WSN-93)

Обобщенная ссылка:

general_ref = parameter_ref | variable_ref. (WSN-94)

Ссылка на параметр функции:

parameter_ref = parameter_id. (WSN-95)

Ссылка на процедуру:

procedure_ref = procedure_id. (WSN-96)

Ссылка на ресурс. Под ресурсом здесь понимается обобщение нескольких допустимых типов ссылок. Данный символ используется только в определении импорта определений между схемами (см. ниже):

resource_ref = constant_ref | entity_ref | function_ref | procedure_ref | type_ref. (WSN-97)

Ссылка на схему:

schema_ref = schema_id. (WSN-98)

Ссылка на переменную:

variable_ref = variable_id. (WSN-99)

И, наконец, ссылка на тип, определение которого дано в схеме:

type_ref = type_id. (WSN-100)

Рассмотрев обозначение типа и ссылку на тип, перейдем к определению (объявлению, декларации) типа:

type_decl = 'TYPE' type_id '=' underlying_type ';' [where_clause] 'END_TYPE' ';'. (WSN-101)

Так же, как и к сущности, к определенному в схеме типу может применяться набор правил *where_rules*, накладывающий ограничения на область допустимых значений экземпляра этого типа. Чаще всего значение типа используется в качестве параметра, т.е. значения атрибута сущности. Лежащий в основе тип (*underlying_type*) может быть одним из следующих:

underlying_type = constructed_types | aggregation_types | simple_types | type_ref. (WSN-102)

В EXPRESS существуют две разновидности сконструированных типов:

constructed_types = enumeration_type | select_type. (WSN-103)

Определенные в схеме типы можно, в зависимости от лежащего в основе типа, разделить на три категории:

- Определенный в схеме тип общего вида, в основе которого лежит агрегатный тип *aggregation_types*, простейший тип *simple_types* или ссылка на другой определенный в схеме тип *type_ref*;
- Определенный в схеме тип перечислимого типа (в основе которого лежит *enumeration_type*);
- Определенный в схеме тип выбираемого типа(в основе которого лежит *select_type*);

Для типов каждой из этих категорий в EXPRESS-G принято свое обозначение:

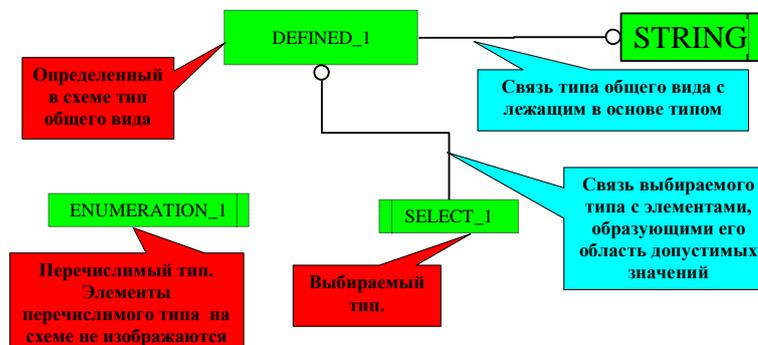


Рис. 24

Сконструированные типы (перечислимый и выбираемый) более подробно будут рассмотрены ниже.

Из приведенных в выводе вариантов лежащего в основе типа имеет смысл начать рассмотрение с простейших типов (так же, как это делалось и для рассмотренных выше областей допустимых значений атрибутов сущности).

В качестве примера приведем простейшие и очень часто используемые в Прикладных протоколах (стандартизованных концептуальных схемах данных) STEP определения. Идентификатор в данном случае – это внешний идентификатор:

```
TYPE identifier = STRING;  
END_TYPE; -- identifier
```

Метка – это строка символов, понятная для человека и поясняющая суть изделия, ресурса, операции, свойства и т.д., например, «заклепка», «кронштейн», «фрезерование», «токарный станок», «конденсатор» и т.д.

```
TYPE label = STRING;  
END_TYPE; -- label
```

Текст – это произвольное неформализованное текстовое описание.

```
TYPE text = STRING;  
END_TYPE; -- text
```

Эти определенные в схеме типы данных семантически эквивалентны и отличаются только своим назначением.

Например, рассмотренные типы данных используются в качестве областей допустимых значений сущности «обобщенное свойство» (*general_property*). Под обобщенным свойством понимается некоторое свойство объекта, например, изделия, ресурса и т.д., которое не формализовано или из-за того, что в данной предметной области это свойство редко встречается или из-за того, что формализация свойства нецелесообразна в силу сложности свойства.

```
ENTITY general_property;  
id : identifier;  
name : label;  
description : OPTIONAL text;  
END_ENTITY;
```

Или версия запроса на выполнение работ

```
ENTITY versioned_action_request;  
id : identifier;  
version : label;  
purpose : text;  
description : OPTIONAL text;  
END_ENTITY; -- versioned_action_request
```

Если при задании области допустимых значений атрибута использовался определенный в схеме тип общего вида, в обменном файле STEP поле соответствующего параметра будет заполняться в соответствии с синтаксически-

ми правилами, определенными для лежащего в основе типа. В данном примере это будут символьные строки:

```
#1 = GENERAL_PROPERTY('pr00-01', 'colour', 'colour of machine tool necessary for aesthetic imagine');
```

У обеих сущностей в определении атрибутов *description* используется ключевое слово *OPTIONAL* (необязательный), которое уже упоминалось в синтаксисе определения явного атрибута. Смысл ключевого слова *OPTIONAL* заключается в том, что в знаке (например, в прикладной модели, представленной в форме обменного файла) у такого атрибута допускается отсутствие значения. Если у атрибута значение отсутствует, в поле соответствующего параметра, в соответствии с выводом:

```
undefined_parameter = '$'. (WSN-104)
```

ставится знак \$, например:

```
#1 = GENERAL_PROPERTY('pr00-01', 'colour', $);
```

Определения приведенных выше сущностей *general_property* и *versioned_action_request* будут выглядеть следующим образом:

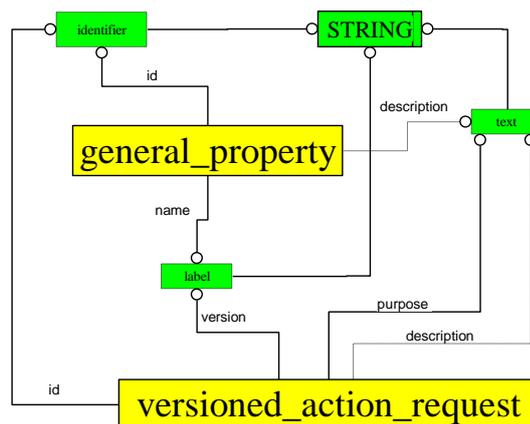


Рис. 25

Обязательные атрибуты изображаются на схеме сплошной линией, необязательные (в объявлении которых используется ключевое слово *OPTIONAL*) изображаются пунктирной линией (см. атрибут *description* в определениях обеих сущностей). Или, учитывая традицию не изображать на схеме простейшие типы данных:

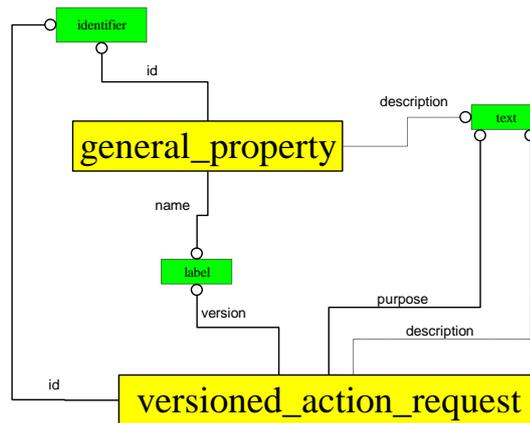


Рис. 26

В языке EXPRESS, как это принято во всех использующих объектный подход формальных языках, одна из сущностей может быть подтипом другой, наследуя при этом все атрибуты и методы своего предка. Поскольку EXPRESS не является языком программирования, методы классов информации в общепринятом виде в EXPRESS отсутствуют. Методами сущностей в EXPRESS можно считать производные атрибуты и местные правила.

Это – простейший вид наследования, пример которого приведен в схеме `sample_geometry`:

```
ENTITY point;
  coords : ARRAY [1:3] OF REAL;
END_ENTITY;
```

```
ENTITY colour_point SUBTYPE OF (point);
  colour : point_colour;
END_ENTITY;
```

Отношение сущности с ее предками и, возможно, потомками, записывается в заголовке определения сущности.

```
sub_super = [ supertype_constraint ] [ subtype_declaration ]. (WSN-105)
```

Для той сущности, которая является подтипом другой сущности, обязательно пишется информация о ее супертипах. Это необходимо для однозначного определения сущности, поскольку сущность наследует атрибуты и правила своих супертипов.

Одна сущность может иметь несколько супертипов, это задается простым перечислением всех супертипов сущности. Это соответствует множественному наследованию, которое является обычным в объектном подходе.

```
subtype_declaration = 'SUBTYPE' 'OF' '(' entity_ref { ',' entity_ref } ')'. (WSN-106)
```

В нотации EXPRESS-G отношение наследования изображается жирной линией, направленной от предка (супертипа) к потомку (подтипу):

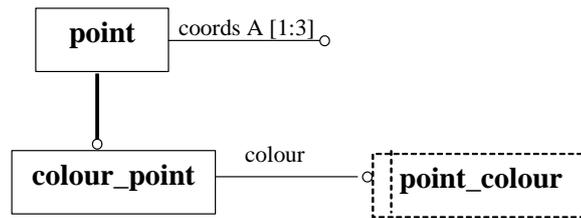


Рис. 27

Рис. Обозначение в EXPRESS-G отношения подтип-супертип

Согласно положениям объектного подхода, экземпляр класса (в STEP – экземпляр сущности) является одновременно экземплярами всех сущностей-предков.

Для родительской сущности соответствующая запись SUPERTYPE, описывающая отношения с ее подтипами (т.е. ограничения, накладываемые на наследование) является необязательной. Раскроем синтаксис ограничений на наследование:

supertype_constraint = abstract_supertype_declaration | supertype_rule. (WSN-107)

Эта запись появляется в следующих случаях:

Для задания того, что сущность является абстрактным супертипом (ABSTRACT SUPERTYPE). Это означает, что не может существовать экземпляров сущности именно этого типа, могут существовать только экземпляры сущностей типов, являющихся потомками этого типа. Например, не может существовать экземпляра сущности типа *млекопитающее*, может быть только экземпляр сущности типа *кошка*, *собака*, *человек* и т.д.. Как видно из определения, сущность может быть объявлена абстрактным супертипом, и при этом могут быть перечислены ее подтипы:

abstract_supertype_declaration = 'ABSTRACT' 'SUPERTYPE' [subtype_constraint]. (WSN-108)

Например, сущность *date_and_time_assignment*, т.е. присвоение даты и времени в том виде, как она определена, сама по себе бессмысленна, т.к. посредством имеющихся у сущности атрибутов нельзя указать, чему присваиваются дата и время:

```
ENTITY date_and_time_assignment
  ABSTRACT SUPERTYPE;
  assigned_date_and_time : date_and_time;
  role : date_time_role;
END_ENTITY; -- 10303-41: management_resources_schema
```

Эта сущность имеет смысл только тогда, когда она представлена своими подтипами, например:

```
ENTITY applied_date_and_time_assignment
```

```

SUBTYPE OF (date_and_time_assignment);
items : SET[1:?] OF date_and_time_item;
END_ENTITY; -- 10303-214: automotive_design

```

Для задания сложных отношений наследования, с использованием ключевых слов AND, ANDOR, ONEOF;

supertype_rule = 'SUPERTYPE' subtype_constraint. (WSN-109)

subtype_constraint = 'OF' '(' supertype_expression ')'. (WSN-110)

supertype_expression = supertype_factor { 'ANDOR' supertype_factor }. (WSN-111)

supertype_factor = supertype_term { 'AND' supertype_term }. (WSN-112)

supertype_term = entity_ref | 'ONEOF' | '(' supertype_expression ')'. (WSN-113)

Для справки.

Экземпляр сущности может быть одновременно экземплярами нескольких терминальных подтипов одного супертипа. Эта возможность имеется по умолчанию, она может быть отменена посредством оператора ONEOF. В EXPRESS-G оператор ONEOF отображается цифрой «1» у корня той линии отношения подтип-супертип, которая ведет от супертипа к тем подтипам, на которые наложено ограничение. В объектном подходе такая возможность называется “множественная классификация данных”. Экземпляр сущности является экземпляром построенной с использованием множественной классификации данных составной сущности тогда и только тогда, когда он является экземпляром двух или более листовых подтипов.

Большинством формальных языков такая возможность не поддерживается.



Рис. 28. Множественная классификация данных

Оператор ANDOR может использоваться только в комбинации с другими операторами для задания сложных комбинаций типов. К примеру, в приведенной схеме сущность станок может быть определена:

ENTITY станок

SUPERTYPE OF (ONEOF(токарный_станок, фрезерный_станок, сверлильный_станок) ANDOR ONEOF(станок_с_ручным_управлением, станок_полуавтомат, станок_автомат))

SUBTYPE OF (изделие);

END_ENTITY;

Заметим, что в данном примере для наглядности мы отступили от правила, согласно которому в EXPRESS может использоваться только латиница. В связи с этим, рассматривая данный пример, следует понимать, что он не подходит для программной обработки.

Наличие ветвления цепочки наследования, сходящееся на более нижних уровнях иерархии к единичной терминальной простейшей сущности, не является множественной классификацией. Например:

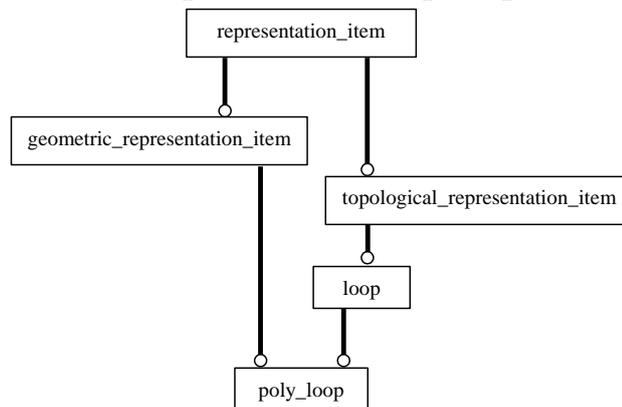


Рис. 29. Наличие одного терминального подтипа не является случаем множественной классификации данных

Контрольные вопросы

8. Какие простейшие типы данных существуют в языке EXPRESS.
9. Что такое множественная классификация данных.

Рассмотрим следующий тип данных, который может использоваться как для задания области допустимых значений атрибутов, так и в качестве основы для определенного в схеме типа общего вида. Это – агрегатный тип данных. Агрегатный тип данных – это коллекция некоторых элементов. В отличие от простейших типов данных, агрегатный тип данных всегда должен объявляться. EXPRESS предусматривает четыре разновидности агрегатных типов, каждая из которых может иметь неограниченную глубину вложенности, т.е. элементы агрегатного типа данных сами могут быть агрегатными типами.

aggregation_types = array_type | bag_type | list_type | set_type. (WSN-114)

Каждая из разновидностей агрегатных типов имеет свои особенности. К таким особенностям относятся:

Упорядоченность. Элемент упорядоченного агрегатного типа имеет позицию (индекс), по которой может быть осуществлен доступ к этому элементу. Доступ к элементам неупорядоченного агрегатного типа возможен с помощью специальной процедуры, использующей итераторы. Итератор обеспечивает последовательный доступ к каждому из элементов агрегатного типа, доступ к каждому элементу осуществляется строго один раз, и при этом последовательность, в которой будут появляться элементы, непредсказуема.

Уникальность. Если агрегат является уникальным, в нем не может быть двух элементов с одинаковыми значениями.

Тип	Упорядоченность	Уникальность	Смысл границ	Обязательность наличия границ	Пример
массив ARRAY	Да, позиции элементов фиксированы, допускаются пустые элементы.	Может быть задана оператором UNIQUE	Нижний индекс; Верхний Индекс (допускаются отрицательные значения индексов)	Да	Зрители в кинотеатре. Каждый зритель имеет свою позицию или индекс – номер кресла. Позиция остается неизменной, некоторые из кресел могут оставаться пустыми. Количество позиций задано изначально и меняться не может. Нумерация кресел в некоторых случаях может начинаться не с 1.
список LIST	Да, позиции элементов переменные, пустые элементы не допускаются.		Наименьшее допустимое количество элементов; Наибольшее допустимое количество элементов;	Нет	Очередь в буфет. По мере обслуживания покупателей их позиция (индекс) постоянно меняется, но при этом последовательность (кто за кем) сохраняется. Покупатели могут покинуть очередь или влезть в очередь, при этом позиции всех последующих покупателей меняются.
набор SET	Нет	Всегда			Люди, едущие в лифте. Каждый человек занимает в лифте строго одно место (уникальность). Люди могут входить и выходить на любых этажах в произвольной последовательности, без какой-либо очередности.
множество BAG	Нет	Никогда			Звери, проживающие в лесу. Если нас не интересуют уникальные свойства каждой особи, то один зверь, например, заяц, волк, лиса, может входить во множество произвольное количество раз.

Особенности различных разновидностей агрегатных типов приводят к тому, что синтаксис их объявления несколько различается. Синтаксис объявления массива:

array_type = 'ARRAY' bound_spec 'OF' ['OPTIONAL'] ['UNIQUE'] base_type. (WSN-115)

Как видно из синтаксиса, для массива задание границ является обязательным. Уникальность элементов массива может быть задана с помощью ключевого слова *UNIQUE*. Если в массиве допускается наличие пустых позиций, это указывается посредством ключевого слова *OPTIONAL*. Синтаксис объявления списка:

list_type = 'LIST' [bound_spec] 'OF' ['UNIQUE'] base_type. (WSN-116)

Синтаксис объявлений набора *set* и множества *bag* идентичен. Их различия (см. таблицу выше) в синтаксисе не отражаются:

bag_type = 'BAG' [bound_spec] 'OF' base_type. (WSN-117)

set_type = 'SET' [bound_spec] 'OF' base_type. (WSN-118)

Во всех случаях *base_type* – это область определения элемента агрегата. Определение *base_type* уже приводилось выше. Поскольку *base_type* может быть, в том числе, и агрегатом, агрегаты могут иметь произвольную глубину вложенности. Для многомерных (вложенных) агрегатных типов допускается использование нескольких видов агрегатов, например, *ARRAY OF BAG OF LIST, SET OF LIST* и т.д.

Определение границ:

bound_spec = '[' bound_1 ':' bound_2 ']'. (WSN-119)

bound_2 = numeric_expression. (WSN-120)

bound_1 = numeric_expression. (WSN-121)

Как и в рассмотренном выше случае с заданием длины строки и точности вещественного числа, при задании границ агрегатного типа данных также могут использоваться числовые выражения произвольной сложности. На практике используется следующее:

Нижняя граница – число;

Верхняя граница – число, или, в случае неопределенной верхней границы – знак «?».

Например,

ARRAY [-1:3] OF INTEGER;

LIST [0:5] OF REAL;

SET [2:?] OF point;

BAG [3:100] OF knot;

Или, в составе сущности:

ENTITY action_directive;

name : label;

description : text;

```

analysis : text;
comment : text;
requests : SET [1:?] OF versioned_action_request;
END_ENTITY;

```

Если границы не указаны, по умолчанию считается, что агрегат имеет границы [0:?].

В схеме в нотации EXPRESS-G агрегатные типы изображаются только вместе с именем соответствующего атрибута в сокращенном виде – первая буква разновидности агрегата (A, L, S или B) и значения границ, примеры см. ниже.

В обменном файле STEP агрегаты имеют одинаковый синтаксис:

```
list = (' [ parameter { ',' parameter } ] '). (WSN-122)
```

Следующие примеры скопированы без изменений из прикладных протоколов. Некоторые из используемых элементов в определениях сущностей пока еще не рассматривались, но они для данных примеров никакой роли не играют. Чтобы эти элементы не отвлекали внимание, они выделены заливкой. Например, вектор узлов и управляющие точки b-сплайновой кривой – это списки *list*:

```

ENTITY representation_item;
name : label;
WHERE
wr1: SIZEOF(using_representations(SELF)) > 0;
END_ENTITY; -- representation_item

```

```

ENTITY geometric_representation_item
SUPERTYPE OF (ONEOF (point,direction,vector,placement,
cartesian_transformation_operator,curve,surface,edge_curve,
face_surface,poly_loop,vertex_point,solid_model,
shell_based_surface_model,shell_based_wireframe_model,
edge_based_wireframe_model,geometric_set))
SUBTYPE OF (representation_item);
DERIVE
dim : dimension_count := dimension_of(SELF);
WHERE
wr1: SIZEOF(QUERY ( using_rep <* using_representations(SELF) | (NOT
('PROCESS_PLANNING_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT'
IN TYPEOF(using_rep.context_of_items))) ) = 0;
END_ENTITY; -- geometric_representation_item

```

```

ENTITY curve
SUPERTYPE OF (ONEOF (line,conic,pcurve,surface_curve,offset_curve_3d,
curve_replica))
SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- curve

```

```

ENTITY bounded_curve
SUPERTYPE OF (ONEOF (polyline,b_spline_curve,trimmed_curve,
bounded_pcurve,bounded_surface_curve,composite_curve))
SUBTYPE OF (curve);
END_ENTITY; -- bounded_curve

```

```

ENTITY b_spline_curve
SUPERTYPE OF (ONEOF (uniform_curve,b_spline_curve_with_knots,
quasi_uniform_curve,bezier_curve) ANDOR rational_b_spline_curve)
SUBTYPE OF (bounded_curve);
degree : INTEGER;
control_points_list : LIST [2:?] OF cartesian_point;
curve_form : b_spline_curve_form;
closed_curve : LOGICAL;
self_intersect : LOGICAL;

```

DERIVE

```

upper_index_on_control_points : INTEGER := SIZEOF(
control_points_list) - 1;
control_points : ARRAY [0:
upper_index_on_control_points] OF
cartesian_point := list_to_array(
control_points_list,0,
upper_index_on_control_points);

```

WHERE

```

wr1: (('PROCESS_PLANNING_SCHEMA.UNIFORM_CURVE' IN TYPEOF(SELF)) OR
('PROCESS_PLANNING_SCHEMA.QUASI_UNIFORM_CURVE' IN TYPEOF(
SELF))) OR ('PROCESS_PLANNING_SCHEMA.BEZIER_CURVE' IN
TYPEOF(SELF))) OR (
'PROCESS_PLANNING_SCHEMA.B_SPLINE_CURVE_WITH_KNOTS' IN
TYPEOF(SELF));
END_ENTITY; -- b_spline_curve

```

```

ENTITY b_spline_curve_with_knots

```

```

SUBTYPE OF (b_spline_curve);
knot_multiplicities : LIST [2:?] OF INTEGER;
knots : LIST [2:?] OF parameter_value;
knot_spec : knot_type;
DERIVE
upper_index_on_knots : INTEGER := SIZEOF(knots);
WHERE
wr1: constraints_param_b_spline(degree,upper_index_on_knots,
upper_index_on_control_points,knot_multiplicities,knots);
wr2: SIZEOF(knot_multiplicities) = upper_index_on_knots;
END_ENTITY; -- b_spline_curve_with_knots

```

В обменном файле

```

#108=B_SPLINE_CURVE_WITH_KNOTS("3,(#99,#100,#101,#102,#103,#104,#105,#106,
#107),.UNSPECIFIED.,.F.,.F.,(4,1,1,1,1,1,4),(0.E0,1.6666666666667E-1,3.333333333333E-
1,5.E-1,6.6666666666667E-1,8.333333333333E-1,1.E0),
.UNSPECIFIED.);

```

В замкнутой оболочке грани представлены в виде набора *set*:

```

ENTITY representation_item;
name : label;
WHERE
wr1: SIZEOF(using_representations(SELF)) > 0;
END_ENTITY; -- representation_item

```

```

ENTITY topological_representation_item
SUPERTYPE OF (ONEOF (vertex,edge,face_bound,face,vertex_shell,
wire_shell,connected_edge_set,connected_face_set,loop ANDOR path))
SUBTYPE OF (representation_item);
END_ENTITY; -- topological_representation_item

```

```

ENTITY connected_face_set
SUPERTYPE OF (ONEOF (closed_shell,open_shell))
SUBTYPE OF (topological_representation_item);
cfs_faces : SET [1:?] OF face;
END_ENTITY; -- connected_face_set

```

```

ENTITY closed_shell
SUBTYPE OF (connected_face_set);

```

END_ENTITY; -- closed_shell

В обменном файле соответствующий параметр представлен с применением тех же синтаксических правил:

```
#1396=CLOSED_SHELL(",(#619,#634,#656,#676,#690,#710,#724,#738,#765,#792,#810,#826,#840,#856,#869,#905,#920,#937,#951,#963,#975,#990,#1007,#1021,#1033,#1045,#1060,#1077,#1091,#1103,#1115,#1130,#1147,#1161,#1173,#1185,#1224,#1263,#1300,#1337,#1353,#1368,#1383,#1395));
```

На значения одного или совокупности атрибутов сущности может быть наложено ограничение уникальности. Чаще всего это применяется к внешним идентификаторам, например, обозначениям изделий, версий изделий, ресурсов, технологических операций и т.д. Синтаксис задания правил уникальности:

unique_clause = 'UNIQUE' unique_rule ';' {unique_rule ';' }. (WSN-123)

unique_rule = [label ':'] referenced_attribute { ';' referenced_attribute }. (WSN-124)

referenced_attribute = attribute_ref | qualified_attribute. (WSN-125)

label = simple_id. (WSN-126)

Например, обозначение версии изделия `product_definition_formation` уникально среди всех версий одного изделия:

ENTITY product;

id : identifier;

name : label;

description : OPTIONAL text;

frame_of_reference : SET[1:?] OF product_context;

END_ENTITY; -- 10303-41: product_definition_schema

ENTITY product_definition_formation;

id : identifier;

description : OPTIONAL text;

of_product : product;

UNIQUE

ur1 : id, of_product;

END_ENTITY; -- product_definition_formation

Понятно, что в обменном файле правила уникальности никак не могут быть определены.

Создание концептуальных схем данных также является предметной областью, хотя и узкой, редкой и специфичной. Программные продукты, работающие в этой области – это различные универсальные программные инструменты: обработчики (компиляторы) концептуальных схем данных, средства инвариантного анализа данных, средства доступа к данным и т.д. Для этой предметной области может быть разработана своя концептуальная схе-

ма данных. Сущностями этой концептуальной схемы данных будут сущности, их свойства (атрибуты), правила, типы данных. В STEP такая концептуальная схема данных разработана и стандартизована. Спецификация этой схемы, имеющей название *sdai_dictionary_schema*, содержится в томе ISO 10303-22.

В данном случае денотатом является EXPRESS-схема, что можно представить следующей схемой:

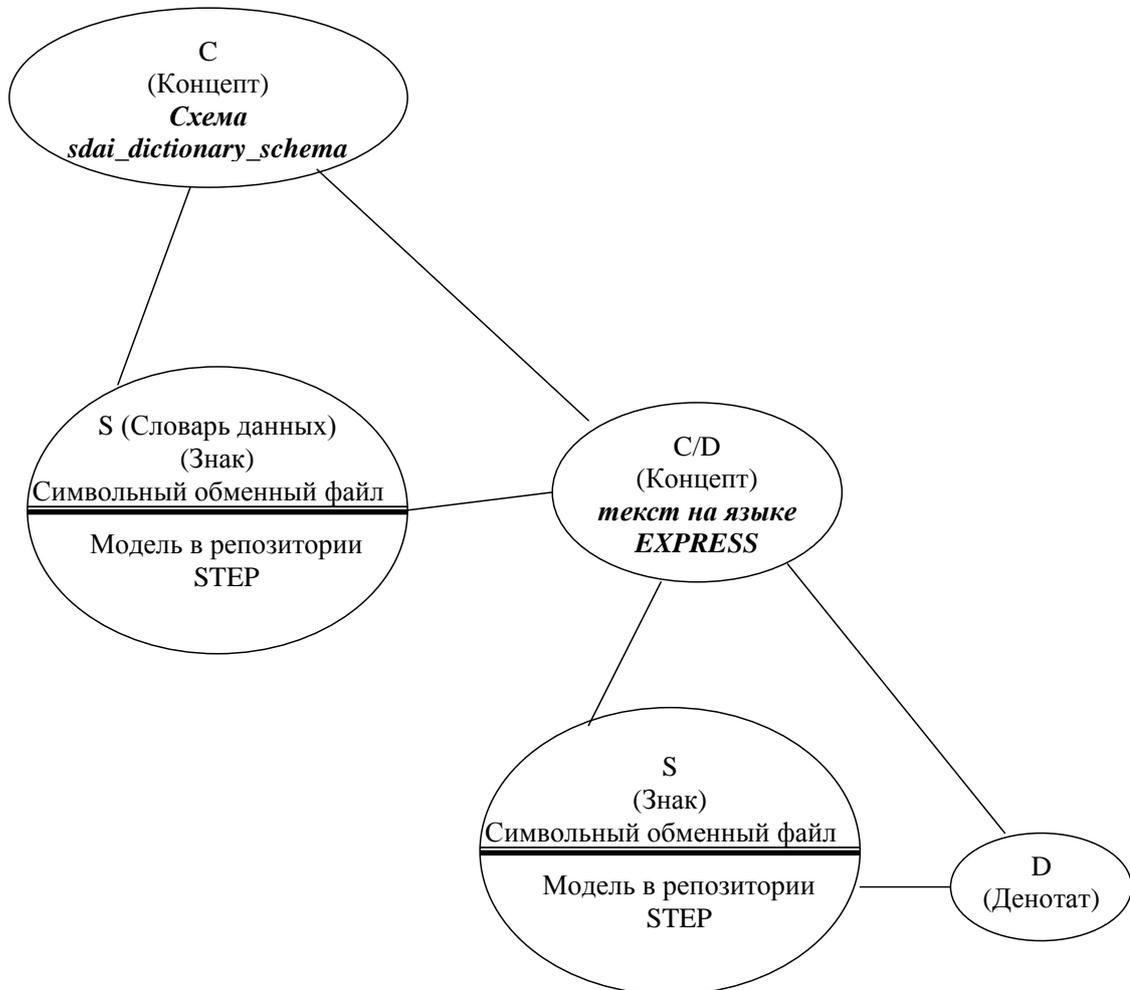


Рис. 30

Изучение схемы словарей данных, отражающей синтаксис языка EXPRESS, позволяет более наглядно представить отношения синтаксических элементов языка. Поскольку ключевым понятием в языке EXPRESS является сущность, с нее и начнем. В концептуальной схеме словарей данных *sdai_dictionary_schema* предусмотрена сущность, отображающая сущности словаря. Обозначение (идентификатор) определения этой сущности – *entity_definition*.

ENTITY *named_type*

ABSTRACT SUPERTYPE OF (ONEOF(entity_definition, defined_type));

name : **STRING**;

where_rules : **SET [0:?] OF where_rule**;

END_ENTITY;

```

ENTITY entity_definition
  SUBTYPE OF (named_type);
  supertypes : LIST [0:?] OF UNIQUE entity_definition;
  attributes : LIST [0:?] OF UNIQUE attribute;
  uniqueness_rules : SET [0:?] OF uniqueness_rule;
  complex : BOOLEAN;
  instantiable : BOOLEAN;
  independent : BOOLEAN;
  parent_schema : schema_definition;
UNIQUE
  UR1: name, parent_schema;
END_ENTITY;

```

Из определения видно, что в атрибутах данной сущности используются две разновидности агрегатов: множество *set* для местных правил и правил уникальности и список *list* для супертипов и атрибутов сущностей. Это объясняется тем, что правила неупорядочены и их проверка может выполняться в совершенно произвольной последовательности, а порядок следования атрибутов необходим для правильного заполнения полей параметров в символическом обменном файле.

Определенный в схеме тип *defined_type* и определение сущности *entity_definition* имеют общий супертип – именованный тип *named_type*. Назначение большей части атрибутов видно из их названия, поэтому остановимся на тех атрибутах, назначение которых неочевидно.

Атрибут *complex* (сложный) показывает, используется ли в данной сущности множественная классификация данных. Поскольку количество вариантов сущностей, включающих более одного терминального определения сущности (т.е. сложных сущностей), может значительно превышать число простых сущностей, предусматривается, что определения сложных сущностей при компиляции EXPRESS-схемы в словарь не заносятся, а создаются динамически по мере необходимости.

Атрибут *insatiable* (способный создавать экземпляры) показывает, является ли сущность абстрактным супертипом.

Атрибут *independent* (независимый) показывает, могут ли для данной сущности создаваться независимые экземпляры (т.е. экземпляры, на которые нет ссылок из других экземпляров сущностей). Это связано с тем, что при импорте определения сущности с помощью спецификации *REFERENCE* для такой сущности не могут создаваться независимые экземпляры.

Инверсные атрибуты (INVERSE) - используются для обратной связи с другими сущностями. Это то, что называется обычно “BACK POINTER” (обратный указатель). Назначение инверсных атрибутов – избежать взаимных ссылок экземпляров сущностей. Вообще в языке EXPRESS не существует прямого запрета на взаимные ссылки, но в реальных схемах Приклад-

ных протоколов STEP взаимные ссылки практически отсутствуют, поскольку они являются при реализации источником потенциальных ошибок. Предполагается, что обратный указатель формируется автоматически при появлении соответствующего прямого указателя и его актуальность поддерживается средствами реализации системы.

Объявление инверсных атрибутов объединено в блок.

inverse_clause = 'INVERSE' inverse_attr { inverse_attr }. (WSN-127)

Объявление инверсного атрибута имеет следующий синтаксис:

inverse_attr = attribute_decl ':' [('SET' | 'BAG') [bound_spec] 'OF' entity_ref 'FOR' attribute_ref ';']. (WSN-128)

При определении инверсного атрибута указывается соответствующий ему прямой атрибут. Областью допустимых значений инверсного атрибута является сущность того типа, который содержит прямой атрибут. Инверсный атрибут может быть и простым, и агрегатным. Это зависит только от того, могут ли ссылаться на экземпляр сущности данного типа (т.е. сущности того типа, который содержит инверсный атрибут) несколько сущностей того типа, который содержит прямой атрибут. Допускаются агрегаты типа *set* (множество) если прямой атрибут может ссылаться на экземпляр сущности, имеющей инверсный атрибут не более одного раза и *bag* (множество), если число ссылок неограниченно.

Сочетание элементов *entity_ref* и *attribute_ref* однозначно определяют соответствующий прямой атрибут.

В обменном файле значения инверсных атрибутов не заносятся, т.к. эти значения являются избыточной информацией.

Правило уникальности представляется сущностью

```
ENTITY uniqueness_rule;
  label : OPTIONAL STRING;
  attributes : LIST [1:?] OF attribute;
INVERSE
  parent_entity : entity_definition FOR uniqueness_rules;
UNIQUE
  UR1: label, parent_entity;
END_ENTITY;
```

Очевидно, атрибуты могли бы быть представлены и набором *set*, но в схеме используется именно такое определение.

Местные правила в словаре представляются сущностью

```
ENTITY where_rule;
  label : OPTIONAL STRING;
INVERSE
  parent_type : named_type FOR where_rules;
UNIQUE
```

UR1: label, parent_type;

END_ENTITY;

Как видно из определения, представление алгоритма проверки правила не стандартизовано и оставлено на усмотрение разработчиков реализации SDAI.

Рассмотрим определения атрибутов в словаре данных. Все три приведенных выше разновидности атрибутов имеют общего предка – сущность `attribute`, обладающий двумя общими для всех атрибутов свойствами: обозначением и наличием сущности, к которой атрибут относится:

ENTITY attribute

ABSTRACT SUPERTYPE OF (ONEOF(derived_attribute, explicit_attribute, inverse_attribute));

name : STRING;

INVERSE

parent_entity : entity_definition FOR attributes;

UNIQUE

UR1: name, parent_entity;

END_ENTITY;

Как видно, сущность указывается через инверсный атрибут, поскольку сущность уже ссылается на атрибут, и тут мы как раз и имеем дело с перекрестными ссылками, наличие которых разработчики STEP всегда стараются избежать.

Во всех трех вышеприведенных примерах для инверсных атрибутов не указываются тип агрегата и границы, т.е. из объявлений видно, что:

- один экземпляр определения правила уникальности может относиться только к одному экземпляру определения сущности;

- один экземпляр определения местного правила может относиться только к одному экземпляру определения именованного типа (который может быть экземпляром определенного в схеме типа или экземпляром определения сущности);

- один экземпляр определения атрибута (который может быть экземпляром определения явного, производного или инверсного атрибута) может относиться только к одному определению сущности.

Возможны и другие варианты, когда одна сущность «документ» может иметь множество представлений (пример взят из Протокола AP240):

ENTITY document;

id : identifier;

name : label;

description : OPTIONAL text;

kind : document_type;

INVERSE

```

representation_types : SET [0:?] OF document_representation_type FOR
represented_document;
END_ENTITY; -- document

```

```

ENTITY document_representation_type;
name : label;
represented_document : document;
END_ENTITY; -- document_representation_type

```

```

ENTITY document_type;
product_data_type : label;
END_ENTITY; -- document_type

```

Атрибут сущности может переопределять атрибут супертипа-предка сущности. При этом атрибут сущности-предка отменяется и заменяется атрибутом сущности-потомка.

Количество атрибутов (а, следовательно, и полей параметров в записи в обменном файле) при этом уменьшается. Естественно, что переменное количество полей значительно усложнило бы разработку программных продуктов, считывающих обменные файлы STEP, поэтому, чтобы не менять количество полей в записи, предусмотрено определение пропущенного параметра:

```
omitted_parameter = '*'. (WSN-129)
```

Уже упомянутый ранее производный атрибут имеет следующее синтаксическое определение:

```
derive_clause = 'DERIVE' derived_attr { derived_attr }. (WSN-130)
```

```
derived_attr = attribute_decl ':' base_type ':=' expression ';'. (WSN-131)
```

Здесь **expression** (выражение) это и есть алгоритм вычисления значения производного атрибута:

```
expression = simple_expression [ rel_op_extended simple_expression ]. (WSN-132)
```

```
simple_expression = term { add_like_op term }. (WSN-133)
```

```
rel_op_extended = rel_op | 'IN' | 'LIKE'. (WSN-134)
```

```
rel_op = '<' | '>' | '<=' | '>=' | '<>' | '=' | ':<:' | ':=:'. (WSN-135)
```

```
term = factor { multiplication_like_op factor }. (WSN-136)
```

```
add_like_op = '+' | '-' | 'OR' | 'XOR'. (WSN-137)
```

```
multiplication_like_op = '*' | '/' | 'DIV' | 'MOD' | 'AND' | '||'. (WSN-138)
```

```
factor = simple_factor [ '**' simple_factor ]. (WSN-139)
```

```
simple_factor = aggregate_initializer | entity_constructor | enumeration_reference | interval | query_expression | ( [ unary_op ] ( '(' expression ')' | primary ) ). (WSN-140)
```

В словаре данных объявление производного атрибута отображается в сущность `derived_attribute`:

```
ENTITY derived_attribute
  SUBTYPE OF (attribute);
  domain : base_type;
  redeclaring : OPTIONAL explicit_or_derived;
END_ENTITY;
```

Как видно из определения производного атрибута, алгоритм его вычисления, так же как и для правил, не отображается в стандартизованное представление словаря данных. Решение этого вопроса оставлено на усмотрение разработчиков реализации.

В нотации EXPRESS-G производные атрибуты отображаются так же, как и явные, но с префиксом (*DER*):

Из-за своей избыточности, значения производных атрибутов в обменный файл не заносятся.

```
ENTITY explicit_attribute
  SUBTYPE OF (attribute);
  domain : base_type;
  redeclaring : OPTIONAL explicit_attribute;
  optional_flag : BOOLEAN;
END_ENTITY;
```

```
ENTITY inverse_attribute
  SUBTYPE OF (attribute);
  domain : entity_definition;
  redeclaring : OPTIONAL inverse_attribute;
  inverted_attr : explicit_attribute;
  min_cardinality : OPTIONAL INTEGER;
  max_cardinality : OPTIONAL INTEGER;
  duplicates : BOOLEAN;
```

WHERE

`role_invertible: inherits_from(domain, inverted_attr.parent_entity);`

`uniqueness_correct: (max_cardinality > 1) OR NOT duplicates;`

`not_negative_max: max_cardinality >= 0;`

`not_negative_min: min_cardinality >= 0;`

`valid_boundaries: min_cardinality <= max_cardinality;`

```
END_ENTITY;
```

Как видно из приведенных определений, состав схемы словарей в первом приближении схож с синтаксической нотацией языка.

select_type = SELECT '(' named_types { ',' named_types } ')'. (WSN-141)

Выбираемый тип данных SELECT.

enumeration_type = ENUMERATION OF '(' enumeration_id { ',' enumeration_id } ')'.
(WSN-142)

typed_parameter = keyword '(' parameter ')'. (WSN-143)

Параметр перечислимого (enumeration) типа представляется в обменном файле следующей конструкцией:

enumeration = '.' upper { upper | digit | '_' } '.'. (WSN-144)

Исполняемые операторы языка EXPRESS.

Эти операторы используются в определении производных (вычисляемых) атрибутов и в определении правил. Возможно также применение исполняемых операторов при задании мощности агрегатных типов данных и при задании длин строковых и вещественных типов данных. В стандартном словаре данных исполняемые операторы не отражены, следовательно, их реализация остается на усмотрение разработчика реализации STEP.

Ниже приведены определения WSN, используемые для записи исполняемых операторов языка EXPRESS. Эти определения подробно рассматриваться не будут.

actual_parameter_list = '(' parameter { ',' parameter } ')'. (E001)
aggregate_initializer = '[' [element { ',' element }]]'. (E003)
aggregate_source = simple_expression. (E004)
aggregate_type = 'AGGREGATE' [';' type_label] 'OF' parameter_type. (E005)
algorithm_head = { declaration } [constant_decl] [local_decl]. (E007)
alias_stmt = 'ALIAS' variable_id 'FOR' general_ref { qualifier } ';' stmt { stmt } 'END_ALIAS' ';' . (E008)
assignment_stmt = general_ref { qualifier } ':=' expression ';' . (E010)
binary_literal = '%' bit { bit } . (E017)
bit = '0' | '1' . (E019)
built_in_constant = 'CONST_E' | 'PI' | 'SELF' | '?' . (E024)
built_in_function = 'ABS' | 'ACOS' | 'ASIN' | 'ATAN' | 'BLENGTH' | 'COS' | 'EXISTS' | 'EXP' | 'FORMAT' | 'HIBOUND' | 'HIINDEX' | 'LENGTH' | 'LOBOUND' | 'LOINDEX' | 'LOG' | 'LOG2' | 'LOG10' | 'NVL' | 'ODD' | 'ROLESOF' | 'SIN' | 'SIZEOF' | 'SQRT' | 'TAN' | 'TYPEOF' | 'USEDIN' | 'VALUE' | 'VALUE_IN' | 'VALUE_UNIQUE' . (E025)
built_in_procedure = 'INSERT' | 'REMOVE' . (E026)
case_action = case_label { ',' case_label } ':' stmt . (E027)
case_label = expression . (E028)
case_stmt = 'CASE' selector 'OF' [case_action] ['OTHERWISE' ':' stmt] 'END_CASE' ';' . (E029)
compound_stmt = 'BEGIN' stmt { stmt } 'END' ';' . (E030)
constant_body = constant_id ':' base_type ':=' expression ';' . (E031)
constant_decl = 'CONSTANT' constant_body { constant_body } 'END_CONSTANT' ';' . (E032)
constant_factor = built_in_constant | constant_ref . (E033)
domain_rule = [label ':'] logical_expression . (E042)

element = expression [':' repetition] . (E043)
 encoded_character = octet octet octet octet . (E045)
 encoded_string_literal = "" encoded_character { encoded_character } "" . (E046)
 entity_constructor = entity_ref '(' [expression { ',' expression }] ')'. (E048)
 escape_stmt = 'ESCAPE' ';' . (E057)
 formal_parameter = parameter_id { ',' parameter_id } ':' parameter_type . (E061)
 function_call = (built_in_function | function_ref) [actual_parameter_list] . (E062)
 function_decl = function_head [algorithm_head] stmt { stmt } 'END_FUNCTION' ';' .
 (E063)
 function_head = 'FUNCTION' function_id ['(' formal_parameter { ',' formal_parameter
 } ')'] ':' parameter_type ';' . (E064)
 general_aggregation_types = general_array_type | general_bag_type | general_list_type | gen-
 eral_set_type . (E067)
 general_array_type = 'ARRAY' [bound_spec] 'OF' ['OPTIONAL'] ['UNIQUE']
 parameter_type . (E068)
 general_bag_type = 'BAG' [bound_spec] 'OF' parameter_type . (E069)
 general_list_type = 'LIST' [bound_spec] 'OF' ['UNIQUE'] parameter_type . (E070)
 general_set_type = 'SET' [bound_spec] 'OF' parameter_type . (E072)
 generalized_types = aggregation_type | general_aggregation_types | generic_type . (E073)
 generic_type = 'GENERIC' [':' type_label] . (E074)
 hex_digit = digit | 'a' | 'b' | 'c' | 'd' | 'e' | 'f' . (E076)
 if_stmt = 'IF' logical_expression 'THEN' stmt { stmt } ['ELSE' stmt { stmt }]
 'END_IF' ';' . (E077)
 increment = numeric..expression . (E078)
 increment_control = variable_id ':=' bound_1 'TO' bound_2 ['BY' increment] . (E079)
 index = numeric_expression . (E080)
 index_2 = index . (E081)
 index_1 = index . (E082)
 index_qualifier = '[' index_1 [':' index_2] ']'. (E083)
 interval = '{' interval_low interval_op interval_item interval_op interval_high
 '}' . (E087)
 interval_high = simple_expression . (E088)
 interval_item = simple_expression . (E089)
 interval_low = simple_expression . (E090)
 interval_op = '<' | '<=' . (E091)
 local_decl = 'LOCAL' local_variable { local_variable } 'END_LOCAL' ';' . (E098)
 local_variable = variable_id { ',' variable_id } ':' parameter_type [':=' expression] ';' .
 (E099)
 logical_expression = expression . (E100)
 logical_literal = 'FALSE' | 'TRUE' | 'UNKNOWN' . (E101)
 lparen_not_star = '(' not_star . (E103)
 not_lparen_star = not_paren_star | ')' . (E107)
 not_paren_star_special = not_paren_star_quote_special | '"""' . (E108)
 not_paren_star = letter | digit | not_paren_star_special . (E109)
 not_paren_star_quote_special = '|' | '"' | '#' | '\$' | '%' | '&' | '+' | ',' | '-' | '.' | '/' | ':' | ';' | '<' | '=' | '>' |
 '?' | '@' | '[' | '\' | ']' | '^' | '_' | '~' | '{' | '|' | '}' | '~' . (E110)
 not_quote = not_paren_star_quote_special | letter | digit | '(' | ')' . (E111)
 not_rparen = not_paren_star | '*' | '(' . (E112)
 not_star = not_paren_star | '(' | ')' . (E113)
 null_stmt = ';' . (E114)
 octet = hex_digit hex_digit . (E117)
 one_of = 'ONEOF' '(' supertype_expression { ',' supertype_expression } ')'. (E118)

`parameter_type = generalized_types | named_types | simple_types . (E122)`
`population = entity_ref . (E123)`
`procedure_call_stmt = (built_in_procedure | procedure_ref) [actual_parameter_list] ';' . (E126)`
`procedure_decl = procedure_head [algorithm_head] { stmt } 'END_PROCEDURE' ';' . (E127)`
`procedure_head = 'PROCEDURE' procedure_id ['(' ['VAR'] formal_parameter { ';' ['VAR] formal_parameter } ')'] ';' . (E128)`
`qualifiable_factor = attribute_ref | constant_factor | function_call | general_ref | population . (E131)`
`qualifier = attribute_qualifier | group_qualifier | index_qualifier . (E133)`
`query_expression = 'QUERY' '(' variable_id '<*' aggregate_source '|' logical_expression ')' . (E134)`
`real_literal = digits '.' [digits] ['e' [sign] digits] . (E135)`
`repeat_control = [increment_control] [while_control] [until_control] . (E143)`
`repeat_stmt = 'REPEAT' repeat_control ';' stmt { stmt } 'END_REPEAT' ';' . (E144)`
`repetition = numeric_expression . (E145)`
`return_stmt = 'RETURN' ['(' expression ')'] ';' . (E148)`
`rule_decl = rule_head [algorithm_head] { stmt } where_clause 'END_RULE' ';' . (E149)`
`rule_head = 'RULE' rule_id 'FOR' '(' entity_ref { ';' entity_ref } ')' ';' . (E150)`
`selector = expression . (E157)`
`simple_string_literal = \q { (\q \q) | not_quote | \s | \o } \q . (E163)`
`skip_stmt = 'SKIP' ';' . (E165)`
`star_not_rparen = '*' not_rparen . (E167)`
`stmt = alias_stmt | assignment_stmt | case_stmt | compound_stmt | escape_stmt | if_stmt | null_stmt | procedure_call_stmt | repeat_stmt | return_stmt | skip_stmt . (E168)`
`string_literal = simple_string_literal | encoded_string_literal . (E169)`
`type_label = type_label_id | type_label_ref . (E184)`
`type_label_id = simple_id . (E185)`
`type_label_ref = type_label_id . (E186)`
`unary_op = '+' | '-' | 'NOT' . (E188)`
`until_control = 'UNTIL' logical_expression . (E192)`
`variable_id = simple_id . (E194)`
`where_clause = 'WHERE' domain_rule ';' { domain_rule ';' } . (E196)`
`while_control = 'WHILE' logical_expression . (E197)`

Спецификация интерфейса между схемами.

Множество определений на языке EXPRESS разбито на отдельные схемы. При этом в языке EXPRESS предусмотрены конструкции, позволяющие сделанные в одной схеме определения постоянных (констант), сущностей, функций, процедур и типов данных использовать в другой схеме, т.е. осуществлять импорт определений, сделанных в другой схеме. При этом возможно переименование определений сущностей и типов данных. В языке EXPRESS предусмотрены две формы импорта:

interface_specification = reference_clause | use_clause . (E086)

Первая из форм – reference (ссылка) предусматривает, что в схеме могут создаваться только зависимые экземпляры импортированных сущностей. Вторая форма – use (использование) позволяет работать с импортированными сущностями так, как если бы они были определены в самой схеме, в кото-

рую был осуществлен импорт. При этом для определений, импортированных посредством спецификации *USE* возможен их последующий импорт в третью схему, т.е. реимпорт. Реимпорт определений, импортированных посредством спецификации *REFERENCE*, невозможен. Синтаксис конструкций сходен. Можно импортировать следующие определения:

Тип определения	Перевод названия	Спецификация интерфейса USE	Спецификация интерфейса REFERENCE
constant	Константа	Нет	Да
entity	Сущность	Да	Да
function	Функция	Нет	Да
procedure	Процедура	Нет	Да
type	Тип данных	Да	Да
rule	Глобальное правило (в нотации Вирта отсутствует, но в тексте упоминается)	Нет	?

Конструкция для импорта посредством спецификации *REFERENCE* (ссылка):

```
reference_clause = 'REFERENCE' 'FROM' schema_ref [ '(' resource_or_rename { ';' resource_or_rename } ')' ] ';'. (E137)
```

Далее раскрыты типы определений, которые можно импортировать. При импорте допускается переименование определения с помощью оператора *AS* (в качестве)

```
resource_or_rename = resource_ref [ 'AS' rename_id ]. (E146)
```

```
rename_id = constant_id | entity_id | function_id | procedure_id | type_id. (E142)
```

Конструкция для импорта посредством спецификации *USE* (использовать):

```
use_clause = 'USE' 'FROM' schema_ref [ '(' named_type_or_rename { ';' named_type_or_rename } ')' ] ';'. (E193)
```

Импортировать можно только поименованные типы, к которым относятся сущностный тип данных и определенный в схеме тип данных. Также, как и в случае спецификации *REFERENCE*, возможно переименование импортированных определений. Переименование также осуществляется посредством оператора *AS* (в качестве):

```
named_type_or_rename = named_types [ 'AS' ( entity_id | type_id ) ]. (E105)
```

В обеих конструкциях список импортируемых определений заключен в квадратные скобки, т.е. является необязательным. Если список импортированных определений отсутствует, импортируются все определения, сделанные в схеме.

Если сущность или определенный в схеме тип импортированы одновременно с использованием спецификаций *USE* и *REFERENCE*, спецификация *USE* имеет приоритет. Импортированные внешние объявления могут ссылаться на идентификаторы, которые не являются видимыми в текущей схеме.

Эти идентификаторы необходимы для полного понимания текущей схемы, и соответствующие им элементы языка EXPRESS импортируются в текущую схему неявным образом. Импортированные неявным образом элементы языка EXPRESS также могут ссылаться на идентификаторы, которые не являются видимыми в текущей схеме, и соответствующие тем идентификаторам элементы языка EXPRESS также импортируются в схему неявным образом.

При импорте констант неявным образом импортируются:

- все определенные в схеме типы, используемые в объявлении импортируемой константы;
- все сущности, используемые в объявлении импортируемой константы;
- все константы, используемые в объявлении импортируемой константы;
- Все функции, используемые в объявлении импортируемой константы.

При импорте определенных в схеме типов данных неявным образом импортируются:

- все определенные в схеме типы, используемые в объявлении импортируемого типа, за исключением того случая, когда импортируемый тип сконструирован как выбираемый (SELECT) тип. В результате импорта выбираемого типа ни один из поименованных типов, входящих в список выбора, неявным образом импортирован не будет;
- все константы и все функции, используемые в объявлении импортируемого типа;
- все константы и все функции, используемые в правилах для областей значений импортируемого типа;
- Все определенные в схеме типы, сконструированные как выбираемый (SELECT) тип, которые в своем списке выбора имеют импортируемый тип.

При импорте сущностей неявным образом импортируются:

- все сущности, которые являются супертипами импортируемой сущности;

ПРИМЕЧАНИЕ - Подтипы импортируемой сущности, неявным образом импортированы не будут, даже если подтипы входят в выражение *SUPERTYPE OF* импортируемой сущности.

- все правила, ссылающиеся на импортируемую сущность и ноль или более других сущностей, все из которых явным или неявным образом импортируются в текущую схему;
- все константы, определенные в схеме типы, сущности и функции, используемые в объявлении атрибутов импортируемой сущности;
- все константы, определенные в схеме типы, сущности и функции, используемые в правилах области значений (WHERE) импортируемой сущности;
- все определенные в схеме типы, построенные как выбираемый (SELECT) тип, имеющие в списке выбора импортируемую сущность.

В результате того, что при неявном импорте учитывается только фраза *SUBTYPE OF* граф отношений подтип/супертип при импорте может быть усечен. Алгоритм, используемый для вычисления допустимых комбинаций

экземпляров сущностей усеченного графа отношений подтип/супертип описан в Приложении С.

При импорте функций неявным образом импортируются:

- все определенные в схеме типы и сущности, используемые в объявлении параметров импортируемой функции;
- все определенные в схеме типы и сущности, используемые в определении возвращаемого значения импортируемой функции;
- все определенные в схеме типы и сущности, используемые в объявлении местных переменных внутри импортируемой функции;
- все константы, функции и процедуры, используемые внутри импортируемой функции.

При импорте процедуры неявным образом импортируются:

- все определенные в схеме типы и сущности, используемые в объявлении параметров импортируемой процедуры;
- все определенные в схеме типы и сущности, используемые в объявлении местных переменных внутри импортируемой функции;
- все константы, функции и процедуры, используемые внутри импортируемой процедуры.

При импорте правила¹ неявным образом импортируются:

- все определенные в схеме типы и сущности, используемые при объявлении местных переменных внутри импортируемого правила;
- все константы, функции и процедуры, используемые внутри импортируемого правила.

В словаре данных спецификация импорта отображается следующей сущностью:

```
ENTITY interface_specification;
  current_schema : STRING;
  explicit_items : SET [1:?] OF explicit_item;
  implicit_items : SET [0:?] OF implicit_item;
END_ENTITY;
```

Здесь `explicit_items` – это множество определений, импортированных явным образом, а `implicit_items` – множество определений, импортированных неявным образом.

```
ENTITY interfaced_item
  ABSTRACT SUPERTYPE OF (ONEOF (explicit_item, implicit_item));
  foreign_schema : STRING;
END_ENTITY;
```

```
ENTITY explicit_item
  ABSTRACT SUPERTYPE OF (ONEOF (used_item, referenced_item))
  SUBTYPE OF (interfaced_item);
  local_definition : named_type;
  original_name : OPTIONAL STRING;
END_ENTITY;
```

¹ В нотации Вирта импорт глобальных правил (RULE) не предусматривается, следовательно, здесь имеется несоответствие

```

ENTITY implicit_item
  SUBTYPE OF ( interfaced_item );
  local_definition : implicit_interface_type;
END_ENTITY;

```

```

ENTITY used_item
  SUBTYPE OF ( explicit_item );
END_ENTITY;

```

```

ENTITY referenced_item
  SUBTYPE OF ( explicit_item );
END_ENTITY;

```

```

TYPE implicit_interface_type = SELECT
  (named_type,
   global_rule);
END_TYPE;

```

При разработке EXPRESS-схем, используемых в Прикладных протоколах STEP (подробнее о процессе разработки Прикладного протокола см. в разделе «Методика разработки Прикладного протокола») заимствуются определения, сделанные в других концептуальных схемах данных (EXPRESS-схемах). При этом для указания заимствования используются конструкции USE и REFERENCE.

В Прикладных протоколах STEP предусмотрено два вида распечатки (listing) EXPRESS-схем: короткая распечатка (short listing), в которой приведены только те определения, которые сделаны в данном Прикладном протоколе, а все заимствованные определения заданы посредством спецификаций импорта и полная распечатка (full listing), в которую скопированы все импортированные определения.

Ссылки между отдельными листами схем EXPRESS-G

Элементы нотации EXPRESS-G уровня сущностей были рассмотрены выше, в соответствующих разделах. Остался лишь один нерассмотренный аспект.

Поскольку построение диаграмм в EXPRESS -G имеет смысл прежде всего для больших схем, вся диаграмма на одном листе уместиться не может, и ее приходится разбивать на несколько листов (Рис. 31). Для ссылок между листами используются овалы рамки.

Овальная рамка ставится на линиях, представляющих отношения «подтип-супертип» (жирные линии), отношения «атрибут-область определения» (простые линии) и «определенный в схеме тип-используемые типы» (простые линии) в тех случаях, когда линии соединяют объекты, расположенные на разных листах. Отношения в EXPRESS-G направленные, и поэтому имеет значение, исходит ли линия из данного листа схемы или входит в него. Точка входа линии (линий) имеет идентификатор, состоящий из номера данного листа и номера точки входа. Номера точек входа могут быть уникальными в пределах одного листа или уникальными в пределах всего комплекта листов

(последнее принято в комплектах EXPRESS-G диаграмм, входящих в Протоколы применения STEP). После идентификатора точки входа в скобках перечислены номер (номера) листов, с которых приходят линии. На исходящей линии пишется только идентификатор соответствующей точки входа.

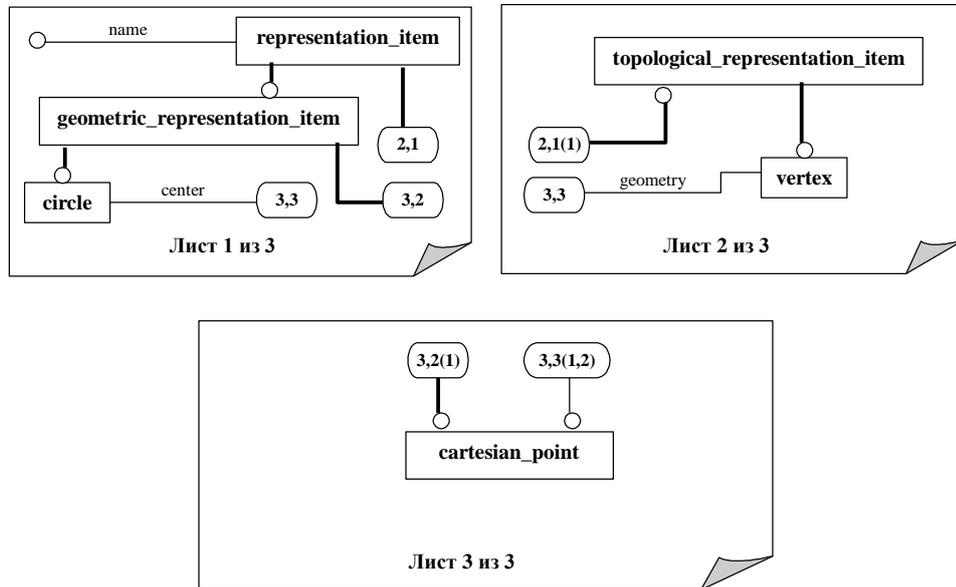


Рис. 31. EXPRESS-G диаграмма, разделенная на три листа

Заголовочная секция обменного файла

Заголовочная секция обменного файла располагается в начале файла и занимает несколько строк (т.е. зачастую – доли процента общего объема обменного файла). Заголовочная секция содержит следующие данные:

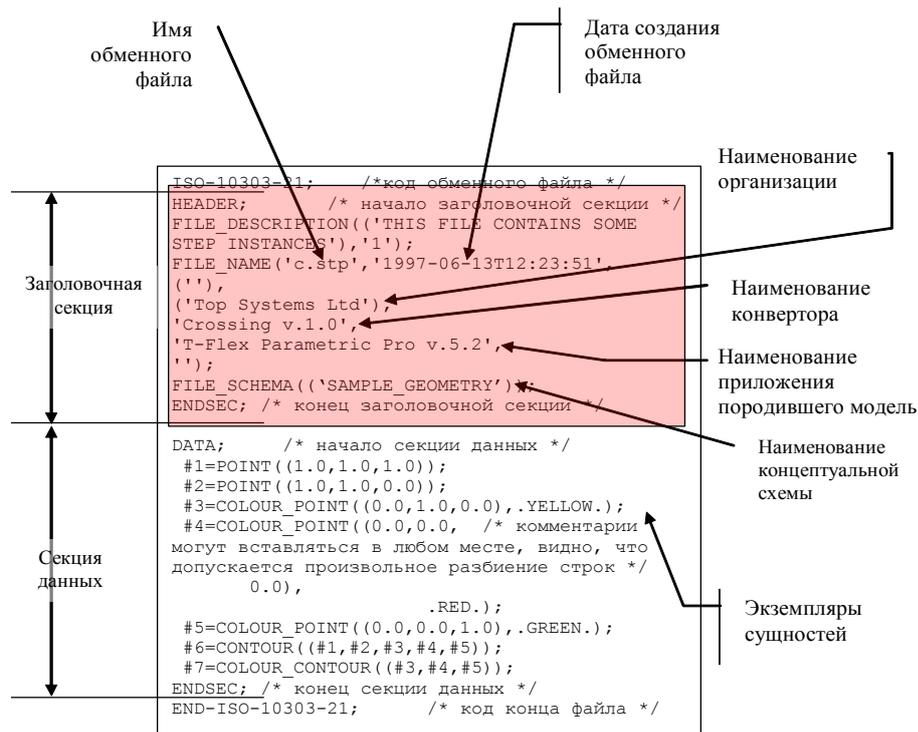


Рис. 32. Заголовочная секция обменного файла

	Тип дан-ных	Содержание данного поля
FILE_DESCRIPTION		
description	Список строк	Набор символьных строк, содержащих произвольную информацию об обменном файле.
implementation_level	Строка	Уровень реализации обменного файла. Может принимать значения 1 или 1,2. На это поле следует обращать особое внимание , т.к. конвертор STEP системы Unigraphics при работе учитывает значение этого поля.
FILE_NAME		
name	Строка	Имя файла (записывается имя файла, которое было ему дано при создании. В некоторых случаях может быть записано полное имя, т.е. путь и имя)
time_stamp	Строка	Дата и время создания обменного файла
author	Список строк	Лица, ответственные за создание обменного файла
organization	Список строк	Организации, ответственные за создание обменного файла
preprocessor_version	Строка	Наименование конвертора,

	Тип данных	Содержание данного поля
		вавшего обменный файл. Как правило, конвертор представляет собой или отдельный модуль приложения или даже отдельное приложение.
originating_system	Строка	Наименование приложения, сгенерировавшего данные, содержащиеся в обменном файле.
authorization	Строка	Лицо, утвердившее содержимое обменного файла
FILE_SCHEMA		
schema_identifiers	Список строк	<p>Наименование EXPRESS-схемы, которой соответствуют записи, содержащиеся в обменном файле.</p> <p>Хотя данное поле и допускает наличие нескольких наименований схем, согласно стандарту, в поле должно содержаться наименование одной схемы.</p> <p>Возможные отступления от данного правила:</p> <ol style="list-style-type: none"> 1. В некоторых реализациях STEP, например, в реализации, используемой на фирме Boeing, используется, помимо стандартной EXPRESS-схемы, собственная EXPRESS-схема внутреннего использования. В таких обменных файлах фирмы Boeing в данном поле приводится две схемы. 2. Планируется принятие новой версии стандарта ISO 10303-21, в которой в одном обменном файле будет содержаться несколько секций данных. В заголовочной секции таких обменных файлов содержится список всех схем, используемых в файле. Хотя новая версия стандарта еще не принята, уже существуют программные продукты, работающие с обменными файлами

Обычно перед загрузкой обменного файла содержимое заголовочной секции выдается в виде специального окошка (таблички):

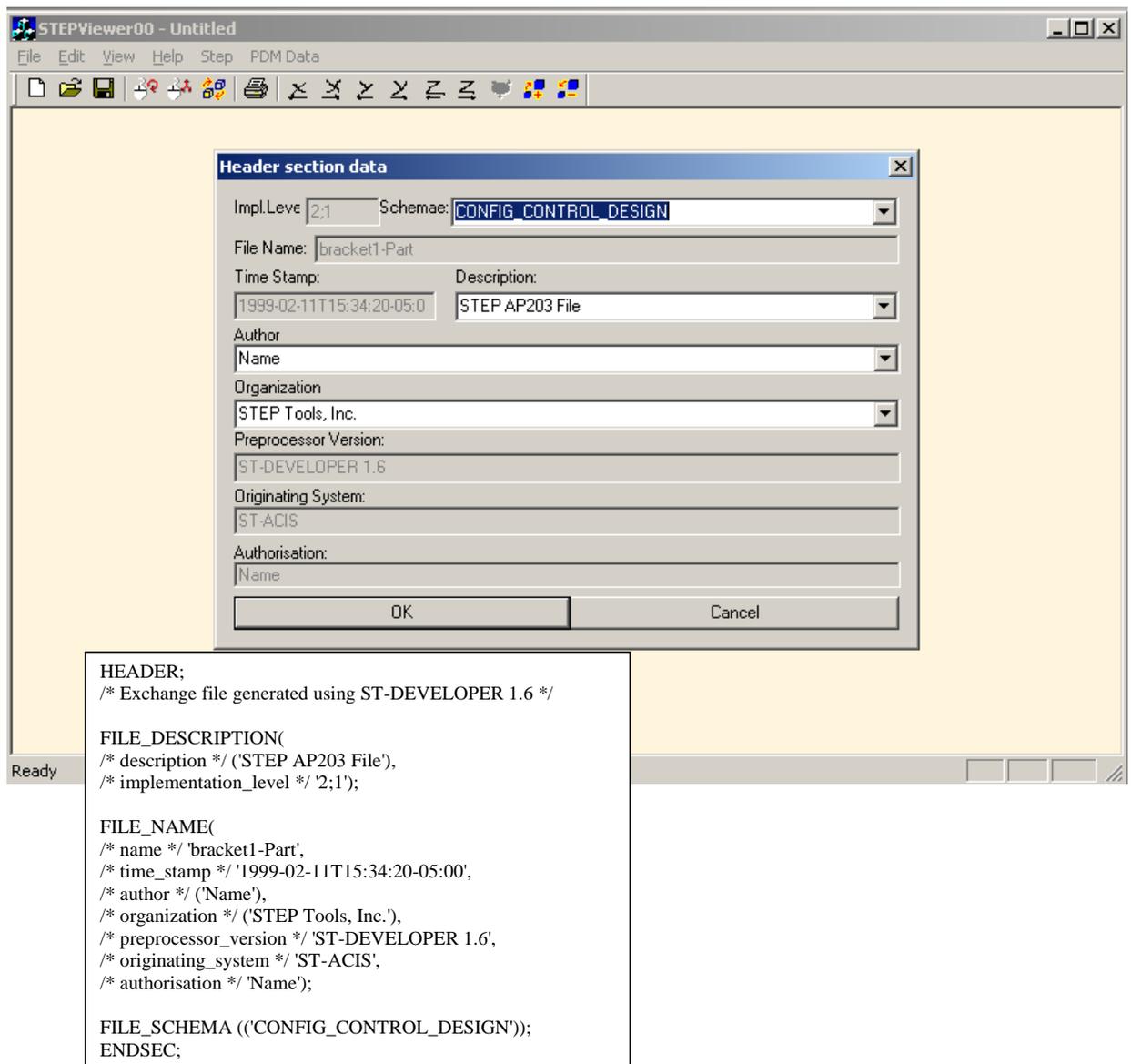


Рис. 33. Состав заголовочной секции и его интерпретация при загрузке (на примере выюера)

Состав заголовочной секции обменного файла представлен также и в форме EXPRESS-схемы (см. ISO 10303-21):

SCHEMA header_section_schema;

ENTITY file_description;

description : LIST [1:?] OF STRING (256) ;

implementation_level : STRING (256) ;

END_ENTITY;

ENTITY file_name;

name : STRING (256) ;

time_stamp : STRING (256) ;

author : LIST [1 : ?] OF STRING (256) ;

organization : LIST[1 : ?] OF STRING (256) ;

preprocessor_version : STRING (256) ;

originating_system : STRING (256) ;

authorisation : STRING (256) ;

END_ENTITY;

ENTITY file_schema;

schema_identifiers : LIST [1:?] OF schema_name;

END_ENTITY;

TYPE schema_name = STRING(256);

END_TYPE;

END_SCHEMA;

Дополнительные определения синтаксиса обменного файла, не вошедшие в описание.

ALPHABET = REVERSE_SOLIDUS "P" UPPER REVERSE_SOLIDUS.

ARBITRARY = REVERSE_SOLIDUS "X" REVERSE_SOLIDUS
HEX_ONE.

CONTROL_DIRECTIVE = PAGE | ALPHABET | EXTENDED2 | EXTENDED4 | ARBITRARY.

END_EXTENDED = REVERSE_SOLIDUS "X0" REVERSE_SOLIDUS.

EXPORT_LIST = "/" ENTITY_INSTANCE_NAME { ","
ENTITY_INSTANCE_NAME } "/".

EXTENDED2 = REVERSE_SOLIDUS "X2" REVERSE_SOLIDUS
HEX_TWO { HEX_TWO } END_EXTENDED.

EXTENDED4 = REVERSE_SOLIDUS "X4" REVERSE_SOLIDUS
HEX_FOUR { HEX_FOUR } END_EXTENDED.

HEADER_ENTITY = KEYWORD "(" [PARAMETER LIST] ")" ";".

HEADER_ENTITY_LIST = HEADER_ENTITY { HEADER_ENTITY }.

HEADER_SECTION = "HEADER;" HEADER_ENTITY HEADER_ENTITY
HEADER_ENTITY [HEADER_ENTITY_LIST] "ENDSEC;".

HEX = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8"
| "9" | "A" | "B" | "C" | "D" | "E" | "F".

HEX_FOUR = HEX_TWO HEX_TWO.

HEX_ONE = HEX HEX.

HEX_TWO = HEX_ONE HEX_ONE.

NON_Q_CHAR = SPECIAL | DIGIT | SPACE | LOWER |
UPPER.

PAGE = REVERSE_SOLIDUS "S" REVERSE_SOLIDUS CHARACTER.

PARAMETER_LIST = PARAMETER { "," PARAMETER }.

SCOPE = "&SCOPE" ENTITY_INSTANCE_LIST "ENDSCOPE" [EXPORT_LIST].

SIGN = "+" | "-".

Контрольные вопросы

10. Какие типы агрегатов существуют в языке EXPRESS и в чем заключаются их особенности.
11. Какие разновидности атрибутов (явные, производные, инверсные) записываются в обменный файл.

5. Стандартный интерфейс доступа к данным SDAI (Standard data access interface)

Второй метод реализации STEP - это протокол доступа к базам данным, содержимое которых определено средствами EXPRESS. Назначение такого протокола заключается в снижении стоимости эксплуатации интегрированной базы данных изделия за счет обеспечения совместимости сложных прикладных программных продуктов с такой базой данных.

SDAI описана в нескольких томах ISO 10303 STEP:

в **томе 22** даны логическая структура данных SDAI и спецификация функций обращения к базе данных в общем виде (без привязки к конкретному языку программирования);

в **томе 23** дана спецификация функций обращения к базе данных на языке C++;

в **томе 24** дана спецификация функций обращения к базе данных на языке C;

в **томе 25** дана спецификация функций обращения к базе данных на языке FORTRAN. Этот том был отменен. В настоящее время номер присвоен проекту реализации SDAI средствами UML.

в **томе 26** дана спецификация функций обращения к базе данных на языке IDL.

в **томе 27** дана спецификация функций обращения к базе данных на языке JAVA.

в **томе 28** дана спецификация структур данных SDAI на языке XML.

В дополнение к операциям доступа к прикладным данным, приложения, построенные с использованием SDAI, имеют доступ к модели сессии, также определенной средствами EXPRESS. Модель сессии SDAI создается и изменяется в качестве побочного эффекта различных операций SDAI. Модель сессии содержит открытые наборы данных, способы доступа, протоколы сообщений об ошибках и т.д. Некоторые из реализаций SDAI могут обеспечивать словарь, содержащий определения данных. Определения данных сделаны средствами языка EXPRESS. Форма, в которой эти определения существуют, также описываются в SDAI.

Логическая организация данных SDAI

В SDAI дано описание логической организации базы данных. База данных состоит из репозитория REPOSITORY, моделей SDAI MODEL, и экземпляров схем SCHEMA INSTANCE. База данных является объектно-ориентированной, поэтому содержимое ее представлено в виде множества объектов (экземпляров сущностей).

База данных, согласно SDAI, должна строиться по следующим принципам. Находящиеся в репозитории экземпляры сущностей (за исключением некоторых их разновидностей - об этом см. ниже) объединены в ассоциации. Такая ассоциация называется SDAI-модель. Один экземпляр сущности может входить только в одну SDAI-модель. SDAI-модель не может включать в себя другие SDAI-модели. SDAI-модель полностью находится в одном репозитории.

Одним из атрибутов SDAI-модели является ссылка на определение собственной "native" схемы EXPRESS. SDAI-модель может иметь только одну собственную схему. Каждый экземпляр сущности из числа входящих в данную SDAI-модель должен быть определен или в этой собственной схеме или в одной из схем, которая определена как интероперабельная с собственной схемой SDAI-модели.

Несколько SDAI-моделей, относящихся к одной схеме и к интероперабельным с ней схемам, могут быть объединены в экземпляр схемы - SCHEMA INSTANCE. Одна SDAI-модель может одновременно входить в несколько экземпляров схем. Экземпляр схемы является полем идентификации (информационной базой). Ссылки из экземпляров сущностей, входящих в экземпляр схемы на экземпляры сущностей, не входящие в экземпляр схемы, не допускаются. Возможность вхождения SDAI-модели одновременно в несколько экземпляров схем позволяет использовать общие информационные ресурсы в разделенном режиме.

Экземпляр схемы (SCHEMA INSTANCE) может включать SDAI-модели, существующие в нескольких репозиториях.

Информации о количестве и составе SDAI-моделей также хранится в виде экземпляров сущностей. Отличие экземпляров сущности, описывающих SDAI-модели от всех прочих экземпляров сущностей лишь в том, что сами экземпляры сущностей, описывающие SDAI-модель ни в какую SDAI-модель в обязательном порядке не входят. В то же время запрета на вхождение этих экземпляров в какую-нибудь SDAI-модель нет. Сама SDAI-модель может быть структурирована с помощью механизма SCOPE (областей видимости). Области видимости образуют иерархическую структуру неограниченной глубины. Определены операции над областями видимости:

формирование областей видимости (создание новой области видимости, добавление в область видимости экземпляра сущности, удаление экземпляра сущности из области видимости, управлению видимостью экземпляров сущностей из-за пределов области видимости);

операции над областями видимости в целом - копирование, удаление и проверка целостности структуры данных.

Классы функций SDAI

Управление средой исполнения и сессией (Environment and session operations) – 15 функций

Операции над экземплярами схем, репозиториями и SDAI-моделями (Schema instance, repository and SDAI-model operations) – 20 функций

Операции над областями видимости экземпляров сущностей (Scope operations) – 10 функций

Операции с типами данных (Type operations) – 4 функции

Операции над экземплярами сущностей (Entity instance operations) – 23 функции

Операции над агрегатными атрибутами экземпляров сущностей (Entity instance aggregate operations) – 25 функций

Операции создания и модификации данных разрешены только над экземплярами прикладных сущностей

Раннее и позднее связывания

Конкретные реализации SDAI могут быть разделены на две большие группы: реализация SDAI раннего связывания (EARLY BINDING) и реализация SDAI позднего связывания (LATE BINDING).

Реализация SDAI с ранним связыванием не содержит словаря данных. Реализация SDAI с поздним связыванием содержит словарь данных, доступный для приложения во время работы.

Раннее связывание

Системы с ранним связыванием предполагают создание поддерживаемых конкретным языком программирования структур данных. Структуры данных отображают определения данных, сделанные средствами EXPRESS. Например, раннее связывание для Прикладного Протокола AP203, сделанное с использованием SDAI C++, использует для каждого определения, сделанного на EXPRESS, свой класс C++. Преимущество такого подхода состоит в возможности организации проверки типов.

Классы C++ для раннего связывания обычно генерируются компилятором EXPRESS (в специальном режиме генерации классов C++). Определения сущностей ENTITY отображаются в классы, определения типов TYPE отображаются или в классы, или в определения типов (TYPEDEF). Структура наследования, описанная средствами EXPRESS, отображается в иерархию классов C++. Каждый из классов имеет методы доступа и модификации для хранящихся атрибутов, и конструкторы для инициализации новых экземпляров. Ниже приведен простой пример определения сущностей, сделанного средствами EXPRESS, и соответствующие этому определению классы C++.

```
ENTITY Point;
```

```
  x : REAL;
```

```
  y : REAL;
```

```
END_ENTITY;
```

Прикладная программная система может использовать методы классов C++ для создания экземпляров сущностей, присвоения значений атрибутам и записи экземпляров сущностей в репозиторий сущность создается с применением специальной версии оператора new.

```
/* Create a point using the default constructor  
* and use the update methods to set its values. */  
SdaiModelH mod;  
PointH point1 = SdaiCreate(mod, Point);  
Point1->x (1.0);  
Point1->y (0.0);
```

Позднее связывание

SDAI позднего связывания, такое, как SDAI C, использует для доступа к данным словарь данных EXPRESS. При позднем связывании не используются генерируемые структуры. Для всех определений модели EXPRESS используется одна единственная структура данных.

Для поиска значений данных используется словарь данных SDAI. Прикладная программная система использует для присвоения и запроса значений использует ограниченный набор простых универсальных функций, а не специализированные функции для каждого из значений. Ниже приведен программный код прикладной программной системы, построенной на SDAI позднего связывания:

```
/* create new instances */  
SdaiApplInstance point1;  
point1 = sdaiCreateInstanceBN(myModel, "point");  
sdaiPutAttrBN(point1, "X", sdaiREAL, 1.0);  
sdaiPutAttrBN(point1, "Y", sdaiREAL, 0.0);
```

В общем, позднее связывание полезно в тех случаях, когда требуется работа прикладного программного обеспечения с большим количеством определений данных, относящихся к нескольким схемам EXPRESS, в условиях, когда нет необходимости в строгом контроле типов. Определение типов данных, сделанное средствами EXPRESS, может быть изменено без отражения влияния этих изменений на прикладное программное обеспечение.

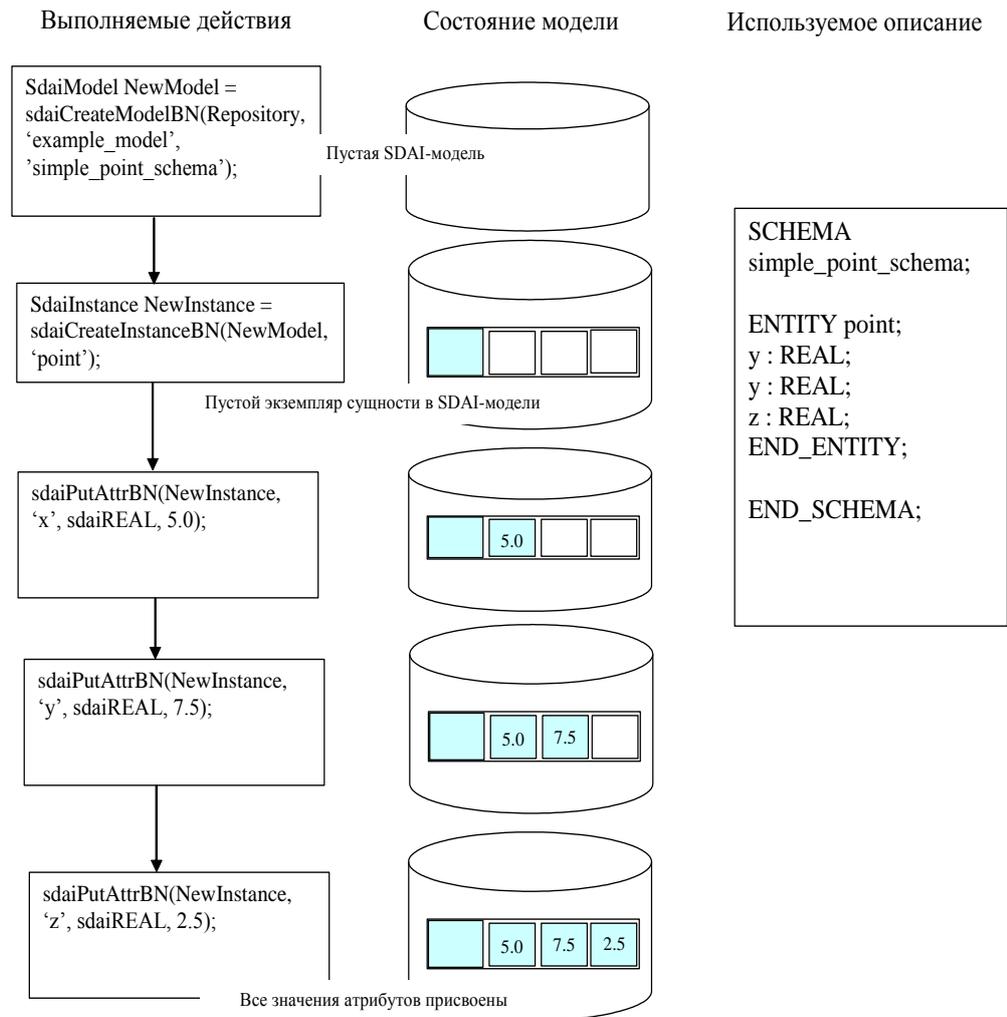


Рис. 34. Организация доступа к данным при позднем связывании.

Следует учесть, что стандартный, не очень большой, Прикладной Протокол STEP может включать сотни определений типов данных (например, Протокол AP203 содержит определения 262 типов сущностей, и это далеко не самый большой из Прикладных Протоколов), Протокол AP214 содержит немногим более 900 сущностей, а AP210 – более 1000 сущностей.. При раннем связывании необходимо каждое из определений преобразовать в классы C++, откомпилировать реализации этих классов и собрать их в библиотеки.

Описанная на языке EXPRESS концептуальная модель данных STEP отличается большим многообразием типов сущностей и свойств сущностей, сложными отношениями наследования и широким использованием агрегатных типов данных, в том числе и с несколькими уровнями вложенности. Такая модель плохо совместима с реляционной моделью данных, и первые попытки организовать хранение данных STEP с использованием реляционных моделей не дали положительных результатов.

Принцип позднего связывания может быть реализован с использованием технологии динамических сетей. Разработка SDAI, первые варианты которого появились в 1992 году, является важным этапом в процессе перехода от реляционных моделей данных к объектно-ориентированным моделям.

Контрольные вопросы

- 12.С помощью каких средств программирования может быть построена реализация библиотеки SDAI.
- 13.Чем отличается принцип раннего связывания от принципа позднего связывания. Преимущества и недостатки каждого из принципов.

6. Методика разработки Прикладных протоколов STEP

Понятие интероперабельности играет в STEP ключевую роль. Поскольку STEP - это стандарт обмена данными, то под интероперабельностью в STEP понимается интероперабельность по данным, или совместимость данных в пределах пространства STEP. Если учесть, что основная единица STEP – это Прикладной Протокол, то в STEP основной аспект интероперабельности – это интероперабельность Прикладных Протоколов.

В стандарте ISO 10303 STEP предусмотрено использование большого числа Протоколов Применения (AP, Application Protocol), содержащих стандартизованные концептуальные схемы прикладных предметных областей. Протоколы Применения могут частично между собой перекрываться. В различных Протоколах Применения могут использоваться одинаковые или сходные между собой понятия (сущности).

Например, понятия изделия и идентификации изделия, исполнителей и организаций, геометрических объектов присутствуют во всех Протоколах. Понятия технологического процесса и структуры изделия имеются в большей части Протоколов, и т.д. Усилия разработчиков стандарта направлены на то, чтобы общие для нескольких Протоколов понятия описывались одинаково.

Следовательно, существует необходимость обеспечить однородность концептуальных моделей предметных областей, таким образом, чтобы в смежных предметных областях схожие или идентичные понятия определялись одинаковыми или родственными между собой сущностями.

При условии того, что это обеспечено, появляется возможность, создав с помощью одного приложения модель, описывающую изделие с точки зрения одной предметной области (одного AP), отделить ту часть модели, которая содержит данные, одновременно соответствующие и другой предметной области (другому AP) и передать эти данные для дальнейшей обработки приложению, работающему в этой другой предметной области. Такая схема передачи данных показана на .

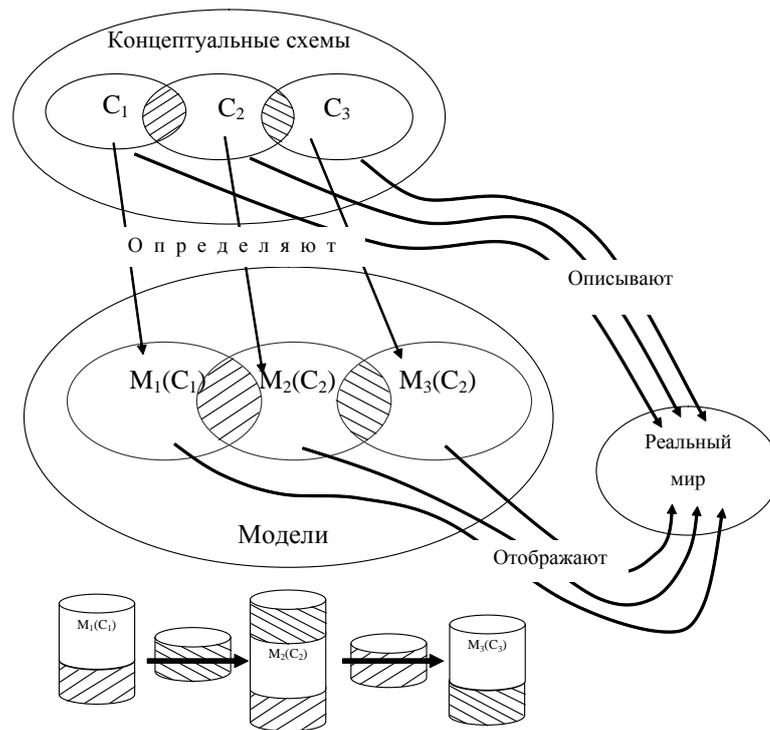


Рис. 35. Применение механизма интероперабельности при передаче данных между смежными предметными областями

Такая однородность поддерживается следующим образом: в стандарте STEP предусмотрено использование «ресурсов», т.е. заранее заготовленных определений самых общих и самых распространенных понятий. Ресурсы STEP определены (описаны) с помощью языка EXPRESS. Частичное перекрытие концептуальных схем в сочетании с тем, что существующие методики разработки Прикладных Протоколов способствуют совпадению перекрывающихся частей концептуальных схем, приводят к возможности передачи данных между различными предметными областями и/или различными этапами Жизненного Цикла Изделия.

Предусмотренный в языке EXPRESS механизм наследования определений сущностей позволяет получать из определений сущностей, включенных в ресурсы STEP, определения сущностей, необходимые для построения концептуальной модели конкретной предметной области.

Приведем пример цепочки передачи данных:

AP203 – структура изделия, конфигурация изделия, геометрические модели компонентов изделия;

AP224 – геометрическая модель одной детали, конструктивно-технологические элементы этой детали, технологический процесс изготовления детали;

AP213 – технологический процесс изготовления детали, используемый при изготовлении инструмент;

AP238 – программы для оборудования с ЧПУ;

AP209 – геометрическая модель изделия, расчетные модели изделия.

Особое место занимает серия согласованных между собой судостроительных Протоколов AP215, AP216, AP217/227, AP218, AP226 и AP234. Эти Протоколы будут рассмотрены далее.

Обеспечение интероперабельности в STEP можно рассматривать в следующих аспектах:

- Методика разработки Прикладных Протоколов STEP, основанная на применении информационных ресурсов;
- Состав информационных ресурсов;
- Существующие в SDAI механизмы обеспечения интероперабельности, т.е. рассмотренный выше механизм экземпляров схем (schema instance), позволяющий компоновать SDAI-модели, относящиеся к различным концептуальным схемам.

Опыт разработки стандартов показал, что невозможно и неэффективно охватить весь круг вопросов одним стандартом.

Поэтому при создании STEP был принят следующий подход:

всю область действия стандарта разбить на ряд пересекающихся “Прикладных областей”, каждая из которых представляет часть жизненного цикла для некоторых видов изделий;

разрабатывать стандартные конструкции описания данных для каждой предметной области.

Для решения этих задач был разработан “механизм”, основанный на понятиях OSI и опыте разработки стандарта IGES: “прикладной протокол” (AP - application protocol).

В июне 1989 TC184/SC4/WG1 ISO приняла следующую стратегию разработки прикладного протокола (AP):

Практическая реализация STEP должна обеспечивать обмен данными в рамках независимых прикладных протоколов (AP) и не требует использования всех конструкций STEP.

Каждый протокол, использующий STEP должен предоставлять возможность описывать всю информацию, необходимую для реализации требований выбранной прикладной области.

Метод разработки прикладного протокола должен иметь механизмы, гарантирующие, что общая информация используется несколькими прикладными протоколами совместно. Протоколы, реализующие общие требования к информации, должны использовать одни и те же базовые конструкции STEP для выполнения этих требований.

Основные требования, предъявляемые к Прикладному Протоколу и методика разработки Прикладного Протокола изложены в документе “Guidelines for the development and approval of STEP Application Protocol” (Руководство по разработке и утверждению Прикладного Протокола STEP).

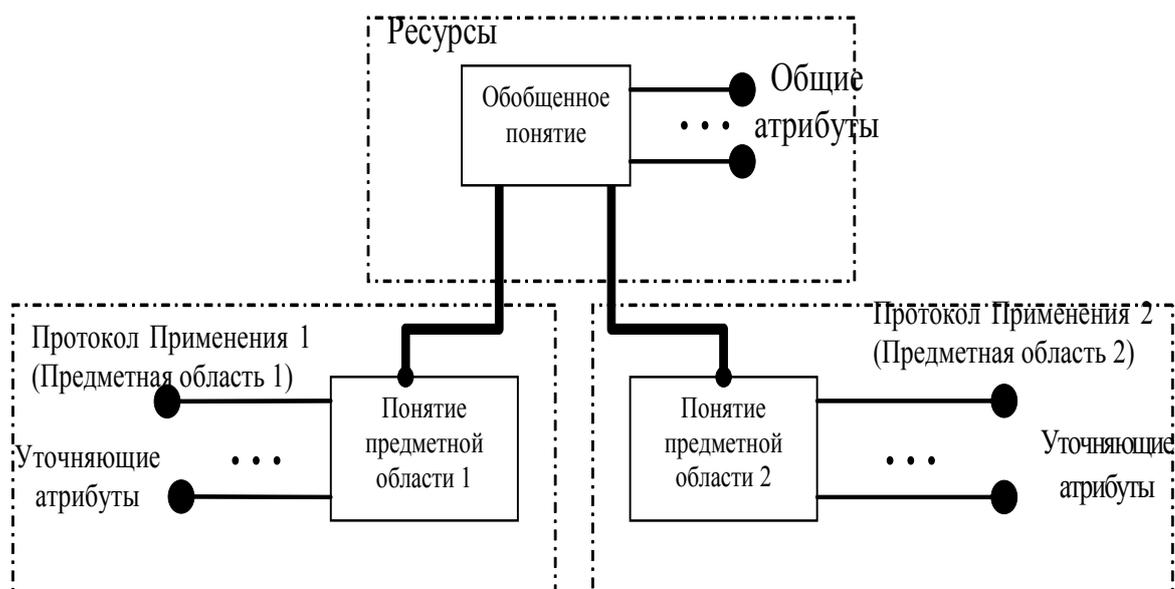


Рис. 36. Принцип применения информационных ресурсов

Комитетом TC184/SC4 были сформулированы следующие принципы, которыми необходимо руководствоваться при разработке каждого прикладного протокола:

Базовые ресурсы STEP могут быть доступны только через конкретные прикладные протоколы;

Базовые конструкции STEP могут интерпретироваться в прикладном протоколе следующим образом:

- добавление ограничений на значения атрибутов;
- изменение необязательности атрибутов (OPTIONAL);
- добавлений правил (WHERE, RULE, и др.), управляющих значениями атрибутов и поведением объектов;
- создание подтипов (SUBTYPE) объектов, при этом подтип может иметь любые дополнительные атрибуты.

Модели прикладного протокола. Каждый прикладной протокол должен включать в себя три типа моделей:

- Функциональную модель предметной области (AAM);
- Концептуальную информационную модель предметной области (ARM);
- Модель интерпретации ресурсов STEP (AIM).

Обозначение	Наименование	Содержание	Средства представления
AAM	Application Activity Model Функциональная модель предметной области	Структура функций, которые создают и используют информацию о производстве изделия в контексте предметной области, охватываемой данным протоколом. Функциональная модель строится в предположении, что в данной предметной области уже достигнут требуемый уровень компьютерной автоматизации	IDEF0

Обозначение	Наименование	Содержание	Средства представления
ARM	Application Reference Model Справочная модель предметной области	Концептуальная информационная модель (концептуальная схема данных), описывающая структуру информационных понятий предметной области и их связи, обеспечивающие реализацию ААМ. ARM строится полностью в терминах предметной области.	EXPRESS, ранее использовались: IDEF1X, диаграммы Чена, NIAM
AIM	Application Interpreted Model Интерпретированная модель предметной области	Концептуальная схема данных, полностью идентичная ARM, но выраженная с помощью сущностей, имеющих в информационных ресурсах STEP. Процесс выражения содержимого ARM, заключающийся в подборе и адаптации содержащихся в информационных ресурсах сущностей называется интерпретацией.	EXPRESS

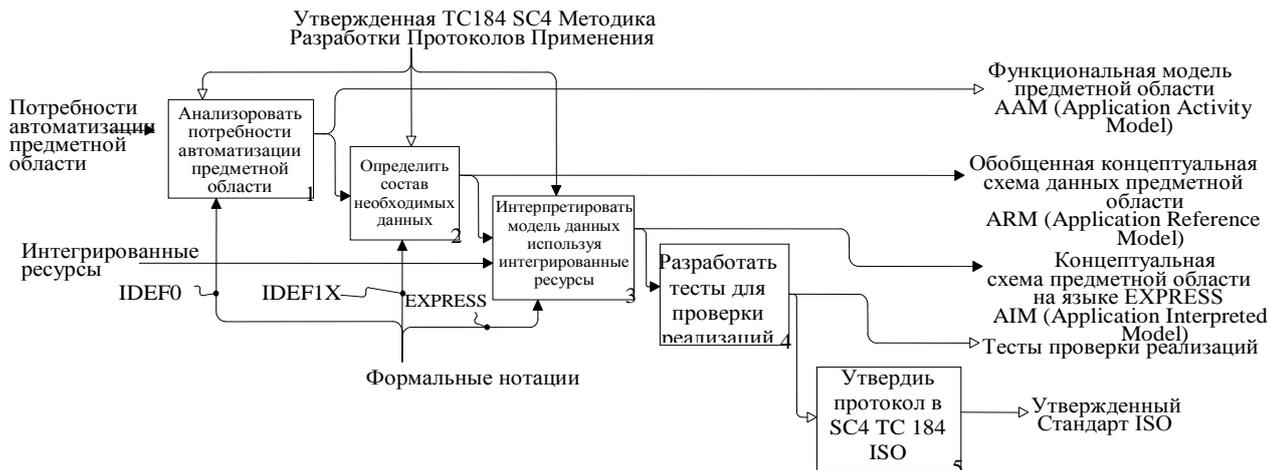


Рис. 37. Процесс разработки Прикладного протокола

Причем ААМ и АРМ управляют разработкой АИМ. ААМ и АРМ являются информационными приложениями прикладного протокола, а АИМ, представленный на языке EXPRESS, является нормативной частью прикладного протокола.

Конструкции АИМ и его прикладная область должны иметь взаимное соответствие;

Общие информационные требования различных прикладных протоколов должны быть реализованы одними конструкциями STEP. Общая часть нескольких протоколов объединяется в специальную конструкцию – АИС (тома 500-й серии).

Прикладному протоколу соответствует также том 300-й серии, содержащий набор стандартных тестов для проверки программного обеспечения на соответствие прикладному протоколу.

Состав Прикладного Протокола

В Руководстве приведен состав Прикладного Протокола:

- Предисловие
- Введение
- Определение области действия
- Нормативные ссылки
- Используемые определения
- Информационные требования
- Определение конструкций
- Прикладная модель
- Таблицы отображения
- Концептуальная модель
- Классы соответствия

Состав приложений к Прикладному протоколу также регламентирован.

Индекс	Обязат.	Содержание
A	Да	Полный текст Прикладного Протокола на языке EXPRESS
B	Да	Таблица коротких имен объектов и типов
C	Да	Проформа PICS
D	Да	Специфические требования к реализации
E	Да	Регистрационные данные Прикладного Протокола
F	Да	Функциональная модель (AAM), включающая определения (гlossарий) и схемы IDEF0
G	Да	Концептуальная информационная модель (ARM), включающая схемы EXPRESS-G (в старых вариантах протоколов - IDEF1x)
H	Да	Схемы EXPRESS-G
J	Нет	Текст на машинном носителе
K	Нет	Руководство по использованию, включая пример обменного файла
L	Нет	Библиография
M	Нет	Описание используемых ресурсов STEP

Процесс разработки, сертификации и утверждения прикладного протокола как стандарта ISO.

Авторы методики разработки прикладных протоколов предположили, что прикладной протокол будет находиться в процессе развития, который определяется все более полным изучением требований предметной области и развитием самой предметной области. Для разработки прикладного протокола формируется коллектив в рамках SC4 (PMAG), либо этим занимается инициативная группа, которая представляет свою работу на утверждение SC4.

Процесс разработки прикладного протокола включает следующие этапы:

Разработка концепции прикладного протокола и утверждение проекта прикладного протокола. На этом этапе проект прикладного протокола включается в проект STEP и прикладному протоколу присваивается порядковый номер. При утверждении представленного на рассмотрение проекта прикладного протокола PMAG SC4 использует рекомендации рабочих групп: WG2, WG3, WG4.

Группа проекта анализирует прикладную область и формирует требования, предъявляемые предметной областью к информации:

- тип изделий;
- этапы жизненного цикла;
- состав функций, необходимых для реализаций жизненного цикла изделий;
- документы, используемые в данной предметной области для описания информации об изделии.

На этом этапе определяются прикладные протоколы, связанные с данной предметной областью. Результатом этого этапа является разработка ААМ.

Эксперты STEP и специалисты предметной области определяют соответствие требований предметной области и основных конструкций и положений STEP.

Группа проекта прикладного протокола представляет план разработки прикладного протокола.

Группа проекта, используя ААМ, концепцию прикладного протокола и требования предметной области, формирует ARM и частные модели понятий предметной области (UOF).

Группа проекта представляет ARM, ААМ и UOF на рассмотрение WG4 для анализа и оценки возможности разработать AIM с учетом интеграции с прикладными протоколами, относящимися к смежным предметным областям. Результатом этого этапа является утверждение ARM прикладного протокола. Обычно результат имеет статус WD (Рабочий проект - Working Draft).

Группа проекта разрабатывает совместно с WG4, выбирает интегрированные ресурсы STEP и их интерпретацию для реализации требований предметной области и ARM. При этом используются конструкции из библиотек AIC.

Группа проекта с помощью WG4 в соответствии с проектом прикладного протокола разрабатывает AIM и таблицу соответствия ARM - AIM. Обычно по завершении этого этапа выпускается проект прикладного протокола, имеющий статус CD (Проект Комитета - Committee Draft).

Группа проекта компилирует AIM и проводит проверку соответствия интегрированным ресурсам STEP.

Группа проекта разрабатывает правила проверки соответствия данных и тестовые примеры для проверки соответствия программного обеспечения прикладному протоколу.

Группа проекта представляет все компоненты прикладного протокола WG4 и WG6 для утверждения в качестве стандарта ISO.

Оформление всей документации проекта. Оформление и выпуск документации ISO со статусом IS.

Этап разработки концепции прикладного протокола и информационных требований предметной области включает анализ предметной области, формирование словаря понятий (терминов) предметной области.

Естественно, этот этап начинается с изучения типа изделий и той части жизненного цикла изделий, для которой разрабатывается данный прикладной протокол.

Анализ типовых сценариев реализации служит основой для разработки иерархической функциональной модели в стандарте IDEF0.

На основе анализа предметной области и ААМ должен быть составлен словарь понятий предметной области - глоссарий.

Так как информационные связи (входы, выходы, управление, механизм) в дальнейшем станут основой для разработки ARM и AIM, для каждой из них необходимо определить, для каждой из связей необходимо определить находится ли она в контексте данного протокола.

Совокупность функций и информационных связей и составляет словарь понятий предметной области.

Здесь существует несколько принципиальных трудностей которые должны быть решены при создании эффективного протокола.

Первая трудность состоит в том, что интеграция между протоколами требует однообразия обозначения одних и тех же понятий в различных контекстах. Наиболее естественный путь преодоления этой трудности - создание глобального словаря-классификатора, подобного классификатору ЕСКД, ОКП или классификатору технологических операций.

Вторая трудность состоит в том, что стандарт предназначен для использования в будущем времени, поэтому целью этого этапа является построение некоторого “будущего” процесса основываясь на существующем производственном опыте и анализе тенденций развития предметной области.

Разрабатываемый на этом этапе проект прикладного протокола, представляемый в TC184/SC4 должен включать:

Заголовок прикладного протокола, который ясно определяет область прикладного протокола и его функциональные возможности (Например, протокол 213 имеет заголовок: “Разработка управляющих программ для оборудования с ЧПУ для машиностроительных деталей”).

Дата подачи.

Организации (внутри комитета TC184 или вне его), поддерживающие проект.

Определение предметной области прикладного протокола.

ААМ.

Доказательство заинтересованности промышленности в данном прикладном протоколе.

Представленный официальными рецензентами анализ эффекта от внедрения прикладного протокола.

Возможные связи с другими прикладными протоколами.

Предполагаемое использование интегрированных ресурсов STEP.

Возможности разработчиков протокола.

План разработки прикладного протокола, включающие сроки выпуска ARM, AIM и стандартного варианта.

Таблица отображения показывает каким образом каждый функциональный блок (UoF, Unit of Functionality) и входящие в него информационные прикладные объекты (application object) отображаются в одну или несколько конструкций AIM (Application Interpreted Model, Интерпретированная <в виде представленных средствами языка EXPRESS сущностей> Модель Предметной Области).

Контрольные вопросы

14. Каково назначение методики разработки Прикладных протоколов STEP.
15. Как документируется процесс разработки Прикладных протоколов?

7. Основы представления данных об изделии в ISO 10303 STEP

Назначение стандарта ISO 10303 STEP (ГОСТ Р ИСО 10303) – представление данных об изделии, в первую очередь, для обмена, а также – для хранения, в том числе долгосрочного (общепринятым ориентировочным сроком в настоящее время считается 50 лет – длительность жизненного цикла многих сложных изделий – самолетов, судов, АЭС и т.д.).

Стандарт ISO 10303 STEP предназначен для использования во многих областях профессиональной деятельности, в первую очередь – производственной. На следующей схеме показаны типы изделий и отрасли промышленности, для поддержки которых требуются STEP и стандарты SC4.

На схеме (Рис. 38) показана цепочка поставок в терминах типов производимых изделий и использования там, куда эти изделия включаются. На нижнем уровне цепочки поставок выявлена промышленная область переработки сырья, а на верхнем уровне – поставщики инфраструктур. На верхнем уровне иерархии расположены Поставщики услуг и конечные пользователи.

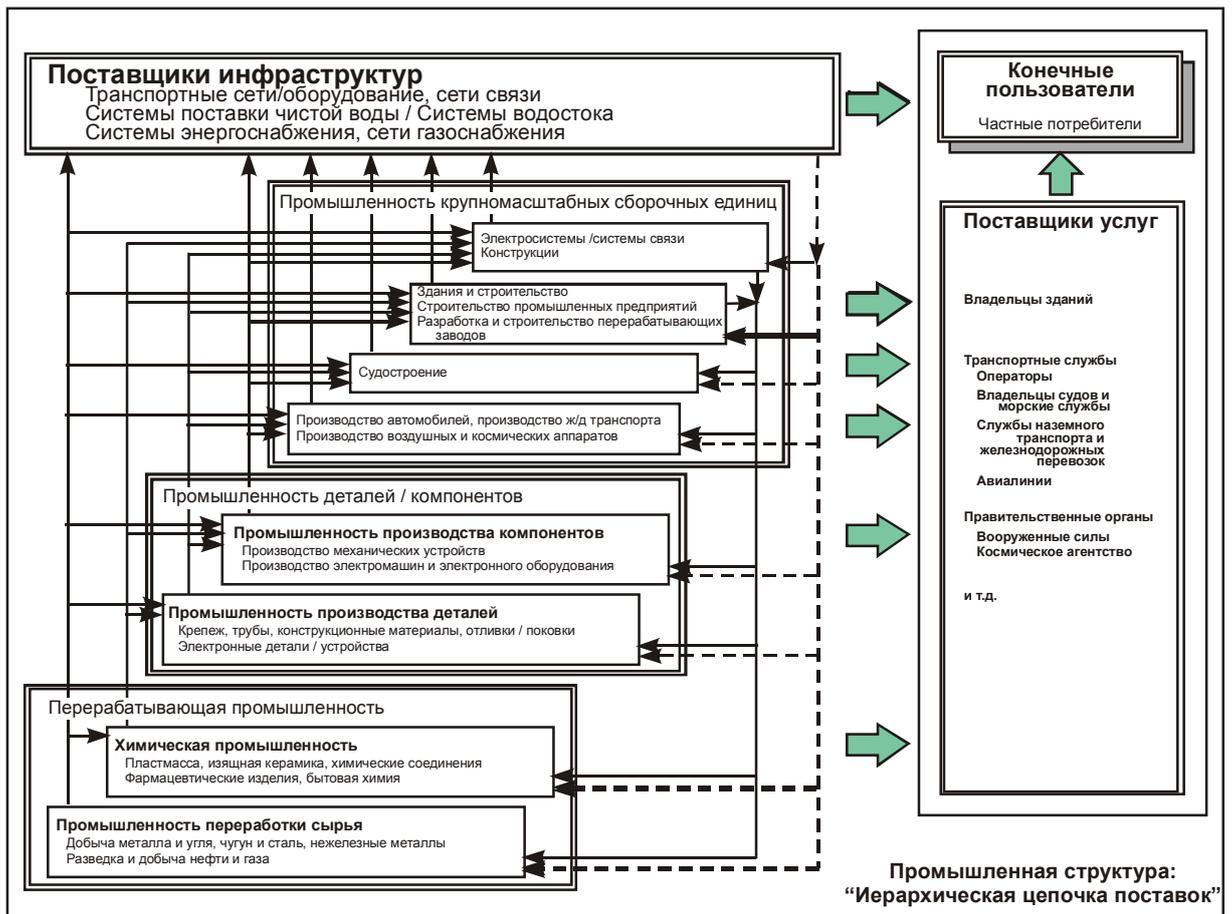


Рис. 38. Цепочка поставок в терминах типов производимых изделий и использования там, куда эти изделия включаются

На следующей схеме (Рис. 39) предоставлен обзор типов деятельностей, которые вовлечены в производство нового изделия и управляют изделием на протяжении его полного жизненного цикла, от колыбели до могилы. Деятели, занесенные в прямоугольники с двойными границами, - это те деятельности, которые входят в область рассмотрения ISO 10303 STEP, а те деятельности, которые занесены в прямоугольники с простыми границами, не входят.



Рис. 39. Обзор типов деятельностей ЖЦ

В STEP принята следующая обобщенная классификация видов данных об изделии:



Рис. 40. Обобщенная классификация видов данных об изделии в STEP

На Рис. 40 более подробно показаны основные определения, связанные с информационным описанием изделия (информационной моделью изделия) в STEP. В STEP предполагается, что изделие может описываться на следующих трех уровнях (см. Рис. 41):

- определение *definition* (41-й и 44-й тома STEP);
- представление *representation* (43-й том STEP и другие тома, содержащие описания отдельных видов представления, например, 42-й том – представление геометрии и топологии);
- визуальное представление *presentation* (46-й том STEP). Этот уровень описания не является для STEP существенным, поскольку основное назначение STEP – построение доступных для компьютерной обработки моделей изделия. Тем не менее, в некоторых из Протоколов STEP предусмотрены сущности, позволяющие описать особенности того, как должно строиться изображение изделия: чертеж и красочная полутонная презентация.

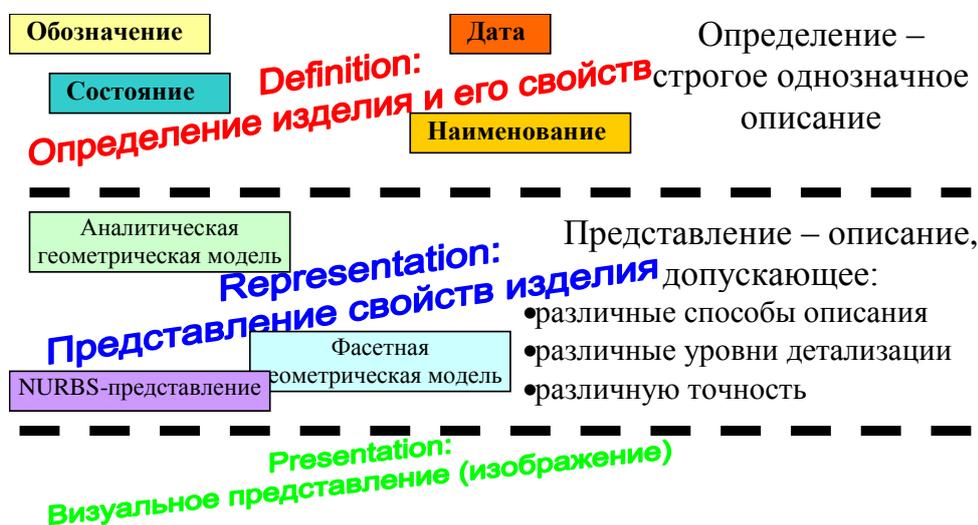


Рис. 41. Основные определения, связанные с описанием изделия в STEP

В STEP данные об изделии организованы следующим образом:

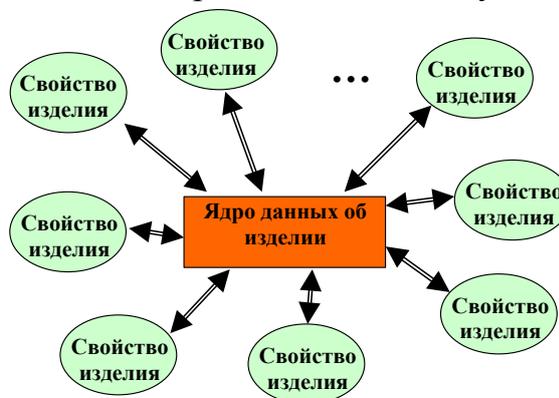


Рис. 42. Организация данных об изделии в STEP

8. Интегрированные информационные ресурсы STEP

При разработке окончательной концептуальной схемы данных STEP, написанной на языке EXPRESS (интерпретированная модель предметной области) большую роль играют информационные ресурсы (см. выше). Сущности концептуальной схемы данных, построенной в терминах предметной области (модель ARM) отображаются в сущности, входящие в информационные ресурсы STEP.

В STEP информационные ресурсы – это заранее заготовленные и утвержденные определения наиболее распространенных объектов и явлений, встречающихся во множестве предметных областей (например, геометрия, изделие и его версии, исполнитель и организация, единицы измерения, процесс согласования и утверждения проектов, присвоение грифов секретности и т.д.).

По степени обобщенности информационные ресурсы STEP делятся на несколько групп – интегрированные информационные ресурсы (самые обобщенные и встречающиеся практически во всех предметных областях – тома 40^й серии), информационные ресурсы предметной области (тома 100^й серии), прикладные интерпретированные конструкции, ориентированные на решение отдельных задач (тома 500^й серии) и прикладные модули (тома 1000^й серии). Рассмотрим некоторые из томов ISO 10303 STEP, содержащих интегрированные информационные ресурсы

Определение изделия в STEP

В информационных ресурсах ISO 10303-41 содержатся сущности, необходимые для задания общего определения изделия.

В стандарте STEP предполагается, что каждое изделие может иметь несколько версий. Каждая версия может иметь несколько определений.

Модель изделия состоит из общей части - каркаса (core model) и ряда отдельных моделей-представлений, описывающих отдельные аспекты изделия.

Общая часть содержит информацию об идентификации изделия, его связях с другими изделиями, о структуре отдельных представлений, о производителях изделия и т.д.

Модели представления описывают отдельные формы и свойства изделия, его использование и производство.

Каждое изделие может иметь несколько “описаний” (product_definition), относящихся к различным этапам жизненного цикла или к разным контекстам описания. Относящиеся к различным этапам жизненного цикла разновидности описаний одного изделия существуют одновременно и связаны с одним объектом - изделием. Это позволяет контролировать насколько свойства изделия соответствуют проекту изделия и требованиям потребителя. Разновидности описания изделия, описывающие изделие на различных этапах его жизненного цикла, могут быть подтипами от описания изделия и соответственно иметь различные имена.

Один элемент представления (`representation_item`) хранится в одном месте, но может быть связано с несколькими разновидностями описания изделия, причем в различных ролях.

Каждая разновидность описания изделия может иметь несколько моделей представления для описания одного и того же свойства изделия (например, для описания формы можно использовать плоский чертеж, геометрическую модель, текст и др.) или описание одного свойства в различных условиях.

Том 41 содержит определения следующих групп ресурсов:

общие ресурсы описания изделия. Эти ресурсы обеспечивают описание независимых от приложений фактов, которые являются общими для всех изделий.

Общие управляющие ресурсы. Эти ресурсы обеспечивают описание данных, которые используются для управления данными об изделии. Две эти группы ресурсов обеспечивают основу для представления в Прикладных Протоколах документации об изделии.

Ресурсы поддержки обеспечивают однородность данных, определяющих идентификацию изделия, единицы измерения, описание исполнителей и организаций.

В общие ресурсы описания изделия входят следующие 4 схемы:

`application_context_schema`. Объекты, определенные в этой схеме, позволяют задать, к какой предметной области относится данный прикладной протокол, какие аспекты изделия отражены в данной модели, и к какому этапу жизненного цикла изделия данная модель относится.

`product_definition_schema` Объекты, определенные в этой схеме, позволяют давать описание общих аспектов изделия. К таким общим аспектам относятся, к примеру, идентификатор и версия изделия, классификация изделия, определения различных взаимоотношений между изделиями, например, отношения часть – целое (отношения входимости). В то же время объектов, позволяющих задать позиционирование компонентов сборки, в этой схеме нет.

Изменения, вносимые в изделие, могут приводить к возникновению новой версии данного изделия или к возникновению нового изделия. Принципы, по которым определяется, приводит ли изменение

- к появлению новой версии
- к появлению нового изделия

стандартом не регламентированы. Эти принципы могут быть разными для разных отраслей и подотраслей промышленности и даже для отдельных предприятий.

В схеме `product_definition_schema` определены два правила, проверяющие отсутствие рекурсий в конструкторском графе и в классификаторе изделий.

`product_property_definition_schema`. Изделие характеризуется своими свойствами. Объекты, определенные в этой схеме, позволяют указывать на свойства изделия и строить связи между свойствами и определениями изделий.

Shape (форма) – это одно из свойств изделия, не зависящее от того, каким образом это свойство представлено. Может существовать несколько представлений одной формы. Например, если проектируется крыло самолета, крыло имеет точную форму, которая получена в результате серии расчетов и экспериментов. Поверхность крыла описана в виде NURBS высокой точности с большим количеством узлов. В то же время для компоновки механизма уборки шасси, расположенного в крыле, такая высокая точность не нужна. Для компоновки достаточно иметь упрощенную плоскогранную модель крыла.

Под формой понимается не только форма изделия или форма фрагмент изделия, но и отношения между формами (взаимное расположение изделий или фрагментов). Форма может быть и концептуальной, не имеющей геометрического представления. Факт вхождения одного изделия в состав другого, являющийся разновидностью формы, также может иметь одновременно несколько представлений, имеющих разную точность и разную степень детализации.

product_propert_representation_schema. Объекты этой схемы позволяют задать связь между свойством (в частности, формой изделия) и представлением свойств (например, геометрической моделью). Объекты этой схемы построены таким образом, что позволяют задавать между элементами множества свойств изделия и элементами множества представлений свойств отношения «многие ко многим». Для этой цели служат объекты `shape_definition_representation`, связывающий любое свойство изделия и представление свойства и `context_dependent_shape_representation`, связывающий отношений свойств (например, позиционирование узлов сборки) и представление отношений свойств (геометрические объекты, определяющие положение узла в пространстве).

Общие управляющие ресурсы представлены одной схемой – **management_resource_schema.** Объекты, определенные в этой схеме, являются абстрактными супертипами, подтипы которых должны связывать объекты, необходимые для управления (документы, время события, утверждение, сертификация, контракт и т.д.), с конкретным назначением этих объектов.

Кроме структур, которые образуют “каркас” описания изделия, в 41 том включены информационные объекты, описывающие общую для всех приложений информацию. Эти информационные объекты составляют ресурсы поддержки (`support_resources`), которые используются всеми моделями-сателитами. Ресурсы поддержки включают двенадцать схем, содержащих определения самых общепринятых и общераспространенных элементов.:

- схема действия (`action_schema`);
- схема утверждений (`approval_schema`);
- схема сертификации (`certification_schema`);
- схема контракта (`contract_schema`);
- схема даты и времени (`date_and_time_schema`);
- схема описания документов (`document_schema`).

Схема описания внешних ссылок (external_reference_schema);
 Схема групп (group_schema);
 схема физических единиц измерения (measure_schema);
 схема описания персонала и организаций (person_organization_schema);
 схема описания степени секретности (security_classification_schema);
 схема ресурсов общих видов определений (support_resource_schema).

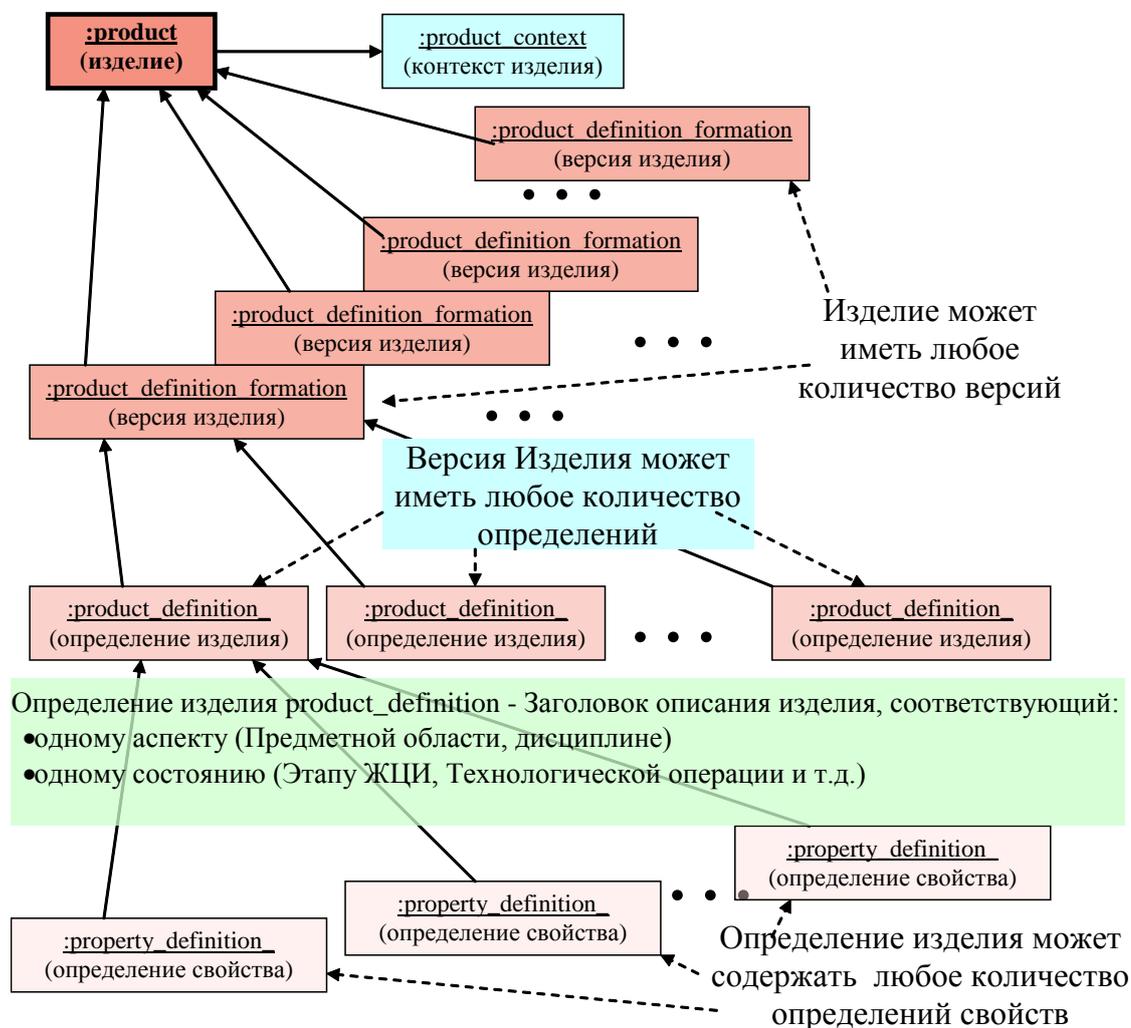


Рис. 43. Принцип построения описания изделия в STEP

Для описания изделия используются следующие сущности:

product – изделие

id – идентификатор изделия

name – название изделия

description – описание изделия

product_definition_formation – версия изделия

id – идентификатор версии изделия

description – описание версии изделия

product_definition_formation_with_specified_source – версия изделия с источником получения

make_of_buy – источник получения. Может принимать следующие значения: make (изготавливается), bought (покупается) и not_known (неизвестно)

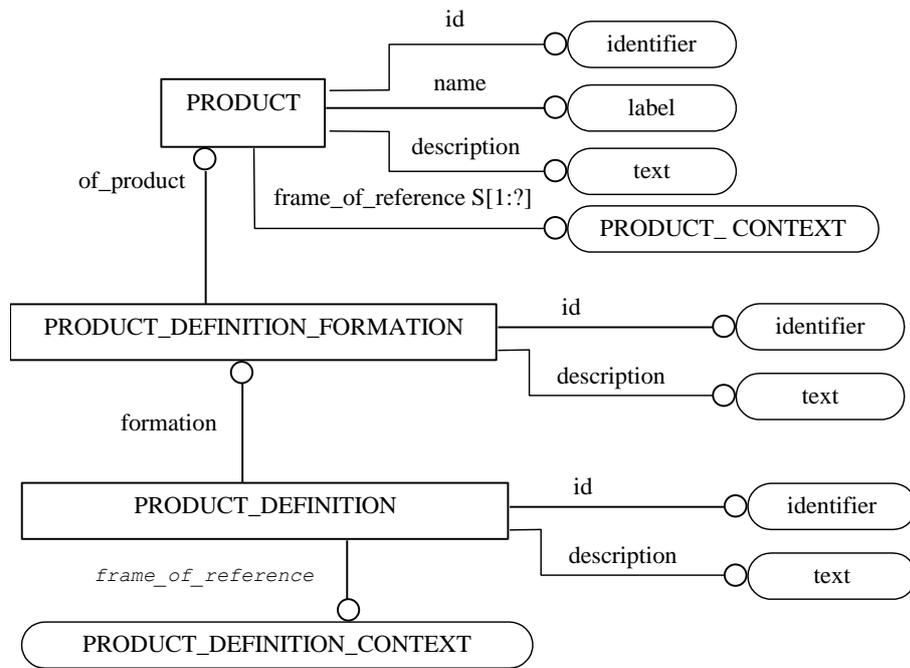


Рис. 44. Сущности, обеспечивающие определение изделия.

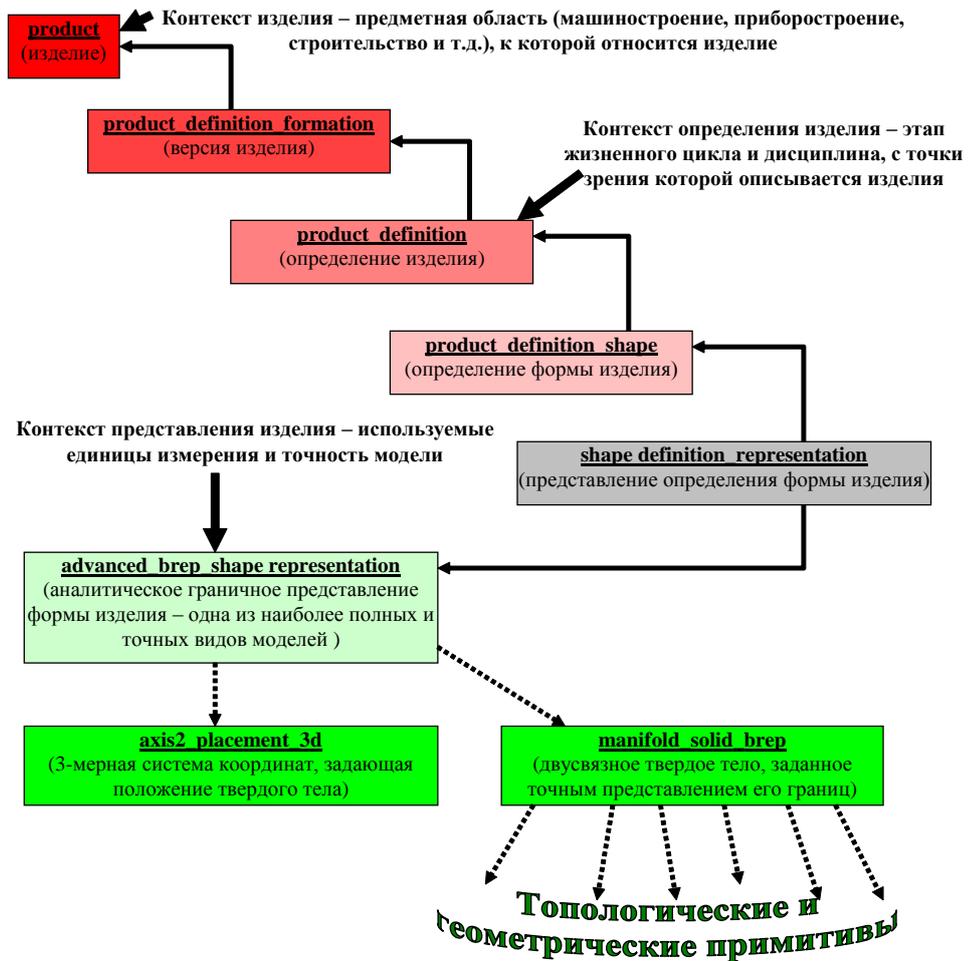


Рис. 45.

Модели представления описывают отдельные формы и свойства изделия, его использование и производство. Каждое изделие может иметь несколько “описаний” (product_definition), относящихся к различным этапам жизненно-

го цикла или к разным контекстам описания. Относящиеся к различным этапам жизненного цикла разновидности описаний одного изделия существуют одновременно и связаны с одним объектом - изделием. Это позволяет контролировать, насколько свойства изделия соответствуют проекту изделия и требованиям потребителя. Разновидности описания изделия, описывающие изделие на различных этапах его жизненного цикла, могут быть подтипами от описания изделия и соответственно иметь различные имена.

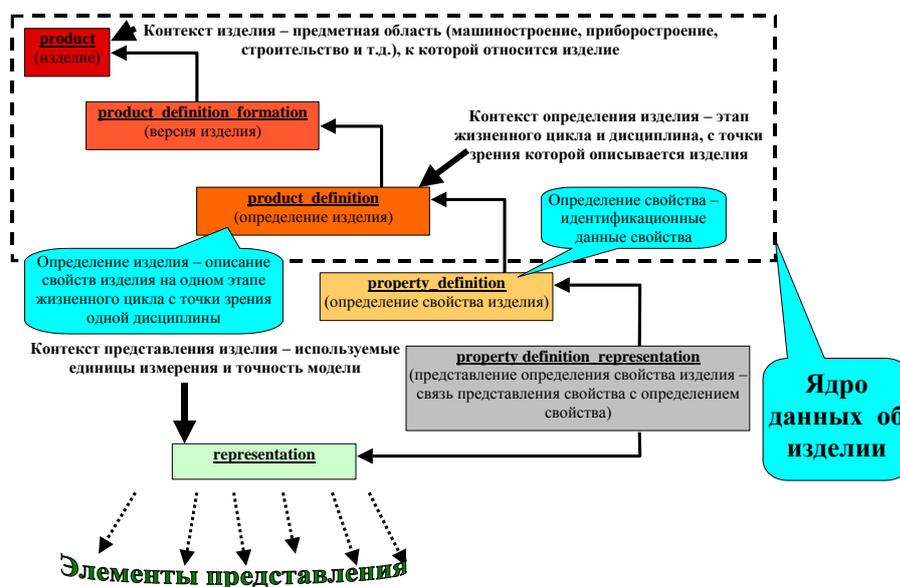


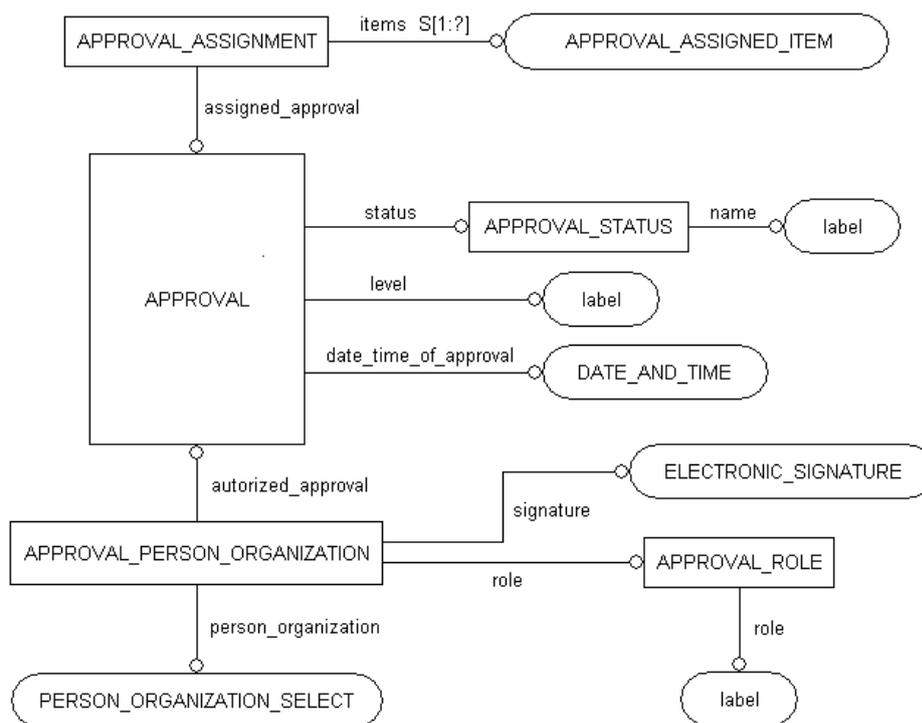
Рис. 46. Связь между ядром данных об изделии и представлением свойств изделия в обобщенном виде

Структура представления имеет контекст, содержащий информацию, относящуюся к структуре в целом. Предполагается, что контекст представления (**representation_context**) относится одновременно к нескольким структурам представления. Это свойство отражено в интегрированных ресурсах (см. ГОСТ Р ИСО 10303-43).

Любой набор информации об изделии может быть охарактеризован состоянием (статусом), являющимся результатом выполнения некоторого авторизованного действия. Примерами таких действий являются: *разработка, проверка, согласование, утверждение* и т.д.

Присвоение статуса аналогично подписанию бумажного документа, поэтому должны быть указаны роль и полномочия подписывающего лица.

Статус описывается объектом APPROVAL, который, в свою очередь, ссылается на объект APPROVAL_STATUS. Каждый экземпляр объекта APPROVAL_STATUS описывает один из возможных статусов. Общее число возможных статусов зависит от прикладной области. Лицо, присваивающее статус, указывается объектом APPROVAL_PERSON_ORGANIZATION. Данные, которым присваивается статус, указываются с помощью объекта APPROVAL_ASSIGNMENT.



APPROVAL	Присвоенный статус
Status	Ссылка на статус
Level	Уровень
date_time_of_approval	Дата присвоения статуса
APPROVAL_PERSON_ORGANIZATION	Лицо, присваивающий статус в соответствии с заданной ролью
person_organization	Сотрудник, присваивающий статус
Role	Роль сотрудника, присваивающего статус
signature	Электронно-цифровая подпись
APPROVAL_ROLE	Роль, выполняя которую лицо утверждает данные (присваивает им статус)
Role	Роль
APPROVAL_STATUS	Статус, присваиваемый объектам
Name	Наименование статуса
APPROVAL_ASSIGNMENT	Связь статуса с данными
assigned_approval	статус, который присваивается данным
Items	Данные, которым присваивается статус

Рис. 47. Взаимосвязь объектов, описывающих статус

Организация геометрических моделей в STEP.

Одно из основных и наиболее используемых в информационных технологиях свойств изделий – это его (геометрическая) форма.

В STEP задается связь ядра данных об изделии (относящихся к разряду т.н. PDM-данных) с его геометрической моделью (геометрической формой – shape). Для задания этой связи используются те же сущности и/или их подтипы, что и для связи ядра данных об изделии с представлением свойства в общем виде:

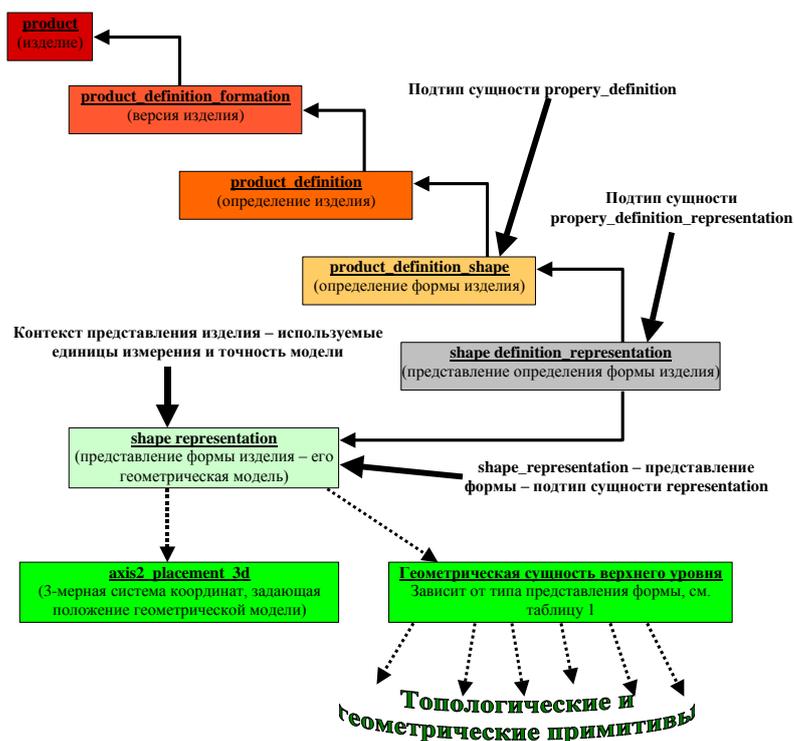


Рис. 48. Связь между ядром данных об изделии и его геометрической формой.

Геометрическое представление формы изделия – одно из основных представлений изделия в машиностроении. В компьютерных приложениях существуют три основных подхода к построению геометрических моделей. Все они представлены в томе 42.

Проволочные (каркасные) модели.

Модели, которые представлены набором кривых. Иногда могут добавляться отдельные поверхности, которые не образуют замкнутой оболочки.

Дают общее представление о форме изделия.

Требуют наименьших компьютерных ресурсов.

Поверхностные модели.

Модели, которые представлены набором поверхностей, образующих полностью или частично замкнутую оболочку. Поверхности, если они не замкнуты, должны быть полностью или частично обрезаны. К полностью замкнутым поверхностям относятся сфера и тор, частично замкнутые поверхности – конус и цилиндр. Сплайновые поверхности, т.е. поверхности, заданные набором точек, могут не обрезаться, поскольку их граница неявно определяется крайними точками. Также могут не обрезаться кинематические поверхности – линейчатая поверхность, поверхность вращения, поверхности заметания (например табулированный цилиндр).

Позволяют строить реалистичное изображение изделия с удаленными невидимыми линиями и поверхностями.

Требуют больших компьютерных ресурсов.

Твердотельные модели.

Модели, описывающие форму изделия в виде набора твердых тел, для которых однозначно определено, какая их часть заполнена веществом, а какая – пустая. Твердотельные модели могут быть представлены или в виде отдельных твердых тел, или в виде комбинации твердых тел, связанных логическими отношениями – объединением, пересечением и вычитанием. Способ представления формы изделия в виде набора твердых тел, связанных логическими отношениями, называется «конструктивная твердотельная геометрия» – CSG, “constructive solid geometry”. В AP203 метод CSG отсутствует, он есть в других AP, например, в автомобильном протоколе AP214.

Одно отдельное твердое тело может быть задано двумя способами: аналитически и граничным представлением.

Аналитические твердые тела описываются в виде уравнения $F(x,y,z) = 0$. При подстановке координаты конкретной точки в уравнение всегда можно определить, лежит ли точка внутри твердого тела ($F < 0.0$), на поверхности твердого тела ($F = 0.0$) или вне твердого тела ($F > 0.0$). В 42-м томе STEP аналитические твердые тела, как и аналитическое представление геометрии вообще, отсутствуют, они есть в некоторых AP, например, в строительном протоколе AP225.

Граничные твердые тела описываются замкнутой оболочкой, состоящей из граней. Принадлежность любой точки пространству твердому телу может быть определена методом пересечений: достаточно провести из точки бесконечный луч и посчитать количество его пересечений с оболочкой тела. При четном числе пересечений точка находится вне твердого тела, при нечетном числе пересечений – внутри твердого тела.

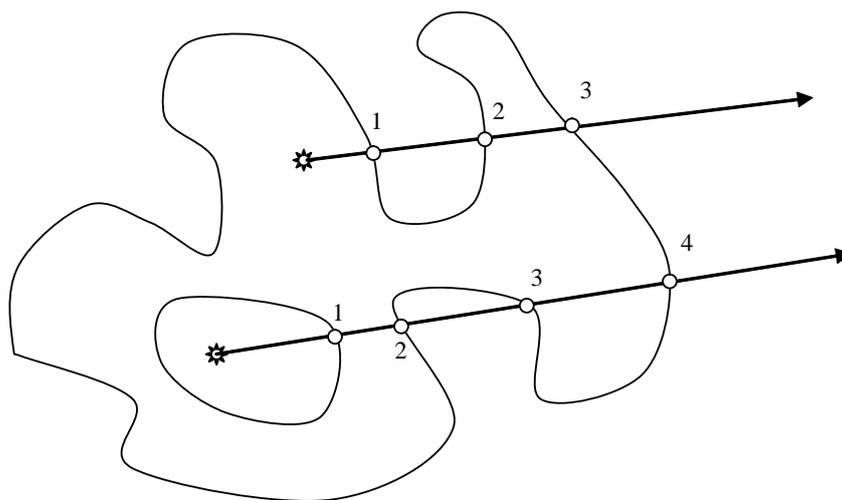


Рис. 49. Определение принадлежности точки грани с помощью метода пересечений

Существуют соглашения об ориентации геометрических объектов, образующих оболочку твердого тела: нормали к поверхностям должны быть на-

правлены от тела, направление контура против часовой стрелки означает внешний контур грани, по часовой стрелке – внутренние отверстия граней.

Разновидностью граничных твердотельных моделей являются фасетные (плоскогранные) модели, когда тело аппроксимируется набором плоских граней. Такие модели имеют, как правило, большие объемы, но за счет того, что все геометрические объекты описываются уравнениями только первой степени, обработка плоскогранных моделей происходит быстрее. Применимость плоскогранных моделей ограничена; например, здание или простейшая деталь могут быть описаны сравнительно легко, а при попытке представить в виде набора плоских граней фюзеляж самолета количество плоских граней может оказаться чрезмерно большим.

Твердотельные модели позволяют автоматически однозначно вычислять объемы, массы, моменты инерции изделия, контролировать непересекаемость изделий, готовить программы для обрабатывающего оборудования с ЧПУ, программы для стереолитографического оборудования, расчетные конечно-элементные модели.

Твердотельная модель требует наибольших компьютерных ресурсов.

Поверхностная модель, если она полностью замкнута, при этом в ней отсутствуют «щели» и ориентация отдельных поверхностей согласована между собой, может быть преобразована в граничную твердотельную модель с помощью так называемого процесса «сшивки» (“sewing”). Большинство современных САПР способны выполнять “сшивку” поверхностей в твердое тело.

	Геометрия			Топология	Дерево построений
	Точки	Линии	Поверхности		
Неструктурированный набор геометрических объектов	X	X	X		
Поверхностная модель	X	X	X	Неявно	
Проволочная модель	X	X		X	
Плоскогранная граничная модель	X	X	X	X	
Аналитическая граничная модель	X	X	X	X	
Конструктивная твердотельная геометрическая модель	X	X	X	X	X

Том 42 содержит ресурсы, определяющие средства геометрического моделирования.

Том 42 включает три схемы, содержащие:
определения геометрических сущностей;
определения топологических сущностей;
определение типов геометрических моделей.

В современных системах геометрического моделирования принято деление информационных объектов, образующих модель, на топологические и

геометрические объекты. Такой подход нашел отражение и в геометрии STEP. Геометрические модели STEP, описанные в ресурсах – двухсвязные («манифолд»). Манифолд (manifold)-модель – это такая модель, в которой каждая точка имеет открытую окрестность, гомеоморфную пространству соответствующей размерности.

Области X и Y являются гомеоморфными, если существует задающая взаимно-однозначное соответствие, непрерывная функция f между X и Y , такая, что существует обратная функция f^{-1} , и f^{-1} также является непрерывной.

Например, самопересекающиеся поверхности и кривые – не манифолд модели. В твердотельной манифолд-модели ребро всегда смежно точно с двумя гранями.

Короче говоря, манифолд-модели – это такие модели, которые соответствуют материальным физическим объектам и могут быть механически изготовлены. Не-манифолд модели – это абстрактные геометрические модели, являющиеся либо промежуточным этапом геометрических построений, либо специфическим приложением (сетка конечных элементов – не-манифолд модель), либо потребностью избежать излишней детализации модели (например, тонкие охлаждающие каналы в массивном блоке могут быть представлены осевыми линиями, а не цилиндрическими отверстиями).

Иногда встречается также термин «псевдо-манифолд». Иллюстрация того, что понимается под манифолд-моделями, псевдо-манифолд-моделями и не-манифолд моделями приведена ниже (Рис. 50).

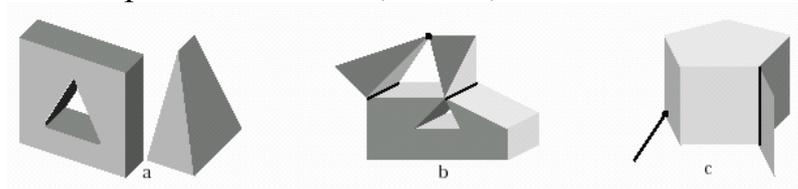


Рис. 50. Виды геометрических моделей: Манифолд (а); Псевдо-манифолд (b) ; Не-манифолд (с)

Существуют два способа определения геометрических сущностей:

- неявный, как уравнение $F(X_1, \dots, X_n) = 0$, где X_1, \dots, X_n – набор декартовых координат. Хотя такой способ является основным в классической геометрии, в компьютерной геометрии он практически не используется.;
- параметрический.

Геометрические сущности STEP – это точки, 2-х и 3-х мерные параметрические кривые и параметрические поверхности. В общем, набор геометрических сущностей, определенных в STEP, соответствует тому набору, который используется в большинстве современных геометрических систем. Предметом геометрической схемы **geometry_schema** является геометрия параметрических кривых и поверхностей:

- определение точек, векторов, параметрических кривых и параметрических поверхностей;
- определение конечных объемов с внутренней параметризацией;
- определение аффинных операторов преобразований;

- точки, определенные непосредственно значениями их координат или путем задания параметров существующей кривой или поверхности;
- определение конических кривых и простейших поверхностей;
- определение кривых, определенных на параметрической поверхности;
- определение общих параметрических сплайновых кривых, поверхностей и объемов;
- определение реплик точки, кривой и поверхности;
- определение эквидистантных кривых, и поверхностей;
- определение кривых, получаемых пересечением.

Отсутствуют линейчатые поверхности, что объясняется тем, что геометрия линейчатой поверхности критически зависит от параметризации граничных (направляющих) кривых и метода соединения пар точек, принадлежащих двум кривым. Однако, линейчатая поверхность, имеющая В-сплайновые граничные кривые, может быть точно представлена В-сплайновой поверхностью. Способ представления геометрических сущностей также схож, например, определение некоторых геометрических объектов полностью совпадает с определением геометрических сущностей геометрического процессора ACIS фирмы Spatial Technology или PARASOLID фирмы Siemens PLM Software.

Вся геометрия должна определяться в правосторонней прямоугольной Декартовой системе координат, имеющей по всем осям одинаковые единицы измерения. Для определения как двумерной, так и трехмерной геометрии используется общая схема. Точки и направления существуют как в двумерной, так и в трехмерной форме. Эти формы различаются строго по наличию или отсутствию значения третьей координаты. Все сложные геометрические сущности определяются с использованием точек и направлений, из анализа которых может быть получена размерность пространства.

Набор топологических сущностей STEP также соответствует общепринятому в настоящее время набору. Если сравнить, к примеру, представление топологических сущностей в процессоре ACIS или PARASOLID, представление топологических объектов согласно стандарту IGES 5.2 и представление топологических объектов согласно STEP ISO 10303-42, то можно отметить почти однозначное соответствие объектов ACIS и STEP и некоторые (впрочем, незначительные) отличия от них в стандарте IGES.

Случайно ли такое сходство? Очевидно, нет, и стандарты, такие, как STEP или IGES находятся в неразрывной связи с передовыми программными продуктами, такими, как ACIS или PARASOLID. И стандарты, и программные продукты, находятся в постоянном развитии, взаимно дополняя и обогащая друг друга. При разработке стандартов учитываются достижения разработчиков программного обеспечения, а разработчики программного обеспечения используют в своей работе стандарты.

Схема `geometric_model_schema` содержит сущности, определяющие различные типы геометрических моделей, т.е. геометрических сущностей выс-

шего уровня – твердых тел, плоских тел, геометрических наборов и т.д. В 42 томе STEP содержатся следующие аспекты геометрических моделей:

- данные, описывающие точную геометрическую форму трехмерных твердотельных объектов;
- модели, соответствующие конструктивной твердотельной геометрии (CSG);
- модели, соответствующие конструктивной твердотельной геометрии, в двумерном пространстве;
- определение примитивов CSG (БЭФ - базовые элементы формы) и полупространств;
- создание твердотельных моделей посредством операций заметания;
- модели с манифолд - граничным представлением (B-ger);
- ограничения, проверяющие целостность граничных манифолд-моделей;
- поверхностные модели;
- каркасные (проволочные) модели;
- геометрические наборы;
- создание репликаций твердотельных моделей в новой позиции.

Следующее не входит в область действия 42 тома:

- модели с не-манифолд граничным представлением. Модели таких типов могут быть представлены в других томах, например, такие модели допускаются в AP214;
- разновидности твердотельных моделей, описываемые через занимаемое ими пространство (такие, как дерево октантов);
- сборочные единицы и механизмы. Это охватывается 44 томом STEP.

***advanced_boundary_representation* аналитическая граничная модель**

Аналитическая граничная модель является наиболее полным, точным и однозначным представлением формы изделия. Эта модель содержит представление формы изделия с помощью граничной твердотельной модели. Топологический объект «твердое тело» состоит из одной внешней оболочки и произвольного числа внутренних оболочек, задающих внутренние пустоты. Каждая из оболочек содержит набор граней, грани ограничены контурами, состоящими из ребер, каждое ребро имеет начальную и конечную вершины. Границы всех объектов явно определены только топологией. Все геометрические объекты, описывающие форму изделия, должны быть связаны с топологическими объектами: поверхности связаны с гранями, кривые связаны с ребрами, точки связаны с вершинами. (Это так называемая «подкладная» - “underlying” геометрия).

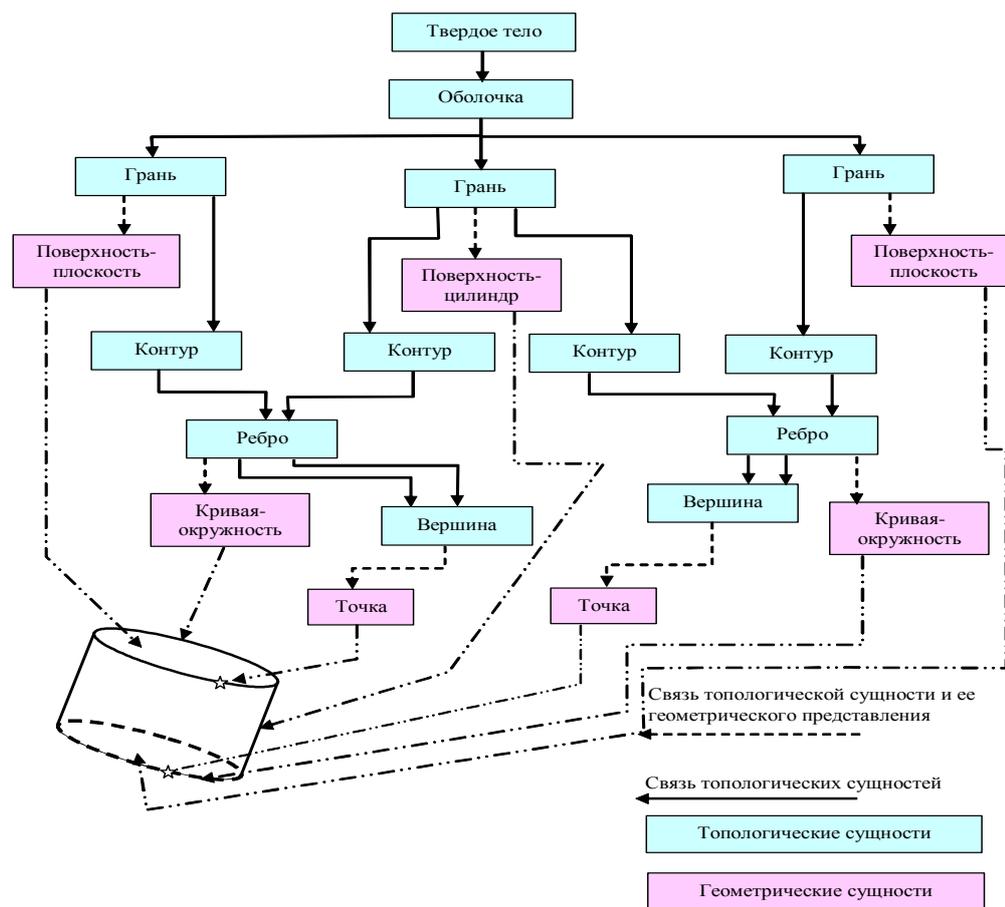
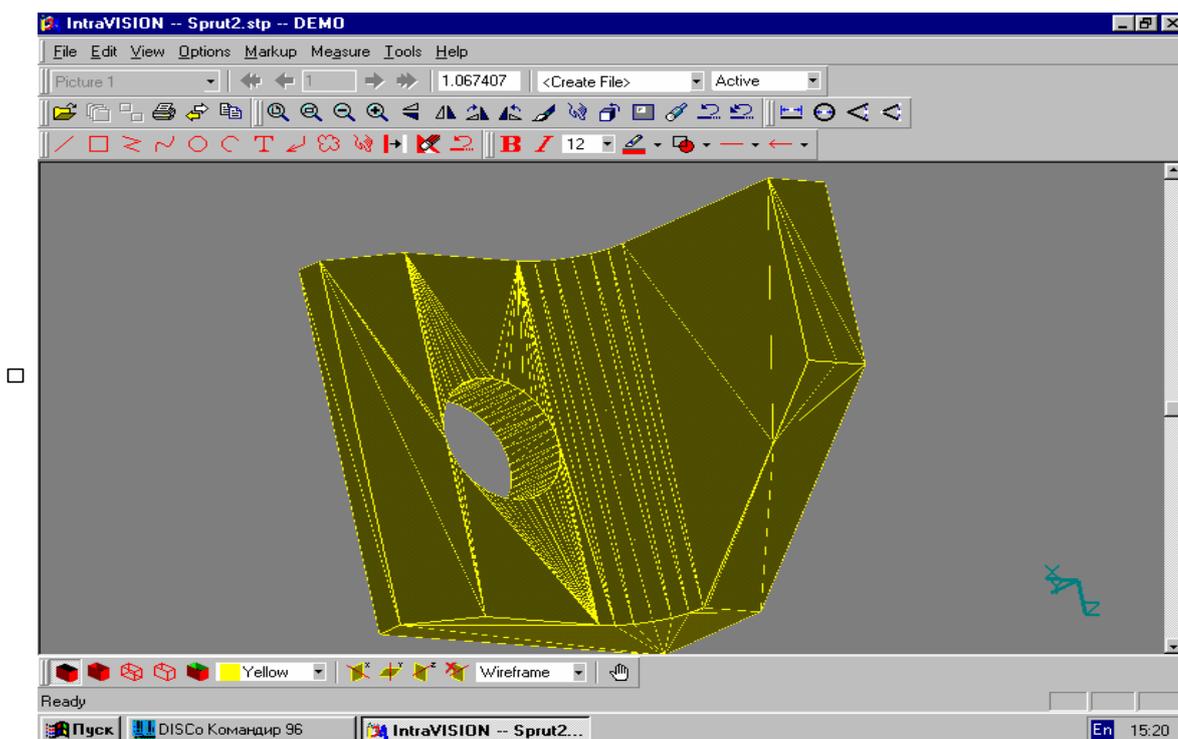


Рис. 51. Структура данных аналитической граничной твердотельной модели

***faceted_boundary_representation* плоскогранная граничная модель**

Плоскогранная (фасетная) граничная модель содержит такое представление формы изделия, в котором граничными поверхностями твердотельной модели являются плоскости, ограниченные ломаными линиями. При этом большая часть объектов задается неявно (например, контур грани задается списком точек, а явное представление прямых ребер отсутствует).



В плоскогранной граничной модели геометрия изделия аппроксимируется набором плоских граней, ограниченных ломаными линиями.

В STEP структура данных плоскогранной граничной модели сходна со структурой данных аналитической граничной модели.

Рис. 52. Плоскогранная твердотельная модель

Преимуществом фасетных моделей является возможность их быстрой обработки, т.к. вся геометрическая модель строится на основе уравнений первого порядка, а недостатком – низкая точность. В последнее время в связи с ростом вычислительных возможностей современных компьютеров вышеуказанное преимущество перестает быть актуальным, и фасетные модели используются достаточно редко. Например, STEP-конверторы CAD-систем CATIA и Unigraphics NX такие модели могут читать, но не генерируют.

***wireframe_with_topology* проволочная модель с топологией**

Проволочная (каркасная) модель с топологией содержит представление формы изделия с помощью проволочной модели, определенной топологией ребер. Данный функциональный блок включает 3-мерные кривые и топологические объекты.

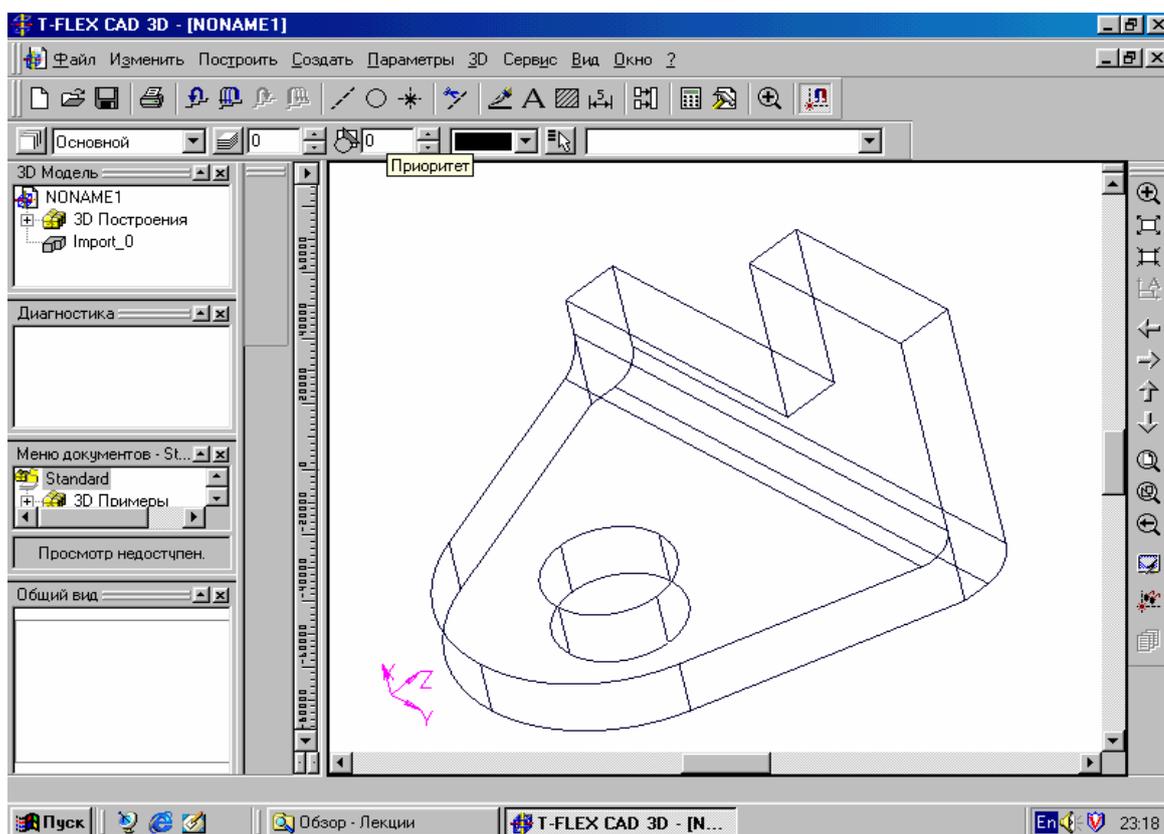


Рис. 53. Проволочная модель

***manifold_surface_with_topology* манифолд-поверхность с топологией**

Отличие этой модели от первых двух состоит в том, что здесь отсутствует объект «твердое тело». Манифолд-поверхности с топологией содержат представление формы изделия, использующее манифолд-поверхности с топологией. Внешняя граница изделия определяется 3-х мерными кривыми, поверхностями и топологическими объектами.

***non_topological_surface_and_wireframe* поверхности без топологии и проволочная модель**

Этот вид моделей содержит представление формы изделия в виде неструктурированного набора геометрических объектов. Геометрическими объектами могут быть точки, кривые и поверхности. Границы кривых определяются явно точками на кривых и явной связью между точками и кривыми, которые этими точками ограничиваются. Границы поверхностей определяются кривыми на поверхностях и явной связью между кривыми и поверхностями, которые этими кривыми ограничиваются. Поверхности и кривые, если они не замкнуты, должны быть явно обрезаны.

***csg_model* конструктивная твердотельная геометрия**

Форма конструктивной твердотельной геометрической модели образуется в результате вычисления дерева конструктивной твердотельной геометрии, в котором листьями являются твердотельные объекты, а узлами - булевы операции (объединение – «или», пересечение – «и», вычитание – «не»). Эта модель обеспечивает геометрическую функциональность для построения твердых тел с использованием булевых операций над ранее созданными исходными простыми параметризованными твердотельными формами, твердотельными **manifold** граничными формами в фасетном или аналитическом виде, и полупространствами.

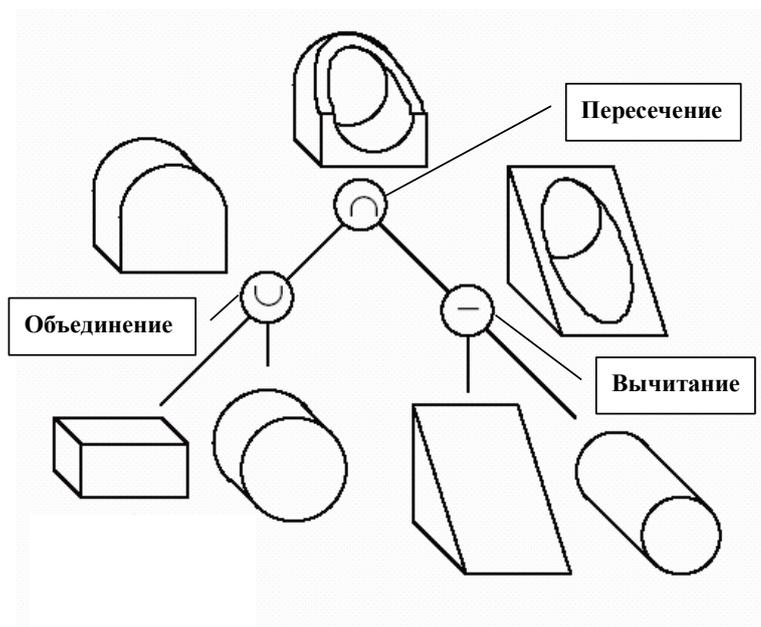


Рис. 54. Конструктивная твердотельная геометрическая модель

Несмотря на несомненную привлекательность передачи информации такого вида, CSG-модели STEP-конверторами современных CAD-систем не поддерживаются по следующим причинам:

- не всегда пользователям желательно передавать своим партнерам не только конечную форму изделия, но и информацию о том, как эта форма была построена, т.к. использованные при построении приемы работы могут содержать профессиональную тайну;
- нет гарантии, что принимающая система, воспроизведя булевы операции, построит точно такую же форму изделия, как и в передающей системе, поскольку алгоритмы геометрических построений в системах могут различаться.

Том 43 Организация связи изделия и его представлений.

Том 43 включает базовые понятия, позволяющие создавать при разработке прикладных протоколов связи между общей моделью и моделями-представлениями. Для геометрических представлений содержатся сущности,

позволяющие задать положение компонента сборочной единицы относительно системы координат сборочной единицы.

Том 44 Состав и конфигурация изделия

Том 44 описывает отношение типа “часть-целое”, когда одно изделие состоит из нескольких компонентов. Содержащаяся в томе 44 схема `product_structure_schema` поддерживает многоуровневый конструкторско-технологический граф общего вида.

В STEP принято, что изделие в целом, комплекс, комплект, узел, сборочная единица, деталь, стандартное изделие и т.д. соответствуют одной сущности – **product** (изделие). Является ли изделие сборочной единицей или деталью, может быть выявлено из анализа конструкторского графа. Те изделия, в которые не входят никакие другие изделия, являются деталями. Дополнительно к этому, возможно уточнение этой информации с применением концепции «категория изделия». Например, то, что изделие является покупным или стандартным изделием, может быть задано с помощью связи изделия с соответствующей категорией «покупное изделие» или «стандартное изделие».

Основная сущность, описанная в схеме, `product_definition_usage`, является подтипом объекта `product_definition_relationship` и описывает связь двух изделий.

В свою очередь два подтипа этой сущности позволяют построить описание либо общей спецификации изделия (`bill_of_material`) (`make_from_usage_option`) либо иерархическую структуру изделия с указанием порядка сборки и позиций на сборочном чертеже (`assembly_component_usage`).

Том определяет `integrated resources`, используемые для представления информации, необходимой, чтобы определить структуру изделия и управлять конфигурацией изделия. Он содержит следующую информацию :

`bill of material`,

спецификацию изделий,

Главные подразделы этого тома (ISO 10303-44):

* `the product structure schema`;

* `the product concept schema`;

* `the configuration management schema`.

`Product structure schema`. Назначение.

* Определяет изделие в терминах состава как набор элементов или других изделий то есть изделие может быть собрано из элементов или произведено, потребляя другие изделия, или и то и другое.

Определяет механизмы для выражения отношений состава.

В томе 44, определяющем структуру изделия, существуют средства, позволяющие задать отношения между двумя изделиями. Отношения эти могут быть трех типов:

одно изделие входит в состав другого изделия;

одно изделие может заменить другое изделие, например, в составе сборочной единицы;

одно изделие может быть изготовлено из другого изделия.

Все вышеуказанные отношения являются свойствами изделий, зависящими от контекста, в котором рассматриваются изделия, в том числе, от этапа Жизненного Цикла. Поэтому вышеуказанные отношения связаны с сущностями **product_definition** (PD), определяющими свойства изделия, соответствующие данному контексту.

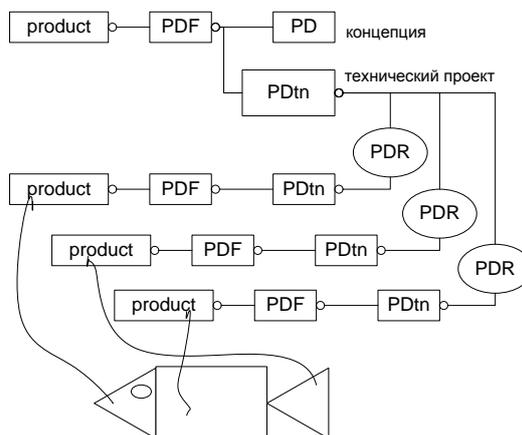


Рис. 55. Различные варианты структуры изделия в зависимости от этапа Жизненного Цикла.

На рисунке выше приведен пример, показывающий что одно и то же изделие, рассматриваемое на этапе концептуального проектирования как единичное изделие, на этапе технического проекта представляется как сборочная единица (СЕ), состоящая из трех компонентов.

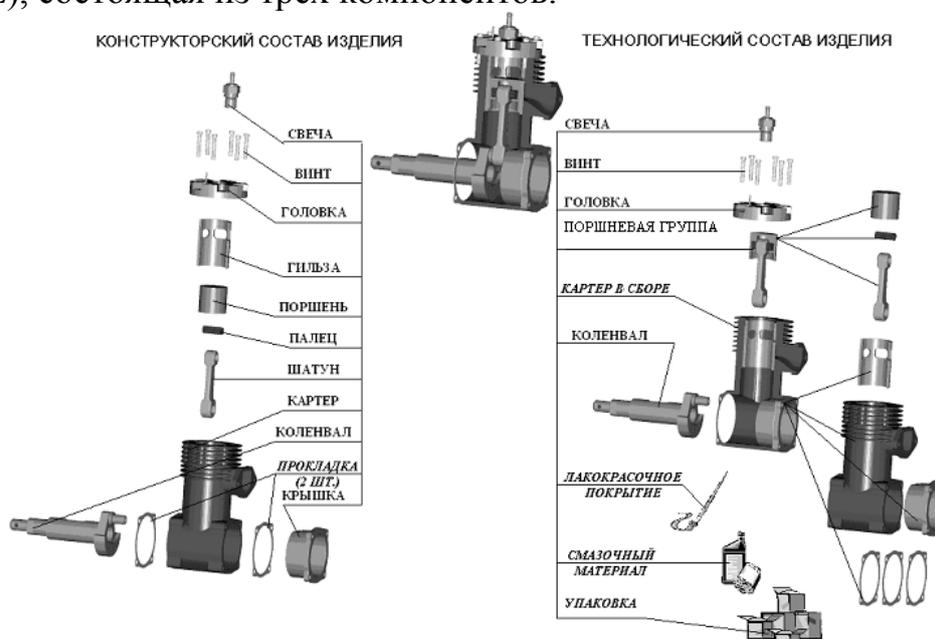


Рис. 56. Конструкторский и технологический составы изделия

Также структура изделия может различаться с точки зрения разных предметных областей (конструкторской и технологической).

Все три типа отношений, которые определены в томе ISO 10303-44, используются и в Протоколе Применения AP203.

Сущность **assembly_component_usage** устанавливает связи между конструкциями с помощью одного из четырех своих подтипов:

сущность **quantified_assembly_component_usage** представляет связь между составляющей сборки и сборкой. Для случая дискретных составляющих несколько вхождений составляющей представлено простой составляющей и числом, задающим количество этих составляющих. В качестве примера можно привести указание о количестве входящих в сборку крепежных элементов, например, болтов, винтов, заклепок и т.д. Для недискретных составляющих количество представлено единицей измерения, отличной от безразмерного числа. Здесь примером могут служить используемые в сборке материалы типа изоляции, смазки, краски, жидкостей и т.д.;

сущность **next_assembly_usage_occurrence** (NAUO) представляет отношение между одним вхождением составляющей и той сборкой, в которую она непосредственно входит. Экземпляр сущности **next_assembly_usage_occurrence** позволяет идентифицировать каждое из вхождений одного и того же компонента сборочной единицы в саму сборочную единицу, например:

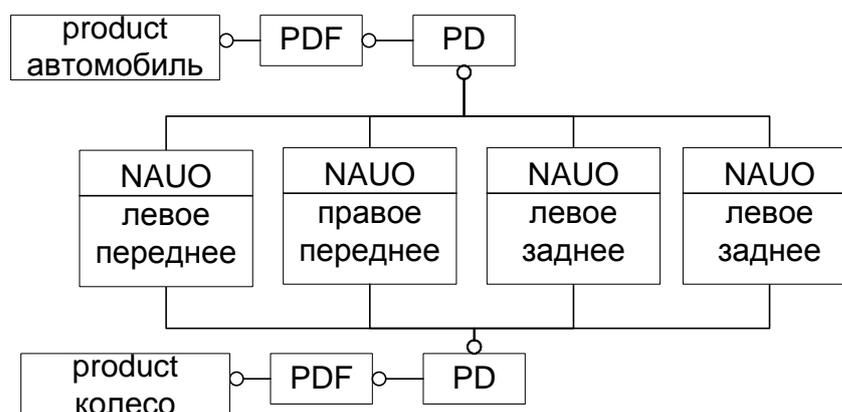


Рис. 57. Компонент сборочной единицы (колесо) входит в сборочную единицу (автомобиль) четыре раза

В приведенном примере каждое из вхождений одного и того же изделия (колеса) в автомобиль имеет свою идентификацию, что позволяет учесть особенности каждого из вхождений;

сущность **specified_higher_usage_occurrence** представляет связь между одним вхождением составляющей в сборочную единицу верхнего уровня. С помощью экземпляров этой сущности можно увязать воедино многоуровневые сборочные единицы;

сущность **promissory_usage_occurrence** представляет связь между составляющей и сборкой верхнего уровня, когда структура изделия, включающая промежуточные сборки, еще не проработана.

Структура изделия моделируется направленным ациклическим графом (DAG). В этих моделях узлы представляют определения изделий, а направленные дуги представляют отношения входимости. В схеме **prod-**

uct_structure_schema узлы соответствуют сущностям **product_definition**, а дуги соответствуют сущностям **assembly_component_usage**.

Сущность **make_from_usage_option** («сделано из») может связывать два описания, относящиеся как к одному и тому же изделию, так и к разным изделиям:

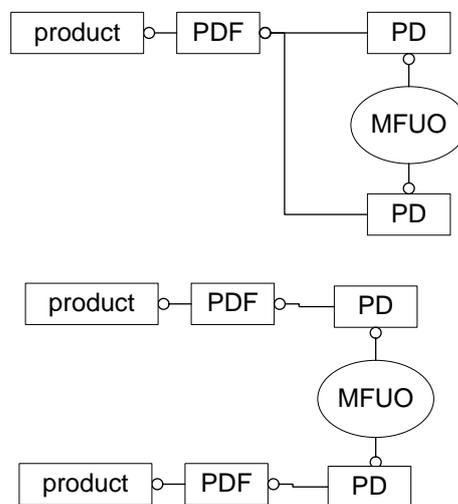


Рис. 58. Два варианта применения отношений «сделано из»

Первый из вариантов позволяет задать связь между различными состояниями одного и того же изделия, например, в процессе его изготовления. Второй вариант показывает связь между изделием и заготовкой (например, сортаментом).

Product concept schema

Представляет концепцию изделия как набор спецификаций изделия, полученного из анализа потребностей клиента в изделии.

Configuration management schema

Описывает изделия, участвующие в изготовлении другого изделия, чьи конфигурации находятся под прямым управлением.

В реальном производстве, независимо от того, является ли производство единичным (опытным), серийным, или массовым, существует множество версий изделий всех уровней.

Под версией изделия понимается отличающийся вариант структуры изделия, имеющий тот же идентификатор, что и исходное изделие. Различные версии могут появляться или в результате отработки конструкции, или же в результате необходимости учета различных потребностей заказчика.

Предметом схемы **configuration_management_schema** является установление связи соответствующих версий изделия при планировании производства. На изделие, готовящееся к производству, делаются ссылки как на **configuration_item**. Это обычно очевидно для заказчиков организации, осуществляющей управление конфигурацией.

К управлению конфигурацией применимы следующие концепции:

В пределах организации управление конфигурацией может применяться к изделиям и компонентам изделий, изготавливаемым в этой организации. На

компоненты изделий, получаемые от поставщиков, управление конфигурацией не распространяется;

Организация определяет, конфигурация каких изделий должна находиться под управлением. Эти изделия становятся узлами конфигурации в данной организации. Это – функциональные элементы верхнего уровня, служащие фокусными точками для управления применяемостью составляющих их элементов более низкого уровня.

Также эта схема обеспечивает возможность записи связей, устанавливаемых для сопряжения ассоциаций.

Управление конфигурацией осуществляется с помощью сущностей `configuration_item`, `configuration_design`, и `configuration_effectivity`.

Схема управления конфигурацией содержит следующие концепции:

Обозначение узлов конфигурации (`configuration_item`) и соответствующих концепций изделия, из которых формируется состав.

Обозначение версии изделия (`product_definition_formation`), реализующей узел конфигурации.

Создание связей соответствующих версий изделий для построения узла конфигурации. Ссылки на эти связи делаются как на сущности применимость (`configuration_effectivity`).

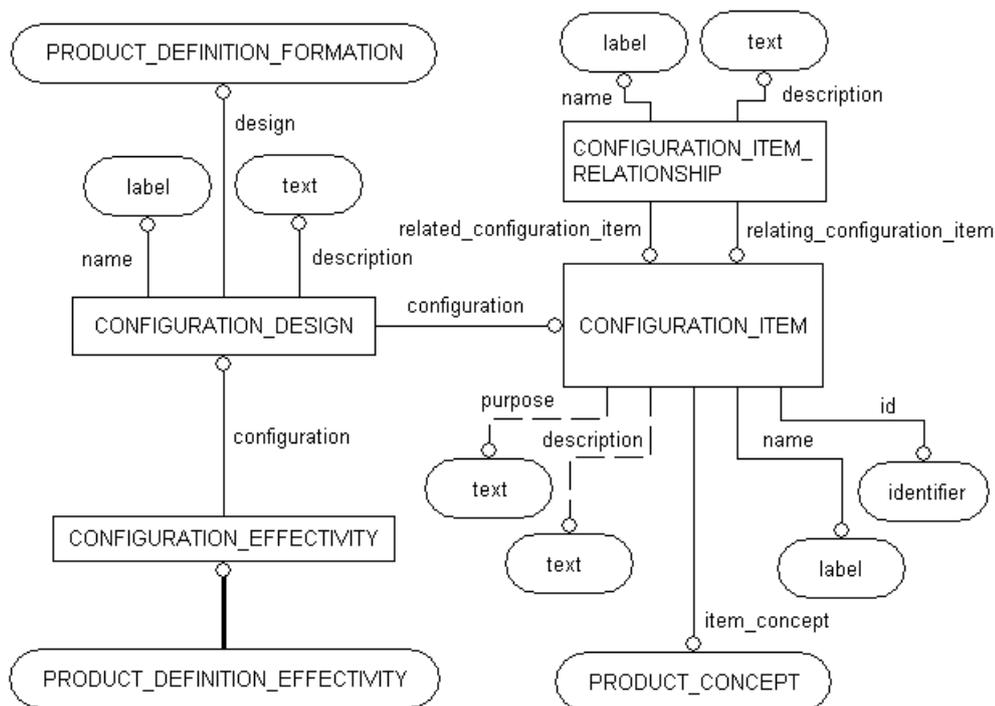
Сущность **`configuration_item`** (узел конфигурации) используется для управления сборкой из составляющих реальных объектов производства. Все управление конфигурацией в организации осуществляется с использованием этих сущностей.

Узел конфигурации может относиться или к изделию в целом или к некоторому компоненту изделия.

Узел конфигурации может быть создан тогда, когда соответствующая версия изделия **`product_definition_formation`** еще не существует.

Связь между узлом конфигурации и соответствующей ему версией изделия устанавливается с помощью сущности **`configuration_design`** (конструкция конфигурации).

Узел конфигурации связан с одной концепцией изделия **`product_concept`**.



PRODUCT_CONCEPT	Концепция изделия
Id	Идентификатор
Name	Обозначение
Description	Описание
Market_context	Сегмент рынка
CONFIGURATION_ITEM	Объект конфигурации
Id	Идентификатор
Name	Обозначение
Description	Описание (опция)
Item_concept	Ссылка на концепт изделия
Purpose	Назначение (опция)
CONFIGURATION_ITEM_RELATIONSHIP	Отношение между объектами конфигурации
Name	Наименование отношения
Description	Описание отношения
relating_configuration_item	Ссылка на родительский объект конфигурации
related_configuration_item	Ссылка на дочерний объект конфигурации
CONFIGURATION_EFFECTIVITY	Правило применения конфигурации
EFFECTIVITY	Применяемость
product_definition_effectivity	Объект, для которого задано правило применяемости
serial_numbered_effectivity	Применяемость по серийному номеру
dated_effectivity	Применяемость по дате
lot_effectivity	Применяемость по обозначению партии
Id	Обозначение правила применяемости
Usage	Ссылка на отношение, для которого задается применяемость
effectivity_start_id	Серийный номер изделия, начиная с которого действует данное правило применяемости.
effectivity_end_id	Серийный номер изделия, заканчивая которым действует данное правило применяемости.
effectivity_start_date	Дата, начиная с которой действует данное правило применяемости.
effectivity_end_date	Дата, заканчивая которой действует данное правило применяемости.
effectivity_lot_id	Обозначение партии, для которой действует данное правило применяемости.

Рис. 59. Взаимосвязь объектов, используемых в управлении конфигурацией изделия

Том 45 содержит информационные ресурсы, необходимые для описания свойств материала. Поскольку свойства материала в настоящее время коммерчески доступными препроцессорами и постпроцессорами STEP не обрабатываются, данный том рассматриваться здесь не будет.

Том 46 содержит информационные ресурсы, необходимые для описания особенностей воспринимаемого представления изделия *presentation*. Согласно общим определениям, воспринимаемое представление *presentation* – это такое представление, которое может быть воспринято человеческими чувствами, т.е. которое можно увидеть, услышать, пощупать, понюхать. Естественно, что на практике речь идет только о визуальном представлении.

Поскольку STEP предназначен, в первую очередь, для передачи и хранения модели изделия, предназначенной не для просмотра человеком, а для компьютерной обработки, сущности из 46 тома пока не имеют столь широкого применения, как сущности 41, 42, 43 и 44-го томов. Тем не менее, уже существуют коммерческие препроцессоры и постпроцессоры STEP, например, в CAD-системах CATIA V5, Unigraphics NX, T-Flex CAD, которые способны обрабатывать информацию о цвете изделия, что относится к элементам воспринимаемого представления.

Том 49 содержит ресурсы, необходимые для представления действий. В первую очередь под действиями понимаются потоки работ по разработке, изготовлению и эксплуатации изделий и технологические процессы. Сущности тома 49 используются для представления технологических процессов в Протоколе AP214 и специализированных технологических протоколах AP238 и AP240. Ниже даны общие принципы представления в STEP технологических процессов.

Создание информационной модели производственных процессов в томе 49 ISO 10303 является развитием работ, которые ведутся Техническим Комитетом TC184 ISO в рамках стандарта ISO 14258 (создание общей модели предприятия).

По тому, что содержание тома 49 претерпело начиная с 1992 года наибольшее число изменений, можно судить, что создание базовой модели дискретного процесса вызывает у разработчиков STEP наибольшие проблемы.

Основная проблема - предложить базовую информационную модель, которая позволила бы выделить общие свойства процессов создания изделий на любых стадиях Жизненного Цикла и создать структуру, позволяющую хранить и общие и уникальные свойства элементов процессов.

Основные принципы описания процессов в томе 49 следующие:

реально протекающий процесс состоит из дискретных элементов (действий - action), каждый из которых имеет конкретные моменты начала и окончания;

множество действий, выполненных с различными объектами в различное время связаны с одним экземпляром объекта *action_method*, идентификатором описания выполнения действия (например, операцией технологического процесса);

связи действия action и action_method описаны ориентированным графом, допуская произвольное число связей различного типа между экземплярами;

с action_method связаны требования к ресурсам, необходимым для его реализации;

с действием action и action_method связаны ресурсы, участвующие в выполнении. Причем связи различных ресурсов с экземпляром действия action даются с указанием статуса связи - “требуемая”, “запланированная”, “используемая”, “реализованная”.

Связь элементов технологического процесса с оборудованием, объектом воздействия, методом.

В данном разделе показано, как средствами Протокола AP214 строится модель технологического процесса сборки изделий. Сущности action, action_relationship и т.д. здесь заимствованы из тома 49.

Модель технологического процесса содержит:

данные о технологическом процессе и его элементах (этапах работ, операциях, переходах, проходах);

структуре технологического процесса (последовательность выполнения и вложенность);

связи элементов технологического процесса с ресурсами (организациями и лицами, оборудованием, инструментом), с предметом труда (промежуточными состояниями фрагментов изделия).

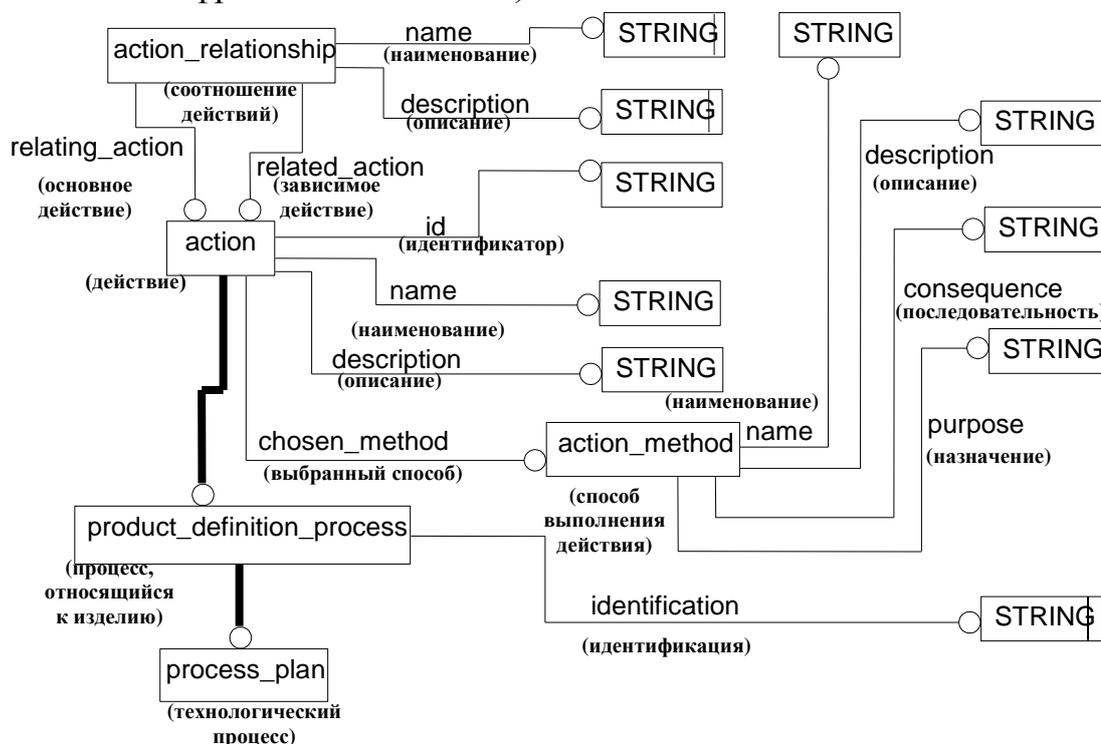


Рис. 60. Сущности, описывающие структуру и идентификацию процесса

Основа модели технологического процесса – это сущности:

action (в виде ее подтипов process_plan – технологический процесс и product_definition_process – элемент технологического процесса);

action_relationship (отношение действий). Экземпляры сущностей могут образовывать следующие элементарные структуры:

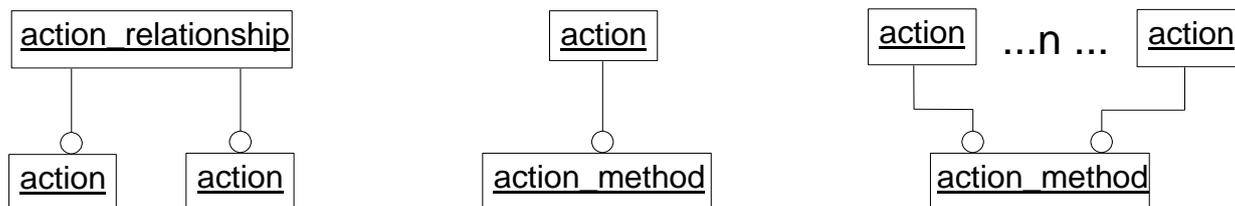


Рис. 61. Соотношения между экземплярами сущностей, описывающих процесс

Из рисунка видно, что:

структура технологического процесса может быть построена с использованием сущностей action в качестве узлов графа и action_relationship в качестве ребер;

один экземпляр сущности action_method может быть связан с множеством экземпляров сущностей action.

Модель технологического процесса должна отражать:

вхождение одних элементов в другие (этапов работ в технологический процесс, операций в этапы работ и т.д.);

последовательность элементов;

классификацию элементов.

Следовательно, представленные экземплярами action_relationship ребра графа могут быть трех типов. Примем, что тип ребра определяется атрибутом name, который может принимать одно из следующих фиксированных значений:

composition для выражения того, что один элемент технологического процесса является составной частью другого элемента;

next для выражения того, что один элемент технологического процесса следует непосредственно за другим элементом или **sequence** для выражения того, что элемент технологического процесса должен выполняться после того, как выполнен другой элемент технологического процесса;

classification для выражения того, что один элемент является подтипом другого элемента (например, фрезерование является подтипом механической обработки). В конечном технологическом процессе могут присутствовать только те элементы, которые являются листьями классификатора. Прочие элементы (абстрактные элементы технологического процесса могут появляться только в Базе Знаний типовых технологических процессов.

Уточнение описания

Сущность action_method дает возможность расширенного и более обобщенного описания элемента технологического процесса (за счет того, что один экземпляр action_method может обеспечивать уточненное описание более чем одного экземпляра action).

Сущности `action_property` и `action_property_representation` позволяют связать `action`, `action_method` и `action_relationship` с описывающей этот экземпляр моделью `representation`.

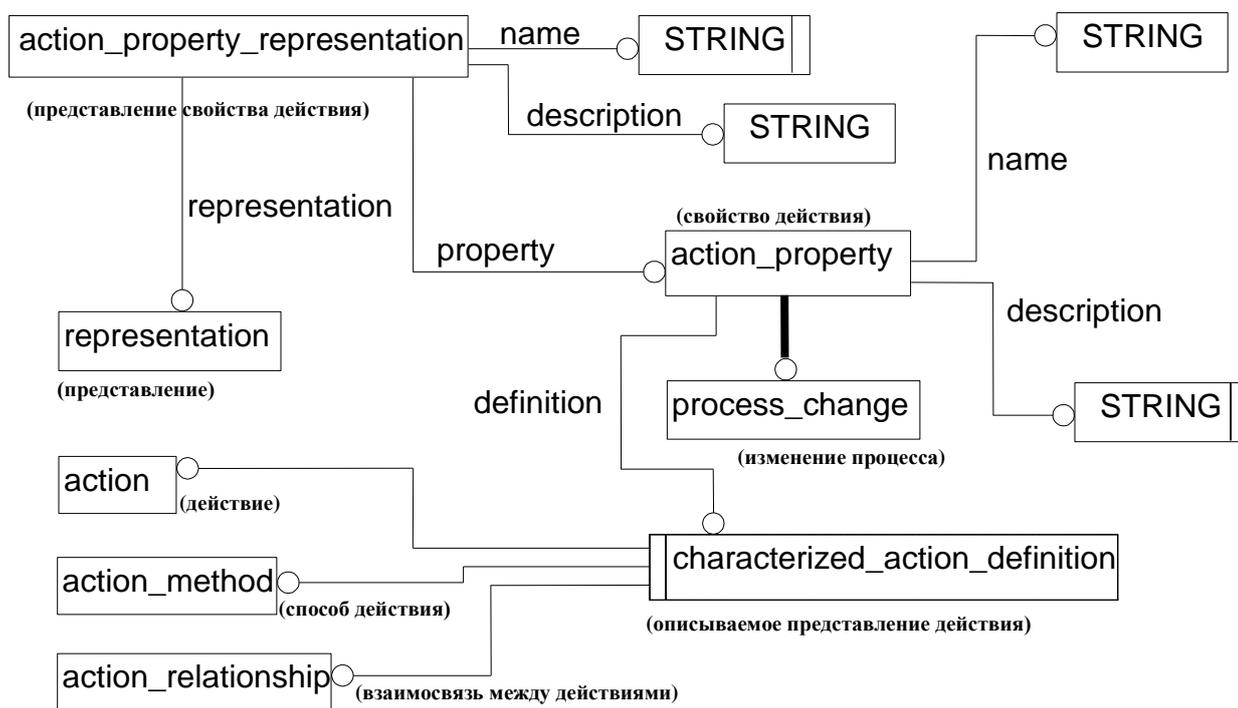


Рис. 62. Сущности, обеспечивающие описание свойств процесса

Типом модели `representation` может быть `property_value_representation` (представление значения свойства). Эта модель обладает, согласно спецификации, следующими свойствами:

модель содержит не более 3-х экземпляров сущностей типов `measure_representation_item` (метрический элемент представления) или `descriptive_representation_item` (описательный элемент представления);

модель содержит не более экземпляра сущности с именем “lower”. Этот экземпляр задает нижний предел значения свойства;

модель содержит не более экземпляра сущности с именем “upper”. Этот экземпляр задает верхний предел значения свойства;

модель содержит строго один экземпляр сущности с именем “specified_value”;

модель содержит информацию об имени значения, например, “максимальная длина” или “минимальный объем” и интерпретацию, например, “требуемое”, “сконструированное”, “оцененное”, “измеренное”.

При этом с помощью сущностей, взятых из других информационных ресурсов (тома 41, 43, 44) обеспечивается связь действий с изделием (для технологического процесса в целом) или некоторыми его элементами (технологические операции связываются с элементами формы – конструктивно-технологическими элементами).

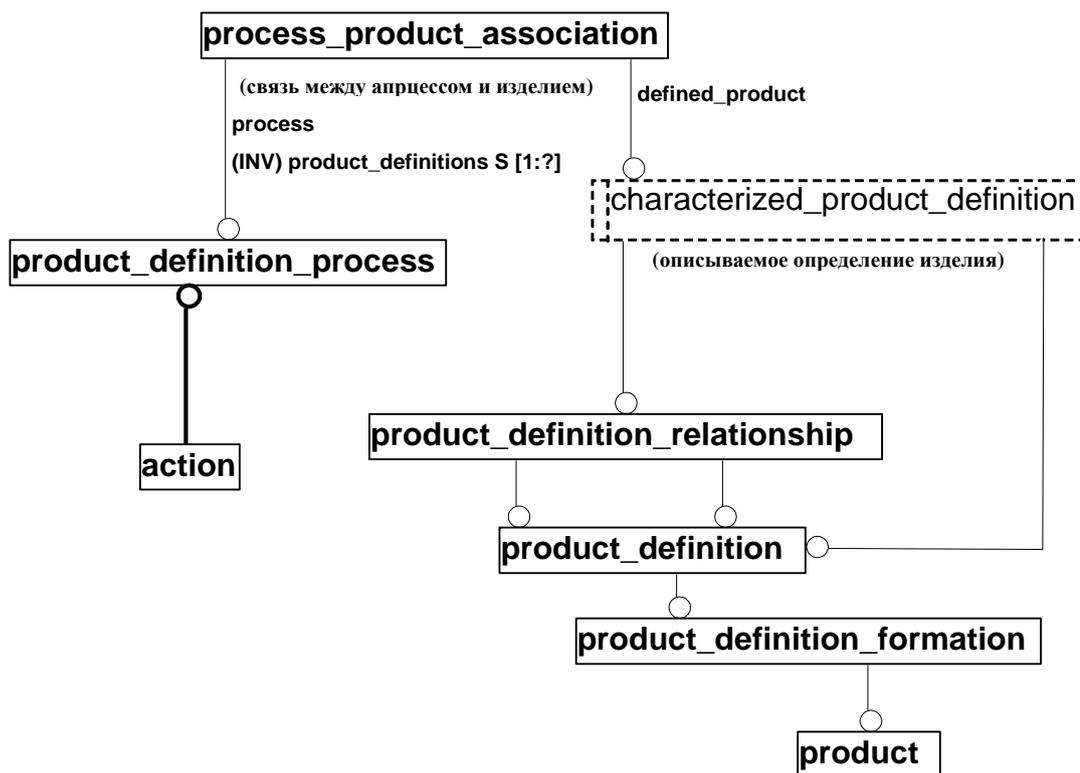


Рис. 63. Связь технологического процесса `product_process_plan` с изготавливаемым изделием.

Для того, чтобы избежать излишнего усложнения схемы, на рисунке часть сущностей изображена упрощенно.

Контрольные вопросы

16. Назначение интегрированных информационных ресурсов STEP.
17. Какие типы геометрических моделей содержатся в информационных ресурсах STEP.
18. Как в STEP представляются структура и состав изделия.

9. Прикладные протоколы STEP

В настоящее время наиболее широко используемыми Прикладными протоколами являются AP203 (Конструкция с управляемой конфигурацией) и AP214 (Ядро данных для автомобильной промышленности). Протокол AP214 предоставляет достаточно широкие возможности по передаче рабочего проекта изделия, однако в CAD-системах коммерчески доступные препроцессоры и постпроцессоры STEP поддерживают только то подмножество AP214, которое совпадает с AP203.

Ниже даны сведения по протоколу AP214 и некоторым другим протоколам.

Протокол **AP214** ориентирован на использование в автомобильной промышленности. Также возможно его использование в промышленности, про-

изводящей наземные транспортные средства (Поезда, трактора, строительные машины).

Surface_condition	C	C1	surface_condition	состояние поверхности
Draughting	D	D1	explicit_draughting	явное черчение
		D2	associative_annotation	ассоциированные чертежные элементы
External_refernce	E	E1	external_reference_mechanism	механизм внешних ссылок
Form_feature	FF	FF1	user_defined_feature	определенный пользователем конструктивно-технологический элемент (КТЭ)
		FF2	included_feature	включенный (предопределенный) конструктивно-технологический элемент (КТЭ)
		FF3	generative_featured_shape	форма, генерируемая конструктивно-технологическим элементом
Geometry	G	G1	wireframe_model_2d	2-мерная проволочная модель
		G2	wireframe_model_3d	3-мерная проволочная модель
		G3	connected_surface_model	связная поверхностная модель
		G4	faceted_b_rep_model	плоскогранная граничная модель
		G5	b_rep_model	граничная модель
		G6	compound_model	составная модель
		G7	csg_model	конструктивная твердотельная геометрия
		G8	geometrically_bounded_surface_model	геометрически ограниченная поверхностная модель
Kinematics	K	K1	kinematics	кинематика
Measured_data	M D	MD 1	measured_data	данные замеров
Property	PR	PR1	item_property	свойства объекта
Presentation	P	P1	geometric_presentation	визуальное геометрическое представление
		P2	annotated_presentation	представление чертежных элементов
		P3	shaded_presentation	полутонное визуальное представление
Product_structure	S	S1	product_management_data	данные управления проектом
		S2	element_structure	структура элемента
		S3	item_defined_structure	структура определения объекта
		S4	effectivity	применяемость
		S5	work_management	управление работами
		S6	classification	классификация
		S7	specification_control	управление описанием

		S8	process_plan	технологический процесс обработки
Tolerance	T	T1	dimension_tolerance	допуски размеров
		T2	geometric_tolerance	геометрические допуски

В Протоколе 31 функциональный блок, 18 классов соответствия. Следует иметь ввиду то, что большинство конверторов поддерживает лишь часть сущностей, входящих в Протокол (классы соответствия CC1 и CC2). К примеру, конверторы MDT, конвертор Solid Works обеспечивают передачу только геометрических моделей в формате AP214, конверторы других систем способны передавать также данные воспринимаемого представления.

Протокол для систем CAE (АСНИ) «Анализ и конструкция композитных и металлических конструкций» AP209.

Данный Протокол содержит средства, позволяющие описывать модели для инженерных расчетов и полученные в результате расчетов данные. Согласно отчету PDES за 2001 год, в ближайшее время конверторы в этот Протокол появятся у CATIA и Pro/Engineer.

Прочностные расчеты по МКЭ и композитные материалы – это предметная область, охватываемая Протоколом AP209. Применение AP209 обеспечивает следующие возможности:

Взаимодействие процессов конструирования и расчета: Компании, использующие различающиеся САПР и системы расчета по МКЭ, получают возможность обмениваться техническими данными о конструкции и результатами расчетов посредством использования файлов стандартного формата. Присутствие в стандарте данных о конфигурации изделия обеспечивает то, что выполняемая различными подразделениями деятельность по конструированию и расчету согласована по версиям изделия.

Имеется возможность осуществлять раздельную, но согласованную между собой по версиям изделия работу с идеальной и конечноэлементной моделями.

Архивирование конструкторских и расчетных данных: Компания получает возможность архивировать связанные с определенной конфигурацией изделия данные САПР и МКЭ в стандартном формате. Это обеспечивает возможность повторного использования данных в будущем, независимо от того, какие произойдут изменения в составе программного обеспечения. Особенно это важно когда МКЭ используется для сертификации изделия, т.к. в этом случае для расследования проблем, возникающих при эксплуатации изделия, могут потребоваться дополнительные расчеты.

Разработка интегрированных систем: Компания получает возможность разработки интегрированной системы, соединяющей в себе основанные на использовании стандартных интерфейсов конструирование и расчеты. Такая интегрированная система может относиться к различным расчетным дисциплинам, таким как прочностные, тепловые и гидродинамические и может использоваться для оптимизации конструкции. Применение стандартного интерфейса обеспечит то, что система не будет привязана к конкретным производителям программного обеспечения.

AP 209 охватывает:

Данные о конечных элементах: Сюда включаются модели, описания расчетных случаев и случаев нагружения и результаты расчета. Модель может быть задана столь подробно, насколько это требуется – если необходимо, вплоть до уровня функций, формы элемента, точек дискретизации и правил интеграции. В изначальную область входит анализ постоянных и собственных частот.

Данные о конфигурации: Версия конечноэлементной модели связана с версией изделия. Это обеспечивает то, что в PDM (Product Data Management – Система управления данными об изделии или Система управления проектом) правильные данные о конечных элементах связаны с правильными данными об изделии.

Геометрию изделия: Могут быть записаны как конструкторская геометрия, так и идеализированная геометрия. Узлы, конечные элементы и грани или ребра конечных элементов могут быть явно связаны с аспектами геометрии изделия.

Для кривой, поверхности или объема геометрической модели можно задать некоторые свойства элементов, нагрузки и граничные условия.

Слои композитного материала: Могут быть подробно описаны слои детали из композитного материала. Могут быть представлены данные о форме, последовательности укладки, свойствах и ориентации отдельных слоев.

В процессе анализа многократно осуществляется обмен информацией:

– информация о форме из САД-системы передается в конечноэлементный препроцессор;

- конечноэлементная расчетная модель передается в систему расчета по МКЭ;

- результаты расчета передаются в постпроцессор. Постпроцессор при этом может быть предназначен только для визуализации, но может выполнять также и подробное конструирование и действия по оптимизации конструкции или проверке.

Каждый из этих интерфейсов может быть стандартизован с помощью AP 209.

Другие разрабатываемые концептуальные схемы данных STEP для целей технического анализа:

thermal and radiative (тепло и излучение): ESA (European Space Agency, Европейское Космическое Агентство) разрабатывает STEP-TAS (Thermal Analysis for Spacecraft, Тепловой Анализ Космических Аппаратов), который принят также NASA. Формально это не стандарт ISO, но он полностью совместим со STEP.

fluid dynamics (гидродинамика): Это существующий стандарт CFD (Computational Fluid Dynamics, вычислительная гидродинамика) - CGNS (см <http://www.cgns.org>). Этот стандарт переработан как STEP AP 237. Пользой от этого будет:

возможность задания границ жидкой среды относительно конструкции; и

возможность передачи результатов CFD для анализа конструкции и других видов анализа.

meshless field representation (бессеточное представление полей): US Navy (ВМФ США), совместно с Boeing, разработал библиотеку программ DTNURBS для представления объемных полей и поверхностей с помощью n-мерных B-сплайнов высокого порядка (см <http://ocean.dt.navy.mil/dtnurbs/>). Это соответствует 50 тому STEP. Это основа для обмена данными между приложениями бессеточного анализа.

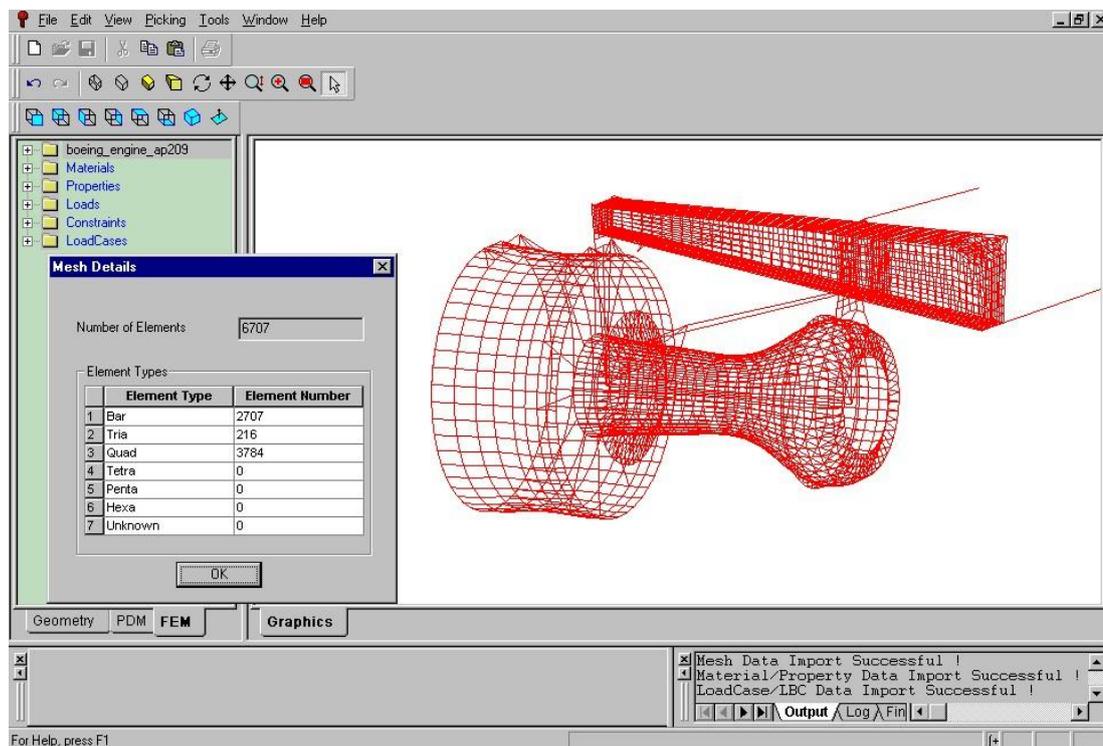


Рис. 64. Предоставленная Boeing Company полная модель двигателя. Изображение получено непосредственно из файла STEP



Рис. 65. Модель слоев, предоставленная Lockheed Martin Aeronautics. Изображение получено непосредственно из файла STEP

Прикладной Протокол **AP210** был определен как стандартизированный, поддающийся компьютерному описанию метод для представления проектов и связи между ними:

- Разводка платы и конструкторское оформление,
- Модули,
- Блоки печатного монтажа (PWAs), и
- Платы печатного монтажа (PWBs).

Это достижение было достигнуто благодаря скоординированным действиям компании Fortune 500, включающее множество компаний, а так же некоторые правительственные структуры. Организация включает в себя Rockwell Collins Inc., The Boeing Company, Delphi-Delco Electronic Systems, IBM Corporation, Naval Supply Systems Command RAMP Program Office, и Национальный Институт Стандартов и Технологии. Дополнительно, большое число контрактов других PDES, корпораций и правительства внесли вклад в развитие, испытание, и проверку достоверности. Эта большая основа помогла гарантировать успешное завершение разработки стандарта и поддержку и потребности AP210 в будущем.

Прикладной протокол AP210 обеспечивает представление данных в контексте таким образом, что данные сгенерированные одним приложением поддавались компьютерной интерпретации другим приложением. Электронный блок, разводка плат и общий дизайн были определены в соответствии со стандартными промышленными требованиями в течение всего процесса разработки AP. AP210 позволяет обмен и совместное использование данных поддающихся компьютерной интерпретации, а так же требований наложен-

ные на изделие и данные заказчиками, поставщиками, партнерами по бизнесу.

Основным разработчиком как самого протокола, так и подавляющего большинства препроцессоров и постпроцессоров является германская фирма LKSoft. Ее хозяин и руководитель Лотар Кляйн считается крупнейшим специалистом в мире по стандартизации данных в электронике.

Определенно, AP210 обеспечивает использование структур данных для обмена электромеханическими проектами, требованиями, принципиальными схемами, расположением отверстий, и сопоставление конфигурации с конфигурацией 2-х или 3-х мерных моделей образца. Подобный протокол был разработан, чтобы представить, определение требований, анализ и производственные стадии электромеханических изделий. Последующий AP (в настоящее время уже выпущенный или находящийся на стадии разработки) будет содержать в себе данные в формате AP 210 обо всем жизненном цикле изделия.

Некоторые Концепции Высокого уровня

AP210 содержит более 900 индивидуальных понятий и 95 функциональных областей высокого уровня. AP210 поддерживает форматы данных, которые представлены в формате Gerber, IDF, GenCAM, ODB ++ и EDIF. Возможности AP210 настолько широки и масштабны, что попытки описать протокол во всех деталях были бы не слишком полными для этого документа. Таким образом, мы попытаемся объяснить только основные принципы функционирования. Поддерживаемые типы изделий, этапы жизненного цикла, создаваемые и используемые технические дисциплины, разводка проводников электронного блока и компоновка определяющие контекст AP210. В следующих абзацах будут подчеркнуты некоторые из аспектов, на которых базируется AP210.

Каткая характеристика.

AP210 поддерживает методологию системного проектирования в течение всего процесса разработки проекта. Требования можно получить из спецификации, а они в свою очередь могут быть применены к почти любой области проекта, с целью повторного использования проекта для управления интерфейсом. Некоторые проекты служат требованиями для других областей проекта (например, соединения в схеме). AP 210 поддерживает выполнение разводки платы и конструкторской документации по правилам стандартного управления.

Возможности структуры протокола

AP 210 поддерживает несколько иерархических моделей, включая: функциональную, физическую, модель требований, модель правил, модель электронного блока, модель печатной платы (PCB), модель материалов для сборки. Функциональная модель поддерживает: свернутую иерархическую таблицу соединений (в схеме), а на плоскости и таблицу соединений с ограниченной маршрутизацией.

Таблица соединений способна поддерживать: обобщенные многоуровневые или много дисциплинарные модели. Связывающую среду (например, печатную плату) – отдельно определяет само изделие. Сигнал зависящий от характеристик функционального модуля определяется типом сигнала, учитывающим вид сигнала (например, ATLAS, Automated Trade Locator Assistance Network), и поддерживающим ссылки на внешние определители. Физические модели поддерживают как одно- так и многокристальные электронные блоки, а так же модули с большим количеством плат.

Технологические Возможности. AP 210 поддерживает универсальные модули и коммутационные модели данных (например, сквозное отверстие, SMT (технология поверхностного монтажа), линии малой ширины, MCMS (многокристальный модуль), литые части, одиночные чипы). Конструкции для возможности серийного производства поддерживаются протоколом (то есть, явно наблюдаются зависимости между необходимой формой блока и требованиями наложенными на разводку печатной платы). Протокол так же обеспечивает определение материала и композиции, коммутации, возможности процесса изготовления, предельные характеристики упаковочного оборудования, и характеристики основания печатной платы (то есть, модели паяного соединения). Конструкторская оптимизация для определенной цепи питания сопоставляется с моделями: основания печатной платы, электронного блока, и элементов используемых при сборке модели. Правила конструкторского исполнения изделия могут быть определены и введены в действие.

Возможности физической модели. AP 210 поддерживает установку компонентов на печатной плате и точную топологическую модель расположения элементов. Протоколом поддерживается функциональное определение электрических, оптических и тепловых характеристик элементов. Поддерживаются системы с сосредоточенными и распределенными параметрами. Основная концепция протокола заключается в девизе "Определи один раз, и все время используй в дальнейшем" и он применим ко всем компоновочным моделям. Конструкторский смысл сохраняется для многослойных печатных плат. Данные о не смонтированной печатной плате легко экстрагируются. Описание материала, по желанию, может быть как простым так и комплексным. Протокол позволяет использовать одни и те же коммутации в различных электронных блоках, в свою очередь коммутации разного типа могут присутствовать в одном и том же электронном блоке и различные уровни электронного блока могут существовать в одной модели. Электронный блок и модель схемы соединений используются в конструкции готового изделия и при разводке плат. Размеры STEP и расстояние между повторяющимися элементами, образцы для испытаний и нормы так же поддерживаются протоколом.

AP 210 способен обмениваться файлами с AP 203 (конфигурация управляемых трехмерных конструкций механических деталей и электронных блоков).

Управление проектированием. AP 210 обеспечивает методологию управления проектированием реальной конструкции, которая поддерживает

обмен данных между приложениями (например, между CAD/ CAM и координатно-измерительной машиной) без помощи человека. Эта способность кладет основу для существенного уменьшения времени производственного цикла и используется как основа для определения размеров и установления допусков геометрического объекта (GD&T GD&T сокр. от Geometric Dimensioning and Tolerancing – Геометрическое образмеривание и задание допусков) в CAD/CAM системах. Методология полностью основана на стандартах ASME (ASME сокр. от American Society of Mechanical Engineers Американское общество инженеров-механиков) Y14. 5M и ISO 1101 для данных GD&T. AP 210 совместим и поддерживает последние разработки ASME, чтобы определить модель базирующуюся на GD&T стандартах: Y14.41 (Цифровое Моделирование).

Возможность протокола по использованию библиотечных компонентов. AP 210 поддерживает информационные структуры, часто находящиеся в «библиотеке», такие как: тип параметра, названия порта, тип порта, модель блока. AP 210 обеспечивает хорошее соединение между имитационными моделями и данными определяющими изделие. AP 210 - единственный стандарт, который обеспечивает поддающуюся проверке корреляцию между двумерной моделью и трехмерным описанием изделия.

Геометрические возможности. AP 210 совместим с IPC, EDIF, Gerber, GDSII и векторными фото плоттерами. Геометрические модели AP 210 основаны на AP 203 и других прикладных протоколах. AP 210 поддерживает трехмерную геометрию. Открытая модель оболочки поддерживает, формочный коммутационный дизайн и технологию с топологически управляемой геометрией.

STEP был разработан промышленностью как стандарт обмена данными об изделии. Основное преимущество протокола это - способность хранить информацию, связывающую полный спектр данных об изделии, однако целью STEP не является замена существующих электрических стандартов. Целью STEP является обеспечение средой, способной хранить, не только данные о электрических компонентах и изделиях, но также и других основных областях жизненного цикла изделия. Теперь Mentor Graphics работает с International TechneGroup Inc., с целью разработки двустороннего транслятора AP 210/Mento. Другие продавцы CAD систем так же думают о будущем сотрудничестве в данной области. Фактически, в следующие несколько месяцев недорогие свободно распространяемые трансляторы STEP будут доступны для Gerber, IDF, GenCAM и других наиболее популярных стандартов. Гипотетически можно предположить, что довольно большое число продавцов производственного оборудования, измерительного инструмента, и т.д. адаптирует свои инструментальные средства для чтения форматированных данных STEP в ближайшем будущем. Возможно что некоторые из этих инструментальных средств, по социальным, экономичным, или политическим причинам, никогда не смогут развиваться до полного взаимодействия с STEP. Однако последняя причина не может служить оправданием в пользу прекращения использования AP 210 или STEP вообще.

Фирмой STEP Tools, Inc. (США) ведутся работы по разработке протокола STEP NC (AP238), позволяющего представлять данные для оборудования с ЧПУ, в первую очередь – для обработки резанием, но в перспективе планируется расширение этих подходов и на другие виды технологических процессов. Цель внедрения AP238 – добиться независимости от постпроцессоров ЧПУ, обеспечить долгосрочное хранение данных для оборудования с ЧПУ и повысить надежность и гибкость данных. Это достигается за счет того, что в одной модели STEP содержится как информация о самой изготавливаемой детали, так и описание технологического процесса со всеми необходимыми режимами и траектория режущего инструмента. Это позволяет корректировать траекторию при обнаружении ошибки (в этом случае траектория инструмента может быть сопоставлена с формой детали и при необходимости исправлена) и усовершенствовать технологический процесс, ориентируясь на особенности конкретного оборудования.

Известно, что при проектировании технологических процессов механической обработки резанием наряду с информацией о геометрическом образе будущей детали, необходима информация о требованиях к точности, шероховатости, физико-химических свойствах детали и отдельных поверхностей, а также параметры заготовки и идентификации. Вся эта и некоторая другая информация составляет описание технологического образа детали в виде технологической модели, которая может быть использована при автоматизированном проектировании технологических процессов.

Таким образом, технологическая модель детали связывает конструкторскую и технологическую систему проектирования.

Вместе с тем, существующие в настоящее время стандартные форматы обмена данных о конструкции (dxf, Igs, sat, и др.) содержат, в основном, геометрическую информацию, которой недостаточно для решения задач технологического проектирования.

В связи с этим, чрезвычайно важным является не только формирование информации о технологическом облике изделия на этапе проектирования, но и поиск доступных форм ее машинного представления.

Но это одна сторона проблемы. Другая сторона заключается в том, что существующие системы автоматизированного проектирования технологии, построенные на основе библиотек типовых решений, не могут использовать данные электронной модели изделия, так как не имеют алгоритма синтеза технологического процесса на основе описания свойств отдельных поверхностей детали и их взаимосвязей. Создание же генерирующих систем, направленных на синтез структуры и содержания технологических процессов на основе свойств конструкции, в настоящее время практически не ведется из-за сложности и большой трудоемкости таких разработок.

Чтобы обойти это ограничение, хотя бы на отдельные группы конструктивно и технологически подобных деталей, в настоящее время вводится в практику разработка и использование единых конструкторско-технологических модулей, представленных в виде параметрических моделей отдельных конструктивов. Эти конструктивы служат не только в качестве

«кирпичей» для конструирования, но и информационной базы в виде технологических модулей для систем технологического проектирования.

Создание описания технологической модели детали (изделия) предполагает определение и классификацию составляющих ее объектов, определение классов параметров (свойств) и отношений, характеризующих эти объекты в отдельности и в совокупности.

Важной проблемой построения технологической модели на этапе конструирования является проблема идентификации. Каждый объект этой модели в процессе создания получает имя, которое необходимо сохранить на протяжении всего процесса проектирования и включить в состав описания модели детали. Все свойства и отношения, характеризующие данный объект должны быть связаны с его именем.

В большинстве существующих систем проектирования информация об этой идентификации не выводится в составе выходных данных (т.е. в описание результатов проектирования), поэтому в настоящее время идентификацию поверхностей, большинство их свойств и отношений, необходимых для технологического проектирования, приходится вводить вручную.

Методом решения данной проблемы является унификация конструкторских и технологических объектов путем использования в качестве стандартных конструктивов при проектировании конструкции типовых технологически подобных поверхностей или их сочетаний. К таким конструктивам относятся, например, отверстия, фаски, торцы, лыски, канавки, пазы, бобышки и многие другие.

Каждый из этих конструктивов может быть представлен в виде параметрической модели, включающей описание свойств конструктива и его связей с другими конструктивами детали. Эти связи описываются с помощью отношений, характеризующих положение одних конструктивов относительно других в пространстве детали.

Необходимость формирования и доступного представления данных для решения технологических задач привела к реализации в «тяжелых» системах типа "Unigraphics" и "Pro/Engineers" механизма User-Defined Features (UDF) – определяемых пользователем конструктивно-технологических элементов. Учитывая наличие в этих системах функции параметризации, этот механизм может быть использован для подготовки данных о технологической модели детали на этапе конструирования.

На схеме представлена существующая модель связей конструктивных и технологических элементов с помощью формата протоколов STEP 214, 224, 238.

Вместе с тем, эти данные носят ограниченный характер и достаточны только для решения частных технологических задач, например, построения схем обработки на станках с ЧПУ типовых конструктивных элементов или планов обработки отдельных поверхностей. Для решения задач по формированию планов обработки детали в целом требуется полное описание технологической модели детали.

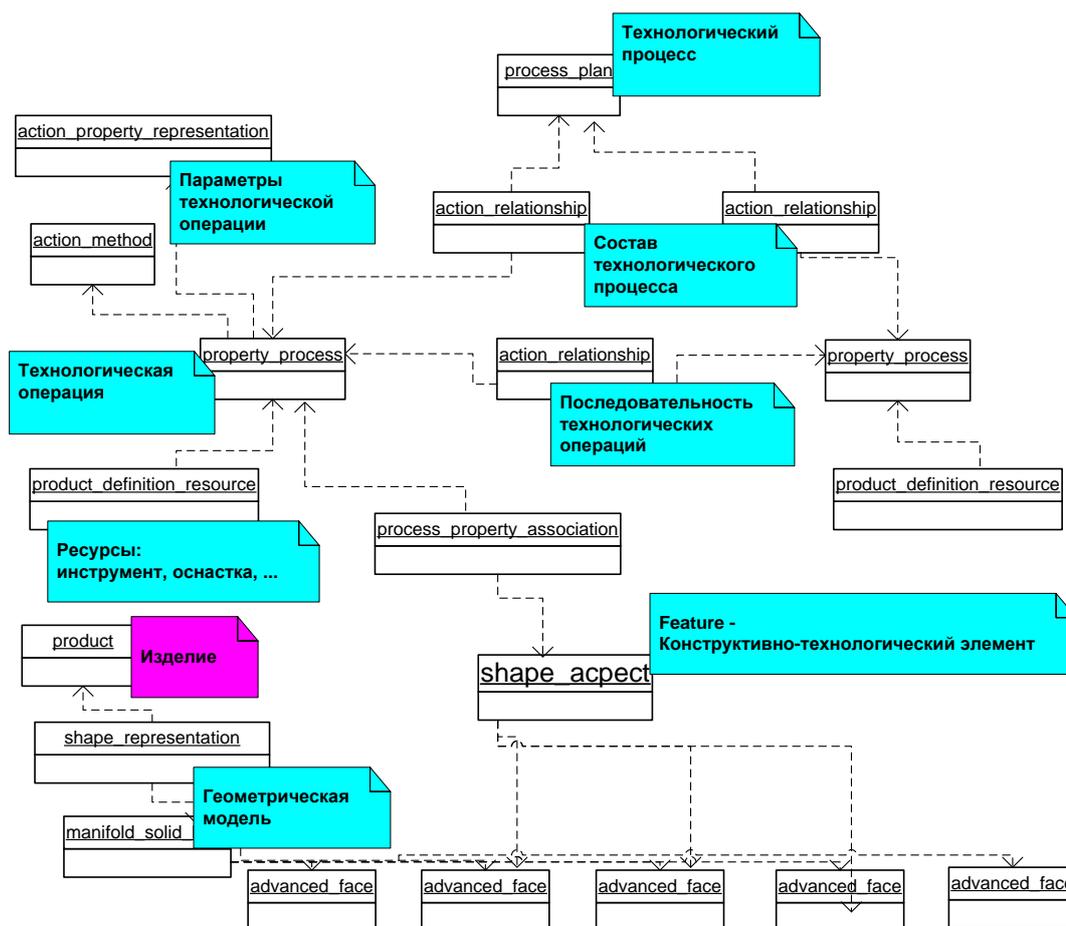


Рис. 66. Связь КТЭ с ТП в технологических протоколах механообработки

Из опыта создания систем автоматизированного проектирования технологии технологическое описание детали должно включать следующие виды информации:

- свойства отдельных поверхностей детали (вид поверхности, требования к отклонениям формы, шероховатость, твердость, собственные размеры и допуски и др.);
- ориентацию поверхностей относительно общей системы координат;
- положение поверхностей относительно друг друга (закрытость по направлениям, примыкание);
- положение металла относительно поверхности;
- координирующие размеры и точность относительного расположения поверхностей (допуски на отклонения размеров, несоосность, неперпендикулярность, биение и др.);
- материал, вид заготовки, технические требования к термической и химико-термической обработке, покрытию и др.

Вся эта информация с помощью механизмов типа UDF может быть соотнесена к отдельным конструктивам в виде их параметрического описания. Однако проблема описания технологической модели детали в целом остается при этом открытой.

Для решения этой проблемы и, одновременно обеспечения возможности для использования методов проектирования технологии на основе библиотек вариантных технологических процессов, необходимо построить такую иерархическую модель структуры детали, в которой каждый уровень содержал бы класс параметрически описанных структурных элементов, которому можно поставить в соответствие вариантный процесс технологии его обработки. Такая иерархическая модель конструкции детали может содержать всю необходимую информацию для проектирования технологии ее изготовления.

В этом случае библиотека вариантных технологических процессов также должна быть представлена в виде иерархической модели, где на каждом уровне параметрически описываются варианты содержания структурных элементов технологического процесса определенного уровня детализации: маршрут, операция, переход, проход. При этом применимость того или иного структурного технологического элемента может определяться как наличием соответствующего класса конструктивных элементов, так и значением его параметров.

При таком подходе с одной стороны реализуется известный метод проектирования технологии на основе конструктивно-технологических модулей, а с другой – обеспечивается возможность автоматизировать процесс получения структуры технологического процесса на всю деталь в целом в соответствии с иерархической моделью структуры, заданной при конструировании детали.

Сформированный план обработки, наряду с геометрической информацией, полученной на этапе конструирования, позволяет соединить этапы конструирования и изготовления. При этом на первый план выходят задачи представления и передачи геометрической и технологической информации в системы формирования управляющих программ и управления станками с ЧПУ.

Для реализации интеграции систем необходимы инструменты унификации представления данных. В STEP стандартизован ряд онтологий (Прикладных Протоколов), в том числе для представления технологических данных. Наибольший интерес представляет Протокол 238, позволяющий описывать, наряду с геометрией механически обрабатываемой детали, технологический процесс, используемые инструменты и режимы обработки.

Данной проблемой активно занимается фирма STEP Tools, Inc. (США). В области интеграции систем проектирования и изготовления механически обрабатываемых деталей в проекте STEP NC, выполняемом фирмой, выделяются следующие задачи, которые предлагается решать за счет использования Прикладного Протокола 238 STEP:

- передачи элементов знания о детали – т.н. КТЭ (конструктивно-технологических элементов), в англоязычной литературе это соответствует понятию “feature”. Хотя практически все современные САПР создают и сохраняют информацию о КТЭ, экспорт таких данных в унифицированном формате реализован только в одной САПР – Pro/Engineer (в экспериментальном конвертере, разработанном корпорацией SDRC);

- проектирование технологических процессов изготовления детали на основе данных о свойствах КТЭ и взаимосвязях;

- подготовки управляющих программ для оборудования с ЧПУ.

На рис. представлены схемы передачи данных в цепи проектирование-изготовление в существующем и рассматриваемом вариантах.

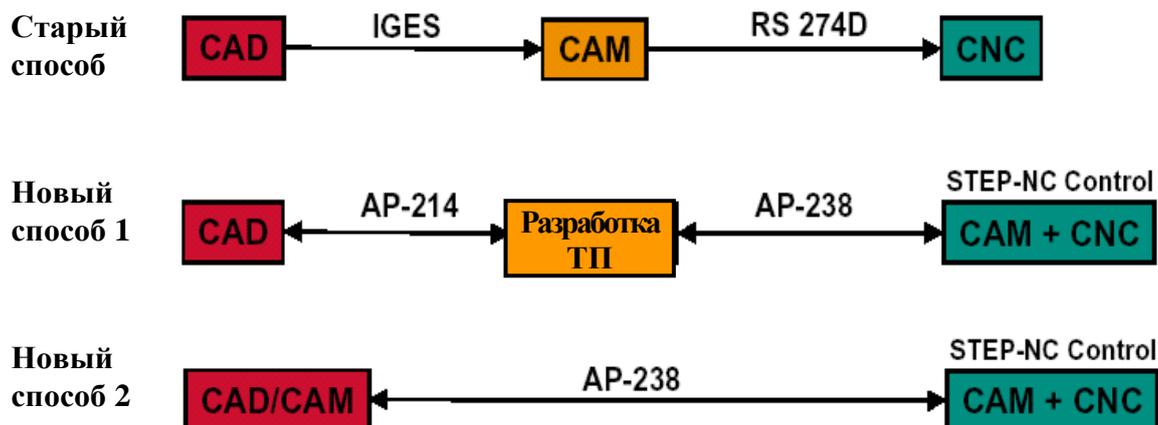


Рис. 67. Варианты передачи данных в цепи проектирование-изготовление

В результате выполнения проекта STEP NC на смену существующему оборудованию придет оборудование, способное воспринимать модели деталей, представленные с помощью AP238. Протоколы STEP взаимно согласованы и потому частично перекрываются. AP238 в настоящее время находится в стадии разработки и согласования. Часть данных AP238, включая геометрическую модель изделия, описание КТЭ, а также состав и некоторые параметры ТП могут быть представлены протоколами AP214 и AP224, которые уже сейчас утверждены как Международные стандарты.

Контрольные вопросы

19. Какие основные предметные области охватываются Прикладными протоколами STEP.
20. Какие преимущества обеспечит применение Прикладного протокола 238 (STEP NC).

10. Заключение

Кроме стандарта STEP и группы основанных на нем стандартов язык EXPRESS используется в ряде других стандартов и проектов:

Стандарты, разрабатываемые TC184/SC4, полностью совместимые со STEP и использующие те же методы описания, методы реализации и информационные ресурсы. Это – стандарты группы STEP, к которым относятся:

ISO 13584 PLIB – стандарт на представление электронных каталогов изделий;

ISO 15531 MANDATE – стандарт на представление производственных данных;

ISO 15925 OIL&GAS – стандарт на представление данных, относящихся ко всем этапам жизненного цикла нефтегазового оборудования.

Существуют и другие стандарты, полностью или частично совместимые со STEP в отношении методов описания и реализации:

POSC – Petrotechnical Open Software Corporation – формат представления геологических и геофизических данных;

DICE – DARPA Initiative In Concurrent Engineering;

CFI – CAD Framework Initiative;

Работы Комитета TC211 по стандартизации средств представления данных в геоинформационных системах;

Работы Подкомитета TC184 SC5 по разработке комплекса стандартов моделирования предприятий;

IFC – Industrial Foundation Classes

ISO 21308 – Согласованный с AP214 STEP международный стандарт на термины и определения для дорожных транспортных средств.

ISO 14649 – представление данных для оборудования с ЧПУ. Этот стандарт использовался при разработке протокола AP238 STEP.

ISO 14048 – данные, описывающие влияние изделия на окружающую среду на различных этапах жизненного цикла изделия.

CIS/2 – данные, описывающие стальные конструкции;

IEEE SCC20 – стандарт на представление данных диагностики сложных систем в области электроники.

11. Библиография

1. ISO TC184/SC4 N1167 2001-08-01: ISO TC184/SC4 SC4 Industrial Data Framework. (Размещено в Интернет по адресу: http://www.tc184-s4.org/SC4_Open/SC4_and_Working_Groups/SC4_N-DOCS/1000-1249/).
2. ISO TC 184/SC4 N535:1998(E) 1998-12-18: Guidelines for the development and approval of STEP application protocols. (Размещено в Интернет по адресу: http://www.tc184-s4.org/SC4_Open/SC4_and_Working_Groups/SC4_N-DOCS/500-999/).
3. NATO CALS Handbook: Version 2, June 2000. (Размещено в Интернет по адресу: http://www3.dcnicn.com/ncmb/nch_iune-2000/handbook.asp).
4. **Шильников П.С.** Путь НТЦ АПМ в Единое информационное пространство//САПР и графика. – 2005. – №2. –С. 56 – 60.
5. **Казарновский А. Ю., Шильников П. С.** Методика применения формата STEP для отображения документов//Международная научно-техническая конференция Информационные технологии в науке, образовании и промышленности: Сборник трудов. – Архангельск: Соломбальская типография, 2005. – С. 103 – 110.
6. **Шильников П.С.** Работа с данными в формате ISO 10303 STEP: Методические указания по лабораторной работе.– М.,2004. (Размещено в Интернет по адресу: http://www.engineer.bmstu.ru/res/rk9/shilnikov/step_2004_06_02/)
7. Overview and fundamental principles // ISO 10303–1: Industrial automation systems and integration: Product data representation and exchange, 1994. – Part 1. – 24p.

8. The EXPRESS language Reference manual //ISO 10303: Industrial automation system and integration: Product data representation and exchange: Description methods, 1994.– part 11. – 270p.
9. Standard data access interface // ISO 10303: Industrial automation system and integration: Product data representation and exchange: Implementation methods, 1998. – Part 22. – 217p.
- 10.ГОСТ Р ИСО 10303-41-99 Системы автоматизации производства и их интеграции. Представление данных об изделии и обмен эти-ми данными. Часть 41 Интегрированные обобщенные ресурсы. Основы описания и поддержки изделий.
- 11.Geometric and topological representation // ISO 10303: Industrial au- tomation system and integration: Product data representation and ex- change: Integrated generic resources, 1994. – Part 42. – 241p.
- 12.ГОСТ Р ИСО 10303-43-2002 “Системы автоматизации производ-ства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 43. Интегрированные обобщенные ресур-сы. Представление структур”
- 13.ГОСТ Р ИСО 10303-44-2002 “Системы автоматизации производ-ства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 44. Интегрированные обобщенные ресур-сы. Конфигурация структуры изделия”
- 14.ГОСТ Р ИСО 10303-45-2000 Системы автоматизации производ-ства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 45. Интегрированные обобщенные ресур-сы. Материалы
- 15.ГОСТ Р ИСО 10303-46-2002 “Системы автоматизации производ-ства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 46. Интегрированные обобщенные ресур-сы. Визуальное представление

- 16.ГОСТ Р ИСО 10303-49-2002 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 49. Интегрированные обобщенные ресурсы. Структура и свойства процесса
- 17.**Прут М.В.** Некоторые аспекты реализации обмена данными между T-Flex CAD и внешними системами с использованием независимых форматов //3-я Международная конференция CAD/CAM/CAE/PDM-2003: Сборник трудов. – М.,2003. (Размещено в Интернет по адресу: <http://lab18.ipu.rssi.ru/>)
18. **Судов Е.В.** Интегрированная информационная поддержка жизненного цикла машиностроительной продукции. Принципы. Технологии. Методы. Модели. – М.: ООО Издательский дом «МВМ», 2003. – 264с.
- 19.Configuration controlled design // ISO 10303: Industrial automation system and integration: Product data representation and exchange: Application protocols, 1994. – Part 203. – 528p.
- 20.**Самсонов О.С., Шильников П.С.** Представление технологических данных с применением CALS-стандартов//Вестник МГТУ им. Н. Э. Баумана. Приборостроение. – 2002. – №1. – С. 59– 64.
- 21.**Шильников П.С.** Средства поддержки CALS-технологий (технологий непрерывной компьютерной поддержки полного Жизненного Цикла Изделия)//Юбилейная научно-практическая конференция АНТОК СНГ:Тезисы докладов. –М.,2001. – С.15– 18.
- 22.**Сычев А.П., Чураев А.А., Шильников П.С.** Разработка программного инструментального комплекса для работы с данными, соответствующими CALS-стандарту ISO 10303 STEP//2-я Международная конференция CAD/CAM/CAE/PDM-2002: Сборник трудов. –М.,2002. –Т.1. –С.30–32.

23. **Норенков И.П., Кузьмик П.К.** Информационная поддержка наукоемких изделий. CALS-технологии.- М.: Изд-во МГТУ им. Н.Э.Баумана, 2002.-320 с.
24. **Овсянников М.В., Шильников П.С.** «Глава семьи информационных CALS-стандартов ISO 10303 STEP // САПР и Графика. 1997, №11, с.76-82.