Р 50.1.030—2001

РЕКОМЕНДАЦИИ ПО СТАНДАРТИЗАЦИИ

Информационные технологии поддержки жизненного цикла продукции

ИНТЕРАКТИВНЫЕ ЭЛЕКТРОННЫЕ ТЕХНИЧЕСКИЕ РУКОВОДСТВА

Требования к логической структуре базы данных

Издание официальное

ГОССТАНДАРТ РОССИИ
Москва

Предисловие

1 РАЗРАБОТАНЫ Научно-исследовательским центром CALS-технологий «Прикладная логистика» при участии Всероссийского научно-исследовательского института стандартизации (ВНИИстандарт)

ВНЕСЕНЫ Техническим комитетом по стандартизации ТК 431 «CALS-технологии»

2 ПРИНЯТЫ И ВВЕДЕНЫ В ДЕЙСТВИЕ Постановлением Госстандарта России от 2 июля 2001 г. № 256-ст

3 ВВЕДЕНЫ ВПЕРВЫЕ

Содержание

РЕКОМЕНДАЦИИ ПО СТАНДАРТИЗАЦИИ

Информационные технологии поддержки жизненного цикла продукции

ИНТЕРАКТИВНЫЕ ЭЛЕКТРОННЫЕ ТЕХНИЧЕСКИЕ РУКОВОДСТВА

Требования к логической структуре базы данных

Continuous acquisition and life-cycle support. Interactive electronic technical manuals.
Requirements for data base logical structure

Дата введения 2002—07—01

1 Область применения

Настоящие рекомендации по стандартизации определяют требования к логической структуре баз данных, используемых для разработки и сопровождения интерактивных электронных технических руководств (ИЭТР) на промышленные изделия, а также баз данных, входящих в состав ИЭТР.

2 Нормативные ссылки

В настоящих рекомендациях использованы ссылки на следующие стандарты:

ГОСТ Р ИСО 10303-21—99 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 21. Методы реализации. Кодирование открытым текстом структуры обмена

ГОСТ Р ИСО 10303-22—2001 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 22. Методы реализации. Стандартный интерфейс доступа к данным

3 Определения

В настоящих рекомендациях использованы следующие термины с соответствующими определениями:

3.1 **база данных:** Организованное управляемое хранилище технической информации.

3.2 **информационный объект:** Смысловая и структурная единица технической информации.

3.3 **разметка текста:** Внесение знаков разметки (тегов) в данные, в соответствии с ИСО 8879 [1], с целью выделения и обозначения отдельных информационных объектов.

3.4 **стиль:** Перечень связанных с информационным объектом правил, регламентирующих отображение данных на устройстве вывода (шрифт, размер, цвет и т. д.).

3.5 **электронная система отображения:** Комплекс программно-технических средств для воспроизведения данных, содержащихся в интерактивном электронном техническом руководстве.

4 Сокращения

В настоящих рекомендациях используют следующие сокращения:

CALS — концепция и идеология информационной поддержки жизненного цикла продукции на всех его стадиях, основанная на использовании единой информационной среды (Continuous Acquisition and Life-cycle Support).

Издание официальное

DTD — файл, содержащий описание информационных объектов базы данных и их атрибутов, а также правил, которым должна удовлетворять логическая структура базы данных (Document Type Definition).

FOSI — спецификация стиля отображения данных на устройстве вывода (Formatted Output Specification Instance).

PDM — управление данными об изделии (Product Data Management).

SGML — язык разметки (Standard Generalized Markup Language).

БД — база данных.

ИЭТР — интерактивное электронное техническое руководство.

ЭСО — электронная система отображения.

5 Общие положения

5.1 Интерактивные электронные технические руководства

Интерактивное электронное техническое руководство (ИЭТР) представляет собой структурированный комплекс взаимосвязанных технических данных, требуемых на этапах эксплуатации и ремонта изделия. Использование ИЭТР позволяет предоставить в интерактивном режиме справочную и описательную информацию об эксплуатационных и ремонтных процедурах, относящихся к конкретному изделию, непосредственно во время проведения этих процедур.

ИЭТР включает в себя БД и ЭСО, предназначенную для визуализации данных и обеспечения интерактивного взаимодействия с пользователем.

БД ИЭТР имеет структуру, позволяющую пользователю быстро получить доступ к нужной информации. БД ИЭТР может содержать текстовую и графическую информацию, а также данные в мультимедийной форме (аудио- и видеоданные).

ЭСО обеспечивает унифицированный для всех ИЭТР способ взаимодействия с пользователем и технику представления информации.

ИЭТР предназначены для решения следующих задач:

- обеспечения пользователя справочным материалом об устройстве и принципах работы изделия;
- обучения пользователя правилам эксплуатации, обслуживания и ремонта изделия;
- обеспечения пользователя справочными материалами, необходимыми для эксплуатации изделия, выполнения регламентных работ и ремонта изделия;
- обеспечения пользователя информацией о технологии выполнения операций с изделием, потребности в необходимых инструментах и материалах, количестве и квалификации персонала;
- диагностики оборудования и поиска неисправностей;
- подготовки и реализации автоматизированного заказа материалов и запасных частей;
- планирования и учета проведения регламентных работ;
- обмена данными между потребителем и поставщиком.

5.2 Классификация интерактивных электронных технических руководств

По функциональным возможностям ИЭТР разделяются на четыре класса.

Класс 1 — индексированные цифровые изображения страниц

ИЭТР данного класса представляет собой набор изображений, полученных сканированием страниц документации. Страницы индексированы в соответствии с содержанием, списком иллюстраций, списком таблиц и т. п. Индексация позволяет отобразить растровое представление необходимого раздела документации сразу после его выбора в содержании. Данный тип ИЭТР сохраняет ориентированность страниц и может быть выведен на печать без предварительной обработки.

Класс 2 — линейно-структурированные электронные документы

ИЭТР данного класса представляет собой совокупность текстов в формате SGML. Оглавление ИЭТР содержит ссылки на соответствующие разделы технического руководства. ИЭТР может содержать перекрестные ссылки, таблицы, иллюстрации, ссылки на аудио- и видеоданные. Предусматривается функция поиска данных. ИЭТР может быть просмотрен на экране и распечатан без предварительной обработки.

Примечание — Основным недостатком ИЭТР классов 1 и 2 является дублирование многократно используемой информации.

Класс 3 — иерархически-структурированные электронные документы

В ИЭТР этого класса данные хранятся как объекты внутри хранилища информации, имеющего иерархическую структуру. Дублирование многократно используемых данных предотвращается системой ссылок на однократно описанные данные.

Так как данные в ИЭТР этого класса организованы иерархически, документация не может быть распечатана без предварительной обработки.

Класс 4 — интегрированные ИЭТР

В добавление к функциям ИЭТР класса 3, ИЭТР данного класса обеспечивает возможность прямого интерфейсного взаимодействия с электронными модулями диагностики изделий. ИЭТР класса 4 позволяет наиболее эффективно проводить операции по поиску неисправностей в изделии, локализации сбоев, подбору запасных частей.

В настоящих рекомендациях рассматриваются требования к организации БД для поддержки интерактивных электронных технических руководств классов 3 и 4.

6 Общие принципы организации баз данных ИЭТР

6.1 Назначение баз данных ИЭТР

БД ИЭТР представляет собой совокупность информационных объектов, содержащих техническую информацию об изделии или требуемом для его эксплуатации или ремонта оборудовании, структурированных определенным образом. Стандартизация структуры БД обеспечивает:

- возможность передачи БД или ее части между различными организациями;
- централизованное управление данными, составляющими ИЭТР;
- возможность автоматизации процесса разработки ИЭТР;
- предоставление доступа к необходимой информации службам и организациям, осуществляющим материально-техническое обеспечение соответствующих изделий.

6.2 Методика описания структуры базы данных ИЭТР

Настоящие рекомендации содержат требования к логической структуре БД ИЭТР. Эти требования относятся как к БД, создаваемым разработчиком ИЭТР для подготовки и сопровождения проектов ИЭТР, так и БД, входящим в состав конкретных ИЭТР. Различия в структурах упомянутых баз данных рассмотрены ниже.

Для описания требований к структуре БД используется терминология языка SGML. В соответствии с требованиями ИСО 8879 [1] структура БД описывается путем декларации (объявления) набора информационных объектов, составляющих БД, их атрибутов, связей и иерархии. Совокупность указанных объявлений в терминах SGML называется описанием логической структуры документа — DTD. Таким образом, БД любого ИЭТР представляет собой совокупность данных, логическая структура которых соответствует некоторому заданному DTD.

Краткие сведения о языке SGML (ИСО 8879 [1]) приведены в приложении А.

7 Общие требования к структуре баз данных ИЭТР

7.1 Основные понятия

База данных ИЭТР включает в себя совокупность информационных объектов различного типа. Каждый информационный объект может иметь набор атрибутов и связей с другими информационными объектами. Информационные объекты связаны друг с другом иерархически.

Информационные объекты условно разделены на два типа: общий (generic) и контекстный (content specific). К информационным объектам общего типа относятся простые объекты (примитивы), такие как фрагменты текста, графические изображения, диалоги. Информационные объекты общего типа могут применяться в любых ИЭТР, независимо от их конкретного назначения и содержания (контекста).

Информационные объекты контекстного типа являются составными и включают в себя объекты общего типа или более простые объекты контекстного типа. Каждый объект контекстного типа представляет собой логически завершенную единицу информации (параграф, раздел и т. д.), форма и содержание которой зависят от назначения и содержания конкретного ИЭТР.

Для описания объектов обоих типов используются шаблоны (templates), регламентирующие перечень обязательных атрибутов объектов, а также иерархию (входимость и взаимосвязь) объектов как общего, так и контекстного типов. В структуре БД объекты общего типа находятся на нижнем уровне иерархии, как показано на рисунке 1.

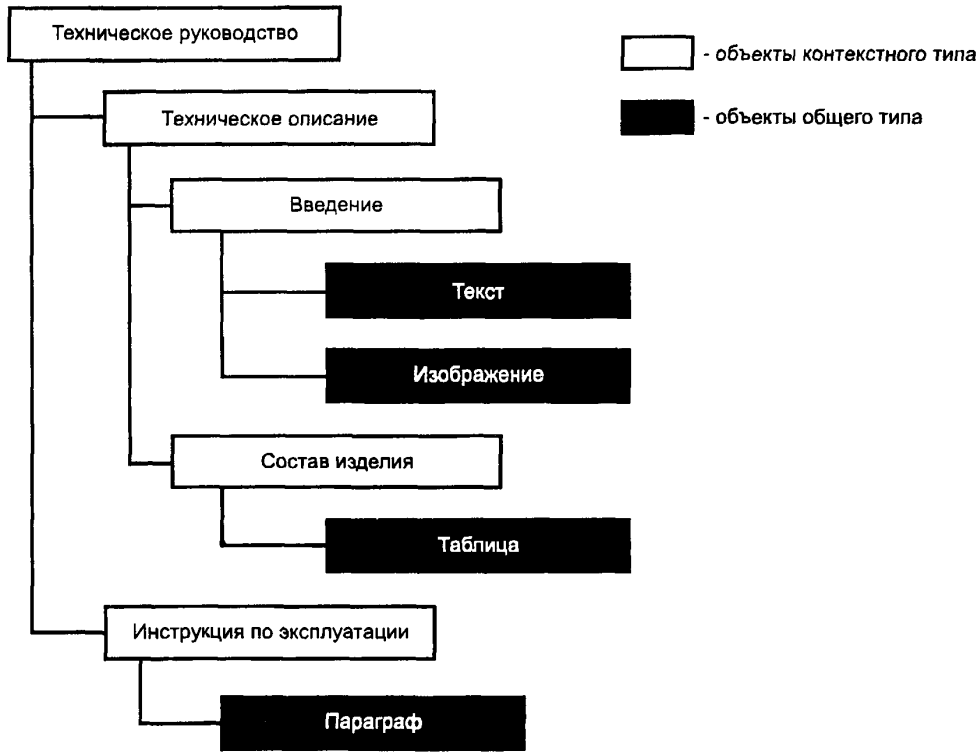


Рисунок 1 — Логическая структура базы данных

Совокупность деклараций (описаний) объектов общего типа, их атрибутов и связей составляет DTD общего типа — единого для любых ИЭТР.

Перечень информационных объектов контекстного типа в БД, их атрибутов и связей определяется спецификой области применения ИЭТР и смысловым содержанием представляемой информации. Совокупность деклараций объектов контекстного уровня, их атрибутов и связей образует DTD контекстного типа. Таким образом, DTD контекстного типа представляет собой набор правил смысловой структуризации данных. Разработка DTD контекстного типа должна осуществляться для каждой предметной области и является объектом отраслевой стандартизации.

Настоящие рекомендации устанавливают требования к стандартизации DTD общего типа и правила создания DTD контекстного типа. Пример DTD контекстного типа приведен в приложении В.

Настоящие рекомендации не накладывают каких-либо ограничений на выбор конкретной системы управления базой данных (СУБД) для создания или применения ИЭТР.

7.2 Требования к базе данных в части обмена данными

Для обеспечения совместимости данных при передаче БД целиком или некоторой ее части, все передаваемые данные должны быть структурированы и размечены в соответствии с требованиями ИСО 8879 [1] и настоящими рекомендациями. Перечень объектов общего типа, используемых в базе данных, должен соответствовать рекомендациям. Каждый объект контекстного типа должен быть описан в применяемом DTD. Формат данных о стиле представления информации должен соответствовать применяемому DTD и спецификации CSS Level 2 [3].

7.3 Сопровождение данных

При внесении изменений структура БД должна обеспечивать автоматическое изменение всех объектов и связей, затрагиваемых таким изменением. Эти требования касаются:

а) удаления, добавления или изменения отдельных информационных объектов или их атрибутов;

б) изменения отношений, включая создание новых связей или удаление старых связей.

7.4 Переносимость данных

Для обеспечения переносимости и мобильности данных физическое хранилище данных должно обеспечивать загрузку (ввод) данных, а также выгрузку (вывод) данных, представленных в формате ИСО 8879 [1] и соответствующих требованиям настоящего документа, а также спецификациям стиля CSS Level 2 [3].

7.5 Обмен информацией с системой управления данными об изделии (PDM)

Для актуализации содержимого базы данных ИЭТР рекомендуется использовать уже имеющуюся цифровую информацию об изделии, накопленную при его проектировании или модернизации. Как правило, эти данные хранятся в системе управления данными об изделии (PDM). Взаимодействие компьютерной системы разработки ИЭТР с системами PDM может осуществляться в соответствии с ГОСТ Р ИСО 10303-21 либо ГОСТ Р ИСО 10303-22.

8 Шаблоны

Для описания информационных объектов используются шаблоны (templates), регламентирующие перечень обязательных атрибутов и иерархию объектов. Каждый информационный объект должен быть описан в каком-либо шаблоне. Шаблон содержит две составляющие: набор семантических правил и перечень атрибутов. Различают три вида шаблонов: простой, шаблон последовательности и шаблон альтернатив.

Шаблоны используются при создании логической структуры БД конкретного ИЭТР следующим образом:

- простой шаблон указывает, что при создании БД должен быть создан отдельный объект с заданными атрибутами и связями;
- шаблон последовательности указывает, что при создании структуры БД должно быть создано несколько объектов, связанных в указанной последовательности;
- шаблон альтернатив указывает, что в структуре базы данных, используемой для разработки ИЭТР, создается несколько объектов в соответствии со списком, указанным в шаблоне.

В процессе публикации конкретного ИЭТР в соответствии с сделанным выбором в базу данных этого ИЭТР попадает только один объект.

8.1 Простой шаблон

Общее описание шаблонов в терминах SGML приведено ниже. Подробное описание синтаксиса и семантики обозначений приведено в приложении А.

```
<!ELEMENT NODE - - (version*, link*, (NODE | NODE-ALTS |
                    NODE-SEQ | %primitive; )*) >
<!ENTITY % a.node
                «id       ID       #IMPLIED
                Name      CDATA    #IMPLIED
                Type      CDATA    #IMPLIED
                Itemid    CDATA    #IMPLIED
                Cdm       NAME     #FIXED   'node'
                Ref       IDREF    #CONREF
                Version   IDREF    #IMPLIED">
```

Первый объект в модели содержания — версия (version) — используется для соотнесения данных с вариантами конфигурации изделия.

Второй объект (link) задает перечень ссылок на объекты, связанные по смыслу. Используется для навигации внутри ИЭТР и прямых переходов от одного раздела к другому.

Шаблон содержит указание о том, что описываемый составной информационный объект может включать в себя другие объекты, которые в свою очередь должны соответствовать простому

шаблону, шаблону альтернатив или шаблону последовательности, или могут представлять собой примитивы.

Макрос a.node описывает обязательные атрибуты описываемого объекта:

1 Id — уникальный идентификатор объекта. Используется для организации ссылок;
2 Name — название раздела документа, содержащегося в данном объекте;
3 Itemid — обозначение изделия, к которому относится данный раздел;
4 Ref — атрибут, используемый для исключения дублирования информации;
5 Type — тип объекта;
6 Cdm — атрибут, указывающий, что объект относится к простому шаблону;
7 Version — ссылка на объект version. Указывает, к какой версии относится данный объект.

8.2 Шаблон последовательности

Шаблон последовательности является механизмом для создания интерактивных последовательностей общения с пользователем. Шаблон последовательности определяется следующим образом:

```
<!ELEMENT NODE-SEQ  - -  ( NODE | NODE-ALTS)+ >
<!ENTITY % a.node-seq
        "id       ID       #IMPLIED
        Cdm       NAME     #FIXED   'node-seq'
        Ref       IDREF    #CONREF"          >
```

В отношении шаблона последовательности действуют следующие правила:

- любой объект, описываемый шаблоном последовательности, должен содержать объекты, соответствующие простому шаблону или шаблону альтернатив;
- объекты отображаются для пользователя в том порядке, в котором они перечислены в шаблоне.

8.3 Шаблон альтернатив

Шаблон альтернатив содержит перечень взаимоисключающих объектов, применимых к различным версиям ИЭТР. Как указывалось выше, в базе данных, используемой при разработке ИЭТР, создаются все объекты, перечисленные в шаблоне. В процессе подготовки публикации конкретного

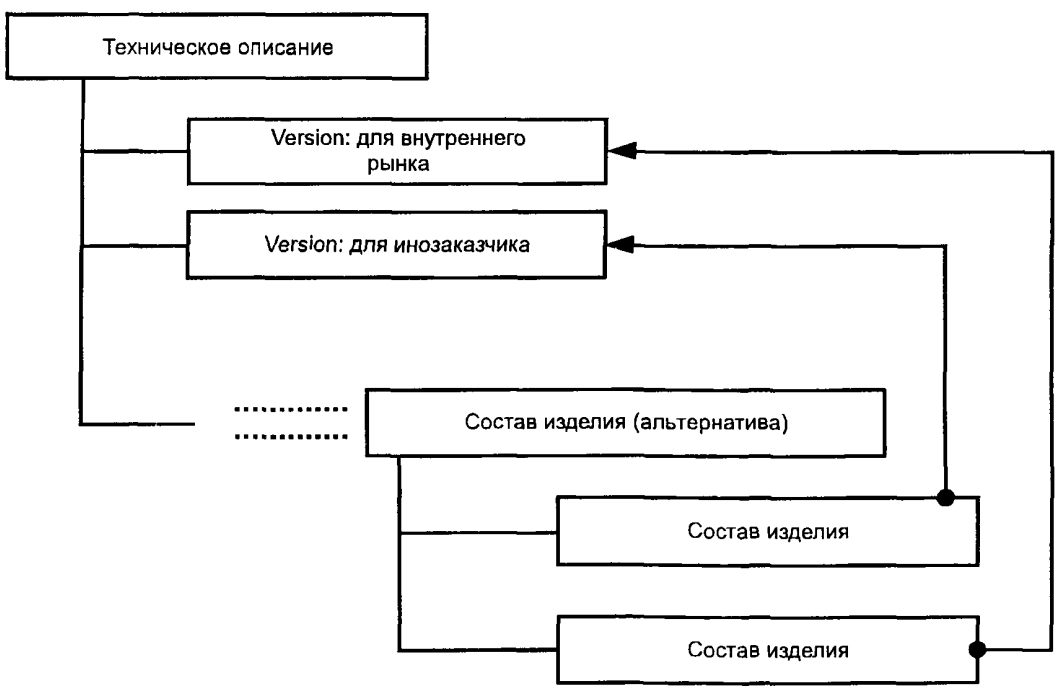


Рисунок 2

ИЭТР из списка взаимоисключающих объектов выбирается один, который и попадает в базу данных этого ИЭТР.

```
<!ELEMENT "NODE-ALTS"     - -     ( NODE )+ >
<!ENTITY % a.node-alts
                "id       ID          #IMPLIED
                Cdm       NAME        #FIXED   'node-alts'
                Ref       IDREF       #CONREF"              >
```

К любому объекту, описываемому шаблоном альтернатив, относятся следующие правила (рисунок 2):

- составной объект может содержать более простые объекты, соответствующие простому шаблону;
- объекты должны принадлежать к одному типу и находиться на одном уровне иерархии;
- объекты должны относиться к разным вариантам конфигурации или версиям изделия;
- для каждой возможной конфигурации (версии) должен быть задан свой информационный объект.

9 Примитивы общего типа

Простые информационные объекты (примитивы) общего типа предназначены для хранения текстовой, графической, аудио- и видеоинформации. Примитивы общего типа делятся на три группы: служебные объекты, визуализируемые объекты, объекты диалога.

К служебным (не визуализируемым) объектам относятся **link** (ссылка) и **version** (версия). Объект **link** позволяет пользователю осуществлять навигацию по базе данных, двигаясь по сети возможных переходов от раздела к разделу. Объект **version** (версия) используется для обозначения вариантов конфигурации или версий изделия.

Визуализируемые объекты предназначены для составления параграфа ИЭТР — основной смысловой единицы технических данных, представляющей собой линейный гипертекстовый документ.

Объекты диалога обеспечивают возможность интерактивного общения с пользователем посредством приема от него ключевой информации для запроса, ее анализа и отображения запрашиваемых данных.

В соответствии с требованиями настоящего документа, каждый объект общего типа относится к какому-либо шаблону, описанному в предыдущем разделе. К объектам общего типа (примитивам) могут быть применены различные спецификации стиля, соответствующие CSS2[3].

9.1 Служебные объекты

Служебные объекты предназначены для структурирования данных и создания гипертекстовых связей внутри базы данных.

9.1.1 Ссылка

Объект **link** обеспечивает навигацию внутри базы данных ИЭТР и взаимосвязь объектов. Любой объект, соответствующий простому шаблону, может иметь одну или более ссылок на другие объекты. Декларация объекта **link** приведена ниже:

```
<!ELEMENT link    - -   ( #PCDATA ) >
<!ATTLIST link
          %a.node;
          Xref              IDREF       #REQUIRED>
```

Атрибут **xref** указывает на целевой объект ссылки, то есть на объект, который будет представлен пользователю, активизировавшему данную ссылку. Текстовые данные (#PCDATA), включенные в модель содержания, представляют собой необязательное текстовое описание данной ссылки.

Пример применения ссылки в документе SGML:

```
<text id=34 name="Инструкция по ...">
        <link xref=278> Переход на техническое описание
        </link>
        <link xref=279> Переход на диагностику неисправностей
        </link>
```

Инструкция по эксплуатации .....
</text>
.....
<techdesc id=278> Техническое описание ..... </techdesc>
.....
<faults id=279> Диагностика неисправностей ..... </faults>
.....

9.1.2 Версия

Объект **version** включен в декларацию простого шаблона, что обеспечивает возможность задания в базе данных несколько версий для одного объекта. Декларация объекта **version** приведена ниже:

```
<!ELEMENT version - - (#PCDATA)>
<!ATTLIST version %a.node; >
```

Текстовые данные (#PCDATA), включенные в модель содержания, должны представлять собой описание версии. Атрибут **name** должен содержать обозначение версии. Объект **version** используется только на этапе разработки ИЭТР. При публикации конкретного ИЭТР выбирается и переносится в БД ИЭТР только та версия, которая актуальна для данного заказчика.

Пример использования описания версий в документе SGML:

```
<techdesc name="Техническое описание ...">
        <version id="V1" name="Вариант для внутреннего рынка">
        </version>
        <version id="V2" name="Экспортный вариант">
        </version>
 .....
</techdesc>
```

9.2 Визуализируемые объекты

Визуализируемые объекты предназначены для построения линейных гипертекстовых документов, представляемых пользователю посредством ЭСО в виде единого целого. Линейный документ представлен объектом **para** (параграф).

9.2.1 Параграф

Параграф представляет собой произвольный набор примитивов, в совокупности образующих законченную смысловую единицу технических данных.

Совокупность примитивов, составляющих параграф, описывается объектом **primitive.** Примитивы могут соответствовать простому шаблону или шаблону альтернатив. Декларации примитивов приведены ниже:

```
<!ENTITY % text         "text  | text-alts">
<!ENTITY % list         "list  | list-alts">
<!ENTITY % table        "table | table-alts">
<!ENTITY % graphic          "graphic   | graphic-alts">
<!ENTITY % grphprim         "grphprim | grphprim-alts">
<!ENTITY % f                "f         | f-alts">
<!ENTITY % audio            "audio     | audio-alts">
<!ENTITY % video            "video     | video-alts">
<!ENTITY % process          "process   | process-alts">
<!ENTITY % dialog           "dialog    | dialog-alts">
<!ENTITY % object           "object    | object-alts">
<!ENTITY % primitive        "%text; | %list; | %table; |
                             %graphic; | %f; | %audio; |
                              %video; | %process; |
                              %dialog; | %object;">
```

Декларация объекта **para**, соответствующего простому шаблону, описывается в терминах языка SGML следующим образом:

```
<!ELEMENT para - -  (version*,link*,(%primitive)+)>
<!ATTLIST para      %a.node;>
```

На экране ЭСО, приведенном на рисунке 3, примитивы, составляющие параграф, отображаются последовательно, в соответствии с указанным порядком и спецификацией стиля.

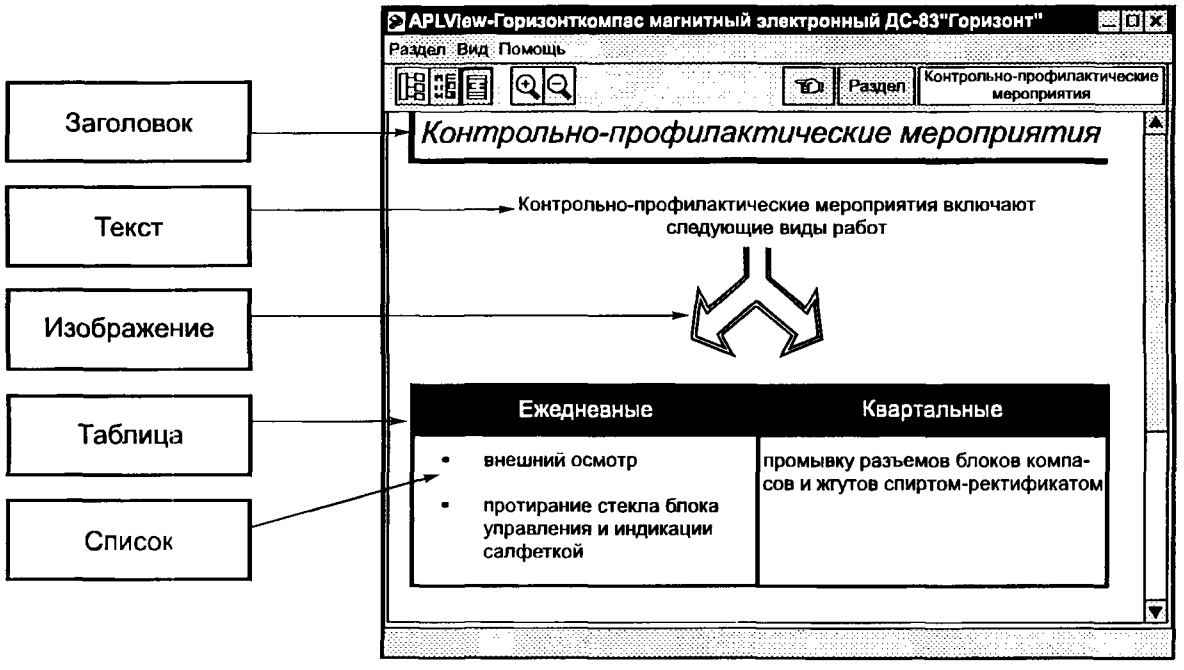


Рисунок 3 — Отображение параграфа при помощи ЭСО

Кроме того, для поддержки вариантов конфигурации и версий вводится объект **para-alts** (альтернативный параграф). Каждый альтернативный параграф относится к определенной версии ИЭТР, и только один попадает в ИЭТР для конкретного заказчика.

<!ELEMENT para-alts - - (para+)>
<!ATTLIST para-alts %a.node-alts;>

Далее в настоящих рекомендациях не будет подробно описываться назначение других альтернативных объектов, предназначенных для решения аналогичных задач.

Текстовая информация должна быть включена в базу данных при помощи объекта **text**, соответствующего простому шаблону. Сами текстовые данные должны быть включены в модель содержания (#PCDATA). Декларация объекта **text** приведена ниже:

<!ELEMENT text - - (link*,(#PCDATA))>
<!ATTLIST text %a.node;>

Для поддержки версионности вводится объект text-alts:

<!ELEMENT text-alts - - (text+)>
<!ATTLIST text-alts %a.node-alts;>

Пример использования текста в документе SGML

<text> Корпуса всех блоков и экранирующие оплетки жгутов должны быть заземлены.</text>

9.2.2 Список

Список представляется в виде перечня объектов, каждый из которых может быть примитивом различного типа. В терминах SGML список определяется следующим образом:

<!ELEMENT list - - (link*,(listitem)+)>
<!ATTLIST list %a.node;>

Каждый объект из списка может содержать произвольное число примитивов различного типа.

<!ELEMENT listitem - - (link*,(%primitive)+)>

Схема отображения списка при помощи ЭСО показана на рисунке 4.

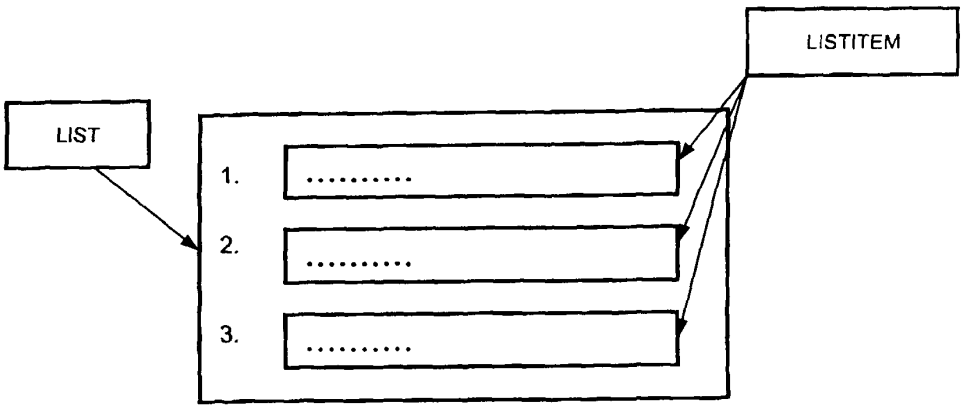


Рисунок 4 — Схема отображения списка при помощи ЭСО

Для поддержки версионности вводится объект **list-alts** (альтернативный список):

```
<!ELEMENT list-alts - - (list)+>
<!ATTLIST list-alts %a.node-alts;>
```

9.2.3 Таблица

Объект **table** (таблица) предназначен для визуальной и логической группировки данных в табличной форме. Таблица задается необязательными головкой и боковиком, включающими текстовые данные, и последовательностью строк таблицы, первая из которых содержит головку, а остальные строки — ячейки таблицы:

```
<!ELEMENT table - - (link*,caption?,trow+)>
<!ATTLIST table %a.node;>
```

Заготовок таблицы содержит обычные текстовые данные

```
<!ELEMENT caption - - (text)>
<!ATTLIST caption
          id        ID      #IMPLIED>
```

Строка таблицы задается последовательностью ячеек. Если строка является головкой таблицы, в качестве ячеек выступают элементы thead, в случае стандартной строки таблицы — tdata:

```
<!ELEMENT trow - - (thead | tdata)+>
<!ATTLIST trow
          id        ID      #IMPLIED>
```

Ячейки таблицы (объекты thead и tdata) определяются следующим образом:

```
<!ELEMENT (thead | tdata) - - (%primitive)+>
<!ATTLIST (thead | tdata)
          id        ID      #IMPLIED
          colspan   NUMBER  #IMPLIED
          rowspan   NUMBER  #IMPLIED>
```

Таким образом, ячейки таблицы могут содержать примитивы произвольного типа. Кроме того, можно объединять ячейки по вертикали и горизонтали (рисунок 5), задав число столбцов, которое занимает ячейка (атрибут colspan), и число строк, которое занимает ячейка (атрибут rowspan).

Для поддержки версионности вводится объект **table-alts** (альтернативная таблица):

```
<!ELEMENT (table-alts - - (table)+>
<!ATTLIST table-alts %a.node-alts;>
```

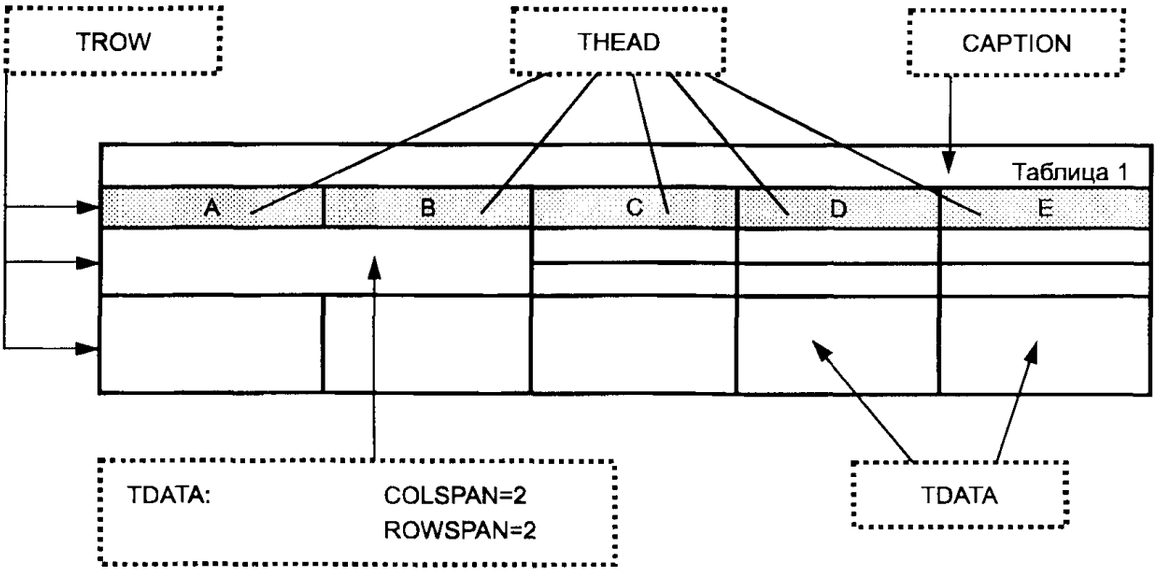


Рисунок 5

9.2.4 Композиционное изображение

Композиционное изображение задается набором графических примитивов заданного взаимного расположения (рисунок 6). Каждый графический примитив может содержать ссылку. Таким

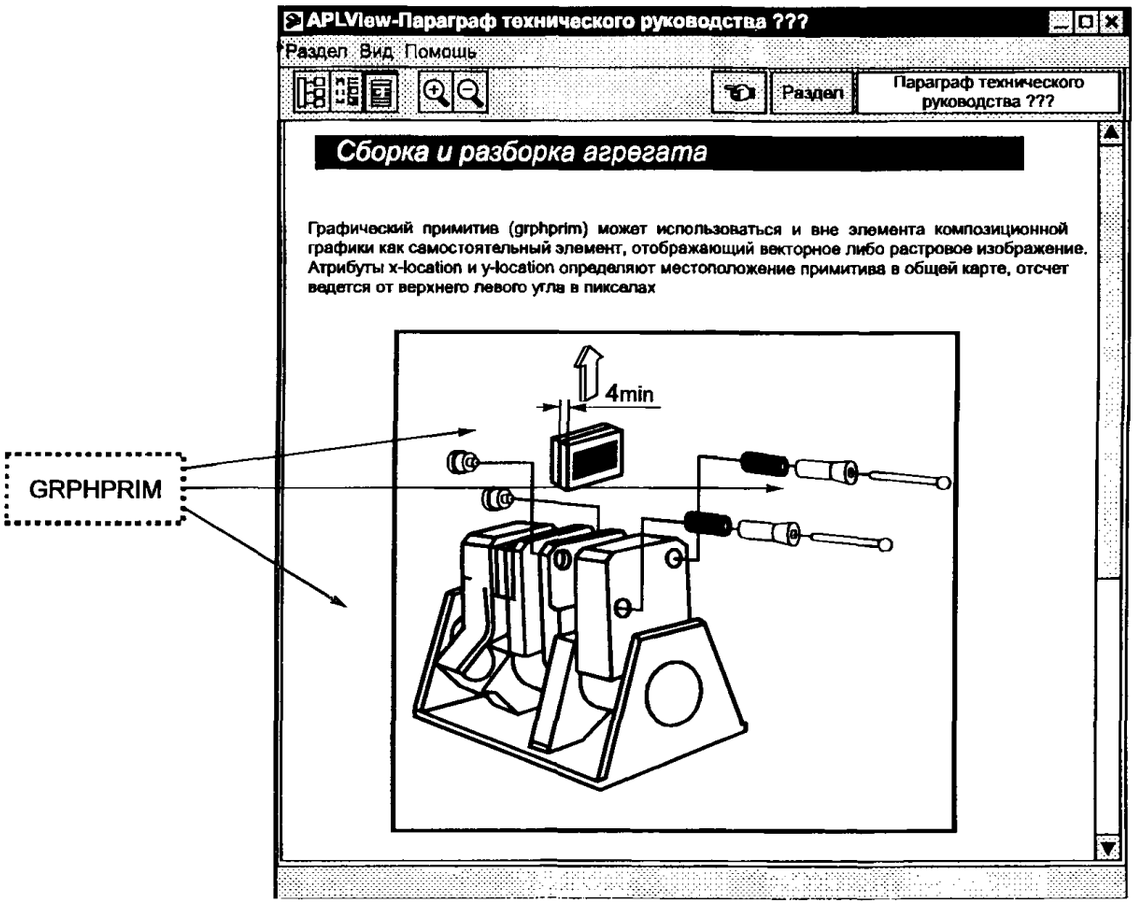


Рисунок 6 — Пример композиционной графики

образом, в ИЭТР на базе графических изображений можно создавать карты ссылок.

<!ELEMENT graphic - - (link*,grphprim+)>
<!ATTLIST graphic %a.node;>
<!ELEMENT grphprim - - (link*,#PCDATA)>

```
<!ATTLIST grphprim      %a.node;
                        x-location   NUMBER   #IMPLIED
                        y-location   NUMBER   #IMPLIED
                        source       NUMBER   #REQUIRED>
```

Графический примитив (grphprim) может использоваться самостоятельно для отображения векторного либо растрового изображения. Атрибуты x-location и y-location определяют местоположение примитива; отсчет ведется от верхнего левого угла в пикселах.

Для поддержки версионности вводятся соответствующие альтернативные объекты **graphic-alts** и **grphprim-alts:**

```
<!ELEMENT graphic-alts  - -  (graphic)+>
<!ATTLIST graphic-alts          %a.node-alts;>
<!ELEMENT grphprim-alts - -     (grphprim+)>
<!ATTLIST grphprim-alts         %a.node-alts;>
```

9.2.5 Аудиоданные

Объект audio позволяет включать в ИЭТР аудиопоследовательности. Декларация объекта приведена ниже:

```
<!ELEMENT audio    - -   ( #PCDATA ) >
<!ATTLIST audio
                        %a.node;
                        source   ENTITY    #REQUIRED
                        start    (manual | auto) 'auto'
                        play     (loop | once) 'loop'>
```

Текстовые данные, включенные в модель содержания, представляют текстовое описание аудиопоследовательности. Атрибуты start и play определяют метод и количество воспроизведений аудиопоследовательности.

Для поддержки версий вводятся объекты:

```
<!ELEMENT audio-alts  - -  (audio)+>
<!ATTLIST audio-alts        %a.node-alts;>
```

9.2.6 Видеоданные

Объект video позволяет включать в ИЭТР видеопоследовательности. Декларация объекта приведена ниже:

```
<!ELEMENT video  - -     (#PCDATA)  >
<!ATTLIST   video % a.node;
            source    ENTITY    #REQUIRED
            start     (manual | auto) 'auto'
            play      (loop | once) 'loop'>
```

Текстовые данные, включенные в модель содержания, представляют текстовое описание видеопоследовательности. Атрибуты start и play определяют метод и количество воспроизведений видеопоследовательности.

Для поддержки версионности вводятся объекты:

```
<!ELEMENT video-alts  - -  (video)+>
<!ATTLIST video-alts        %a.node-alts;>
```

9.2.7 Объект

Объект **object** позволяет включать в технические руководства произвольные предопределенные данные в стандартных форматах, такие как 3D модели.

```
<!ELEMENT object   - -   ( link*, #PCDATA) >
<!ATTLIST object   %a.node;
                   source ENTITY   #REQUIRED >
```

Для поддержки версионности вводятся объекты:

```
<!ELEMENT object-alts  - -  (object)+>
<!ATTLIST object-alts       %a.node-alts;>
```

9.3 Объекты диалога

Объекты поддержки диалога обеспечивают интерактивное взаимодействие с пользователем.

9.3.1 Процесс

Объекта **process** позволяет производить обращение к внешнему программному процессу с передачей установленных параметров. Декларация объекта приведена ниже:

```
<!ELEMENT process - - (link*,parameter*) >
<!ATTLIST process        %a.node;
                source ENTITY      #REQUIRED >
```

Результатом обращения к процессу может являться порядковый номер ссылки (от 1 до *N*, где *N* — число ссылок объекта process), по которой будет осуществляться переход после обращения к процессу.

Параметр, передаваемый процессу, описывается объектом **parameter**. Значение параметра должно быть взято из объекта, на который ссылается атрибут **source**.

```
<!ELEMENT parameter  - -  (#PCDATA) >
<!ATTLIST parameter
                        source IDREF #REQUIRED >
```

Для поддержки версионности вводится объект:

```
<!ELEMENT process-alts  - - (process)+>
<!ATTLIST process-alts        %a.node-alts;>
```

9.3.2 Диалог

Диалог является основным механизмом интерактивного взаимодействия с пользователем. Зачастую требуется запросить у пользователя какую-либо информацию. Эта функциональная возможность обеспечивается объектом **dialog**, включающим в себя различные объекты диалогового взаимодействия. Совокупность объектов диалогового взаимодействия описывается объектом **form**:

```
<!ENTITY % form "button | choice | radio | input | check | selection">
<!ELEMENT dialog - - ((%primitive; | %form;)+,process*)>
<!ATTLIST dialog        %a.node;>
```

Результат диалогового взаимодействия может обрабатываться объектом process.

Для поддержки версий вводятся объекты:

```
<!ELEMENT dialog-alts  - -  (dialog)+>
<!ATTLIST dialog-alts        %a.node-alts;>
```

9.3.3 Кнопки

Кнопка как часть интерфейса описывается элементом button. Кнопка может содержать изображение и текст (рисунок 7):

```
<!ELEMENT button - - (grphprim?,#PCDATA)>
<!ATTLIST button
                %a.node;
                target IDREF        #IMPLIED>
```

Атрибут target определяет объект, на который будет осуществлен переход по нажатию этой кнопки (этим объектом может являться объект process).

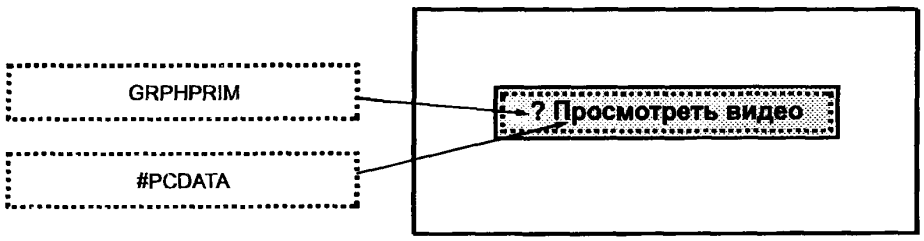


Рисунок 7 — Кнопка

9.3.4 Поле со списком

Поле со списком представляет собой поле ввода в выпадающим меню возможных значений. Оно описывается объектом **choice.** Поле со списком содержит объекты **item**, определяющие перечень возможных значений:

```
<!ELEMENT choice - - (item+)>
<!ATTLIST choice %a.node;
                default          NUMBER #IMPLIED
                target IDREF          #IMPLIED>
<!ELEMENT item - - (#PCDATA)>
```

Необязательный атрибут target определяет объект, на который будет осуществлен переход после выбора пользователем значения (этим объектом может являться **process)**.

В случае, если поле со списком (рисунок 8) является целевым объектом атрибута **source** объекта **parameter**, в качестве значения передается номер выбранного пользователем значения (нумерация начинается с единицы).

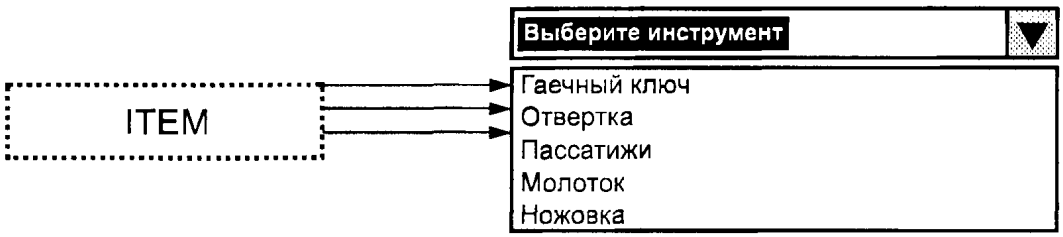


Рисунок 8 — Список

9.3.5 Переключатель

Переключатель представляет собой визуальный объект (рисунок 9), позволяющий выбрать одну из возможных альтернатив. Он описывается объектом **radio**. Переключатель содержит объекты **item**, определяющие перечень возможных альтернатив:

<!ELEMENT radio - - (item+)>
<ATTLIST radio
%a.node;
default NUMBER #IMPLIED>

В случае, если переключатель является целевым объектом атрибута **source** объекта parameter, в качестве значения передается номер выбранной альтернативы (нумерация начинается с единицы).

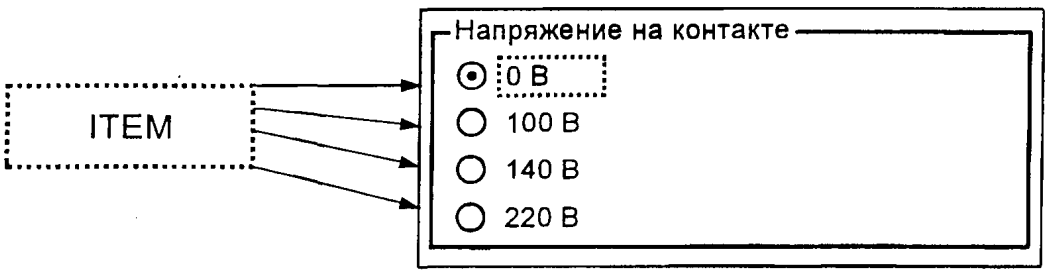


Рисунок 9 — Переключатель

9.3.6 Поле ввода

Поле ввода представляет собой визуальный объект, позволяющий ввести текстовые данные (рисунок 10). Оно описывается объектом **input**. Поле ввода содержит текст, появляющийся по умолчанию:

<!ELEMENT input - - (#PCDATA)>
<ATTLIST input %a.node;>

В случае, если поле со списком является целевым объектом атрибута **source** объекта **parameter**, в качестве значения передается номер выбранного пользователем значения (нумерация начинается с единицы).

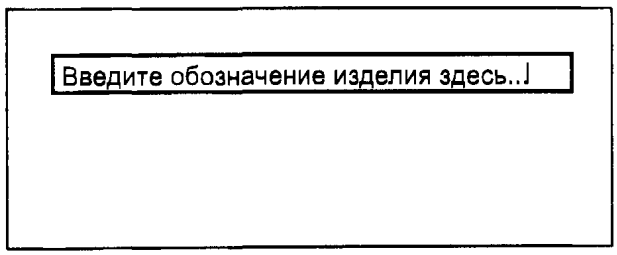


Рисунок 10 — Поле ввода

9.3.7 Флажок

Флажок представляет собой визуальный объект, позволяющий установить его значение или сбросить (рисунок 11). Он описывается объектом **check**.

```
<!ELEMENT check - - (#PCDATA)>
<!ATTLIST check    %a.node;>
```

В случае, если флажок является целевым объектом атрибута **source объекта parameter**, в качестве значения передается единица, если флажок установлен, и ноль — в противном случае.

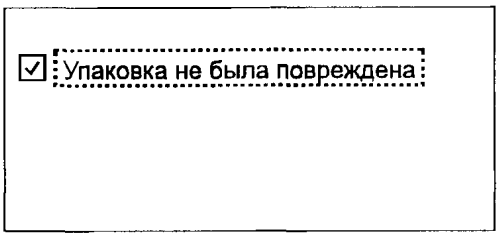


Рисунок 11 — Флажок

9.3.8 Список значений

Список значений представляет собой визуальный объект (рисунок 12), позволяющий выбрать одно значение из нескольких приведенных. Он описывается объектом **selection**.

```
<!ELEMENT selection - - (item+)>
<!ATTLIST selection      %a.node;>
```

В случае, если поле со списком является целевым объектом атрибута **source** объекта **parameter**, в качестве значения передается номер выбранного пользователем значения (нумерация начинается с единицы).

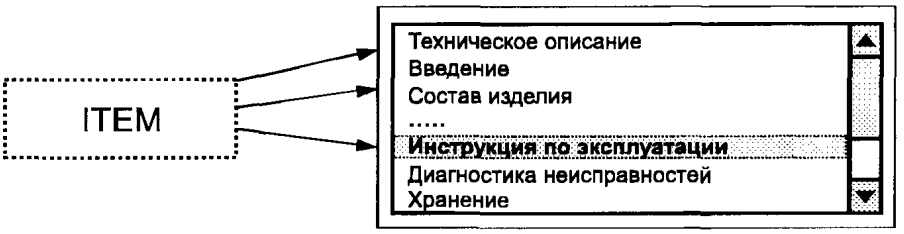


Рисунок 12 — Список значений

9.3.9 Пример использования диалогов

Рассмотрим в качестве примера диалог (рисунок 13), в ходе которого у пользователя запрашиваются параметры неисправности и, в зависимости от указанных параметров, отображаются соответствующие разделы ИЭТР. Например для идентификации неисправности необходимо запросить у пользователя информацию о проведении самотестирования агрегата, информацию о данных на индикаторе и напряжении на контактах ФП-3. Совокупность запрашиваемых данных можно представить в виде диалога следующего вида:

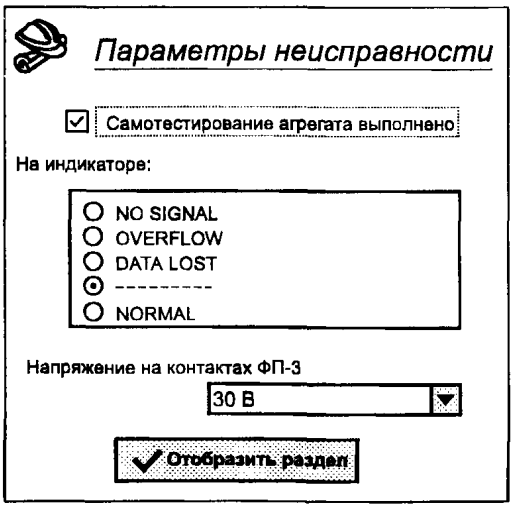


Рисунок 13 — Пример диалога с управляющими объектами

Данный диалог можно ввести в базу данных при помощи описанных выше объектов общего типа. В терминах SGML изображенный выше диалог будет записан следующим образом:

```
<!ENTITY helmet SYSTEM "data/helmet.jpg">
<!ENTITY OK     SYSTEM "data/OK.jpg">
<!ENTITY faultdiags SYSTEM "bin/diags.exe">
.....
<dialog>
<grphprim source=helmet> </grphprim>
<text>Параметры неисправности</text>
<check id=E1>Самотестирование агрегата выполнено</check>
<text> На индикаторе: </text>
</radio id=E2>
    <item>NO SIGNAL</item>
    <item>OVERFLOW</item>
    <item>DATA LOST</item>
    <item>- - - - - - - - -</item>
    <item>NORMAL</item>
</radio>
<text> Напряжение на контактах ФП-3 </text>
<choice id=E3>
    <item>0 Вольт </item>
    <item>30 Вольт </item>
    <item>60 Вольт </item>
</choice>
<button target=fltИСОlation>
    <grphprim source=OK></grphprim>
    Отобразить раздел
</button>
    <process id=fltИСОlation source=faultdiags>
        <link xref=toSwitch> </link>
        <link xref=toLocation> </link>
        <link xref=toNoSignal> </link>
        <parameter source=E1>Диагностика</parameter>
        <parameter source=E2>Индикатор</parameter>
        <parameter source=E3>Напряжение</parameter>
    </process>
</dialog>
```

Заметим, что в базе данных должны присутствовать разделы с идентификаторами toSwitch, toLocation, toNoSignal, которые (по смыслу) должны описывать методы устранения неисправностей при заданных параметрах. Объект **process** должен, в зависимости от параметров, указанных пользователем, осуществлять переход на один из этих разделов.

Данный диалог можно более наглядно представить в виде схемы, представленной на рисунке 14.

Описанные выше декларации объектов общего типа составляют в совокупности DTD общего типа. Файл DTD общего типа приведен в приложении В.

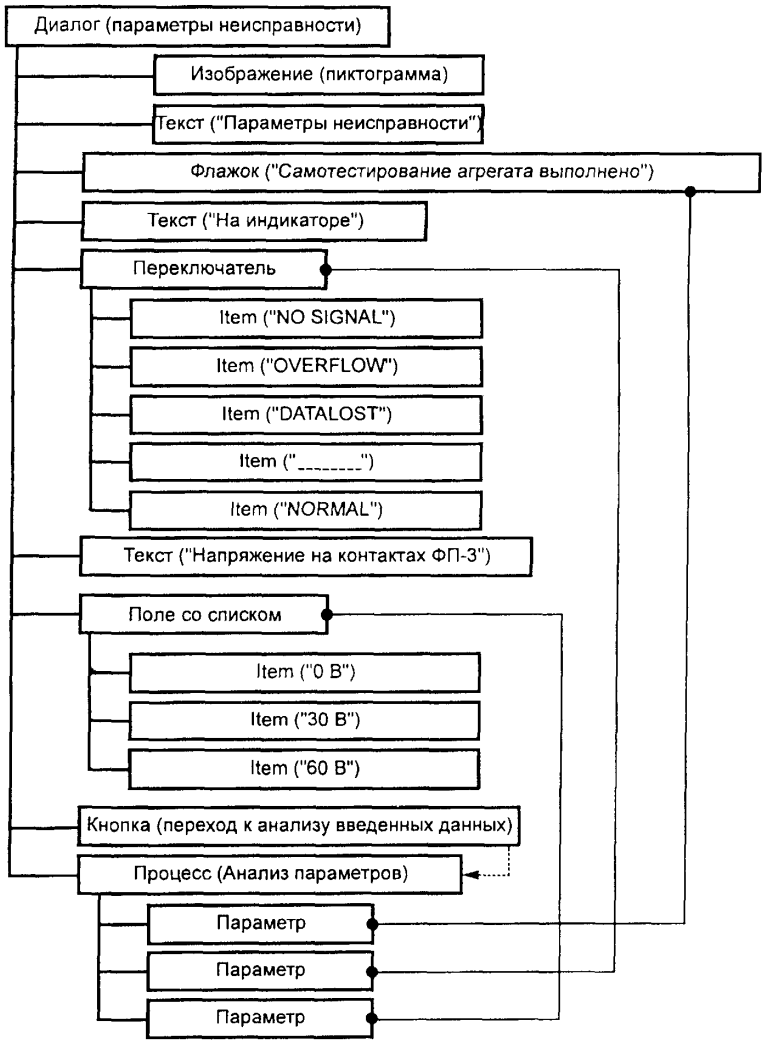


Рисунок 14 — Схема диалога

ПРИЛОЖЕНИЕ А
(справочное)

Краткое описание языка SGML (ИСО 8879)

Под разметкой текста подразумевается внесение в текстовые данные специальных вставок (тегов) с целью выделения и обозначения отдельных информационных объектов

А.1 Общие сведения о языках разметки

Разбиение текста при разметке заключается в установке в начале выделяемой части специальной вставки, называемой открывающим тегом, и в конце выделяемой части — вставки, называемой закрывающим тегом

Пример неразмеченного текста

```
Жегалкин Иван Владимирович Петров Андреи
Владимирович Введение в языки разметки 1989
год Под языками разметки
```

Пример размеченного текста

```
<AUTHOR ID=1>Жегалкин Иван Владимирович</AUTHOR>
<AUTHOR ID=2>Петров Андреи Владимирович</AUTHOR>
<TITLE>Введение в языки разметки <TITLE>
<DATE>1989 год </DATE>
<CONTENT>Под языками разметки
```

Открывающий тег имеет следующий формат

```
<имя_объекта      имя_атрибута1=значение_атрибута1
                  имя_атрибута2=значение_атрибута2
                  имя_атрибута3=значение_атрибута3
                        >
```

Закрывающий тег имеет формат

</имя_ объекта >

Объектом является часть размеченного текста, включающая в себя открывающий и закрывающий теги, а также все данные, находящиеся между ними

Объект может включать в себя другие объекты

Пример

```
<A>
        <B>
                <D> . .. </D>
        </B>
        <C>

        </C>
</A>
```

В данном примере объекты B и C входят в объект A, а объект D — в объект B

Любой корректный SGML документ состоит из двух частей

1) DTD — формальное описание (декларация) объектов, их атрибутов и связей,

2) данные, в которых знаками разметки выделены объекты, объявленные в DTD

А.2 DTD — описание логической структуры документа

DTD представляет собой обычный текстовый документ, состоящий из некоторого количества записей в определенном формате

А.2.1 Описание элемента

Объект объявляется (декларируется) в DTD следующим образом:

А.2.2 Правила минимизации

Правила минимизации представляют собой строку из двух символов, разделенных пробелом. Первый символ показывает, обязательно ли в тексте документа должен присутствовать открывающий тег для объекта данного типа : если символ равен "-", то для данного объекта открывающий тег обязателен, если же первый символ равен "О" (прописная буква, а не ноль), то открывающий тег можно опустить. Второй символ аналогичным образом определяет необходимость указания закрывающего тега.

А.2.3 Модель содержания

Модель содержания определяет обязательную структуру сложного, составного объекта и является обязательной частью декларации объекта. Рассмотрим пример:

<!ELEMENT techdesc - - (title,version,system)>

Правила минимизации показывают, что у элемента **techdesc** обязательно должны присутствовать открывающий и закрывающий теги. Модель содержания говорит о том, что в объекте **techdesc** обязательно должны присутствовать объекты **title, version** и **system** в указанной последовательности.

А.2.4 Индикаторы

В модели содержимого могут присутствовать индикаторы, определяющие количество объектов в составном объекте:

<!ELEMENT techdesc - - (title?,version+,system*)>

"?" — объект может присутствовать, либо нет (0 или 1).

"*" — объект может присутствовать неограниченное число раз подряд, но может и не присутствовать вовсе (≥ 0).

"+" — объект должен присутствовать, по крайней мере, один раз (≥ 1).

Можно установить индикатор сразу на группу объектов, выделив группу в скобки:

<!ELEMENT techdesc - - (title,(version,system)+)>

А.2.5 Конъюнкция объектов

В модели содержания можно снять ограничения на порядок объектов, а также использовать логическую операцию дизъюнкции объектов:

<!ELEMENT techdesc - - (title,version & system)>

В данном случае, в составе объекта **techdesc**, объекты **version** и **system** могут в произвольном порядке следовать за объектом **title**.

А.2.6 Дизъюнкция объектов

<!ELEMENT techdesc - - (title,version | system)>

В данном случае объект **techdesc** должен содержать два объекта, причем вторым объектом может быть либо **version,** либо **system**.

А.2.7 Исключения

Исключения являются необязательным дополнением к модели содержания. Исключения позволяют указать объекты, которые

а) могут входить в составной объект, хотя и не указаны в модели содержания;

б) не должны входить в составной объект.

```
<!ELEMENT techdesc - - (title?,version*,system+)
                       +(crossref)
                       -(drawing)>
```

Данный пример показывает, что в объект techdesc в произвольном месте может входить объект crossref, но не может входить объект drawing.

А.2.8 Ключевые слова

Кроме имен объектов, в модели содержания могут встречаться ключевые слова языка SGML: #PCDATA, ANY и EMPTY.

Ключевое слово #PCDATA означает, что объект может содержать произвольные текстовые данные, например

<!ELEMENT title - - (#PCDATA)>

Ключевое слово ANY означает, что объект может содержать произвольные данные:

<!ELEMENT version - - ANY>

Ключевое слово EMPTY означает, что объект не должен содержать другие объекты. Как правило, если модель содержания объявлена как EMPTY, закрывающий тег разрешается опустить:

<!ELEMENT br - O EMPTY>

А.2.9 Атрибуты объектов

Перечень атрибутов объекта определяется записью следующего вида:

```
<!ATTLIST имя_объекта
        имя_атрибута1     тип_атрибута     по_умолчанию
        имя_атрибута2     тип_атрибута     по_умолчанию
        имя_атрибута3     тип_атрибута     по_умолчанию
        ..........>
```

Поле **имя_объекта** указывает имя объекта, для которого задается перечень атрибутов. Далее идут описания атрибутов, где поле имя_атрибутаNN задает имя атрибута объекта. Имена атрибутов должны состоять из латинских букв и цифр, причем имя обязательно должно начинаться с буквы.

Атрибут может иметь один из приведенных ниже форматов.

a) NAME — значение атрибута может состоять из букв, цифр, символов "—" и "."

c) NMTOKEN — значение атрибута может состоять только из букв и цифр.

c) NUMBER — значение атрибута может состоять из цифр.

d) NUTOKEN — значение атрибута может состоять из букв и цифр, но должно начинаться с цифры.

e) CDATA — значением атрибута могут быть произвольные символьные данные.

f) ID — значением атрибута является уникальный идентификатор данного объекта, состоящий из букв и цифр.

g) IDREF — значением атрибута должен являться идентификатор какого-то другого объекта.

h) Список возможных значений — в скобках через символ "|" перечисляется список возможных значений для данного атрибута.

Значение поля **по_умолчанию** определяет, является ли атрибут обязательным для объекта, а также может задавать значение атрибута по умолчанию. Перечислим возможные значения поля **по_умолчанию**:

а) #REQUIRED — атрибут является обязательным для данного объекта;

б) #IMPLIED — атрибут может отсутствовать;

с) #CURRENT — если значение атрибута не указано, значение будет таким же, как и у аналогичного атрибута последнего обработанного объекта того же типа;

d) #CONREF — в случае, если указано значение для данного атрибута, модель содержимого будет интерпретироваться как EMPTY.

e) любое значение из списка — в случае, если тип документа задан списком возможных значений, значение поля по_умолчанию определяет, какое значение является значением по умолчанию для данного атрибута.

Пример декларации объекта и его атрибутов:

```
<!ELEMENT image - O   EMPTY>
<!ATTLIST image
        name          CDATA                        #IMPLIED
        sysid         CDATA                        #REQUIRED
        encoding      (cgm | bmp | jpeg | wmf | gif)  'cgm'
        minsize       NUTOKEN                      #IMPLIED
        x-location    NUMBER                       #REQUIRED
        y-location    NUMBER                       #REQUIRED
        transform     NAME                         #IMPLIED
        >
```

Пример корректного выделения объекта в документе:

```
<IMAGE      name="вид сбоку"
            sysid="data/sideview.cgm"
            x-location=12
            y-location=0 transform="rotate90cw">
```

А.2.10 Макросы SGML

В языке SGML предусмотрены специальные конструкции, реализующие функции макроподстановки. Указание макроподстановки записывается в следующем виде:

<!ENTITY «строка 1» «строка 2»>

После объявления макроподстановки выражение **«&строка 1»**, обнаруженное в тексте, будет заменено на **«строка 2»**.

Макроподстановки могут использоваться при написании DTD. В этом случае после ключевого слова ENTITY указывается знак процента.

<!ENTITY % название_объекта «значение объекта»>

Макроподстановки используются также для включения в DTD ранее декларированных объектов, в частности описанных в других стандартах. Например указание

<!ENTITY % hytime PUBLIC
"-//ANSI X3V1.8M//DTD Hypermedia/Time-based Document//EN"
"hytime.dtd">
%hytime;

объявляет объекты, описанные в ИСО/МЭК 10744 [2].

А.3 Размеченный документ SGML

Размеченный документ SGML представляет собой текстовый файл либо совокупность текстовых файлов, размеченных в соответствии с некоторым DTD. Любой размеченный документ SGML должен начинаться с объявления типа документа:

<!DOCTYRE techdesc SYSTEM "defs.dtd">

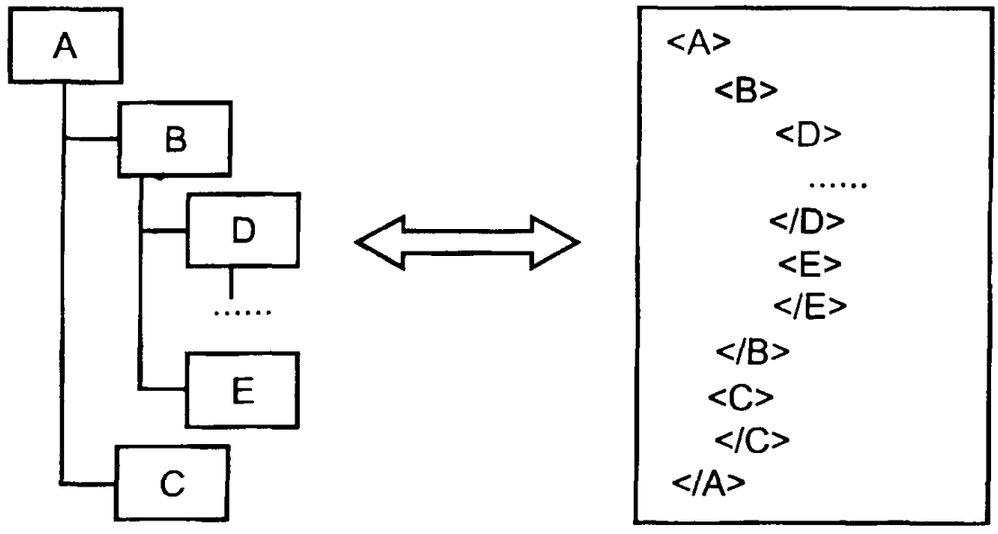
или

<!DOCTYRE techdesc PUBLIC "-//RUS-CALS //DTD Content Data Model //EN ">

где после ключевого слова DOCTYPE указывается корневой объект документа, затем ставится ключевое слово SYSTEM или PUBLIC и указывается местоположение DTD.

Данные, размеченные в соответствии с ИСО 8879 [1], часто бывает удобно представлять в форме графических диаграмм, отображающих иерархию объектов.

Пример. Следующие две формы представления эквивалентны:



А.4 Комментарии

Для улучшения восприятия размеченного текста разрешается в тело документа вставлять комментарии. Комментарий представляет собой текст, поясняющий смысл размеченных данных, описаний объектов и т. п. Синтаксически комментарий представляет собой произвольные текстовые данные, расположенные между открывающим маркером комментария "<!--" и закрывающим маркером комментария "-->".

Наиболее часто комментарии используются в описаниях объектов в DTD.

Пример:

<!ELEMENT fltdiags - - (task,step*,result)>
<!-- Данный объект содержит данные о процедуре диагностики неисправности -->

ПРИЛОЖЕНИЕ Б
(справочное)

Набор объектов для представления математических формул

В данном приложении приведены декларации объектов, предназначенных для представления математических формул.

Б.1 Объект «формула»

Объект «формула» включает в себя набор всевозможных комбинаций объектов, составляющих формулу. Эти объекты задаются макросом **formula**:

<!ENTITY % formula "#PCDATA | f | sup | sub | heap | frac | sqrt | square | power | root | fence | integral | plex | matrix">

Объект «формула» объявляется с использованием указанной макроподстановки следующим образом:

<!ELEMENT f - - (%formula;)+>
<!ATTLIST f %a.node;>

Для обеспечения вариантности используется объект «альтернативная формула»:

<!ELEMENT f-alts - - (f)+>
<!ATTLIST f-alts %a.node-alts;>

Б.2 Нижний и верхний индексы

Верхний и нижний индексы определяются объектами **sup** и **sub** соответственно:

<!ELEMENT sup - - (%formula;)+>
<!ELEMENT sub - - (%formula;)+>

Б.3 Надсимвольные и подсимвольные знаки

Надсимвольные и подсимвольные знаки определяются объектами **heap, over** и **under**.

<!ELEMENT heap - - ((%formula;)+,(over | under))>
<!ELEMENT over - O (%formula;)+>
<!ELEMENT under - O (%formula;)+>

Пример:

Надсимвольный знак **<heap>**выражение**<over>**выражение**</heap>**
Подсимвольный знак **<heap>**выражение**<under>**выражение**</heap>**

Б.4 Дробь

Дробные выражения определяются объектами **frac, over**.

<!ELEMENT frac - - ((%formula;)+,over)>

Пример

<frac>числитель**<over>**знаменатель**</frac>**

Б.5 Корень

Корень определяется объектом **sqrt**.

<!ELEMENT sqrt - - (%formula;)+>

Пример

<sqrt> выражение**</sqrt>**

Б.6 Квадрат

Квадрат определяется объектом **square**.

<!ELEMENT square - - (%formula;)+>

Пример

<square>выражение**</square>**

Б.7 Степень

Степенные выражения определяются объектами **power, degree,of.**

<!ELEMENT degree - O (%formula;)+>
<!ELEMENT of - O (%formula;)+>
<!ELEMENT power - - (%degree,of)>

Пример

<power> <degree>степень**<of>** выражение **</power>**

Б.8 Степенной корень

Степенной корень определяется объектами **root, degree,of.**

<!ELEMENT root - - (degree,of)>

Пример

<root><degree>степень**<of>** выражение **</root>**

Б.9 Интеграл

Интеграл определяется объектами **integral, from,to,of.**

<!ELEMENT integral - - (from,to,of)>
<!ELEMENT from - O (%formula;)+>
<!ELEMENT to - O (%formula;)+>

Пример

<integral><from> выражение **<to>** выражение **<of>** выражение **</integral>**

Б.10 Сумма, произведение, объединение, пересечение

Сумма, произведение, объединение, пересечение определяются объектами **plex, from,to,of.**

<!ELEMENT plex - - (#PCDATA,from,to,of)>

Пример

<plex>символ**<from>** выражение **<to>** выражение **<of>** выражение **</plex>**

В качестве символа могут выступать знаки суммы, произведения, объединения или пересечения.

Б.11 Матрица

Матрица определяется объектами **matrix, col** и **above.** Матрица задается последовательностью столбцов:

<!ELEMENT matrix - - (col)+>
<!ELEMENT above - O ((%formula;)+,above*)>
<!ELEMENT col - - ((%formula;)+,above*)>

Пример

<matrix>
<col>*выражение***<adove>***выражение***<adove>***выражение* **<adove>...</col>**
<col> .....
</matrix>

Б.12 Скобки

Скобки задаются объектом **fence.**

<!ELEMENT fence - - (%formula;)+>
<!ATTLIST fence
type CDATA #IMPLIED>

Атрибут **type** определяет вид скобок:

круглые	—	**round**
квадратные	—	**square**
фигурные	—	**brace**
угловые	—	**angle**
интервал (...]	—	**open left**
интервал [...)	—	**open right**

Центральный разделитель в скобках задается объектом **separator:**

<!ELEMENT separator - O EMPTY>

Пример

<fence type="round">выражение**<separator>** выражение **</fence >**

Б.13 Пример использования элементов математических формул

Рассмотрим в качестве примера, математическое выражение $H = \left\{ x \in X \mid \sum_{i=1}^{n} e^{2i} \int_{0}^{i} \cos \frac{\pi x}{i} dx = 1,\ n = 3...10 \right\}$.

В терминах вышеперечисленных объектов на языке SGML описание приведенной формулы имеет вид:

```
<f>H=
        <fence type="brace">
                x&laysin;X
        <separator>
                <plex>&sum; <from>i=1<to>n<of>
                        e<sup>2i
                        <integral><from>0<to>i<of>
                                cos
                                <frac>
                                        &pi;x
                                <over>
                                        i
                                </frac>
                                dx
                        </integral>
                </plex>
                =1,n=3...10
        </fence>
</f>
```

ПРИЛОЖЕНИЕ В
(справочное)

DTD общего типа

Настоящее приложение содержит пример полного DTD общего типа.

```
<!ENTITY %a.node
                "id       ID              #IMPLIED
                name      CDATA           #IMPLIED
                type      CDATA           #IMPLIED
                itemid    CDATA           #IMPLIED
                cdm       NAME            #FIXED    'node'
                ref       IDREF           #CONREF
                version   IDREF           #IMPLIED" >
<!ENTITY %a.node-seq
                "id       ID              #IMPLIED
                Cdm       NAME            #FIXED    'node-seq'
                Ref       IDREF           #CONREF" >
<!ENTITY %a.node-alts
                "id       ID              #IMPLIED
                Cdm       NAME            #FIXED    'node-alts'
                Ref       IDREF           #CONREF" >
<!ELEMENT link    - - ( #PCDATA ) >
<!ATTLIST link %a.node;
      xref        IDREF       <#REQUIRED>
<!ELEMENT version - - (#PCDATA)>
<!ATTLIST version     %a.node; >
<!ENTITY % text           "text     | text-alts">
<!ENTITY % list           "list     | list-alts">
<!ENTITY % table          "table    | table-alts">
<!ENTITY % graphic                  "graphic      | graphic-alts">
<!ENTITY % grphprim                 "grphprim     | grphprim-alts">
<!ENTITY % f                        "f            | f-alts">
<!ENTITY % audio                    "audio        | audio-alts">
<!ENTITY % video                    "video        | video-alts">
<!ENTITY % process                  "process      | process-alts">
<!ENTITY % dialog                   "dialog       | dialog-alts">
<!ENTITY % object                   "object       | object-alts">
<!ENTITY % primitive "%text; |%list; |%table; |
              %graphic; |%f; | %audio; |
              %video; | %process; |
              %dialog; | %object;">
<!ELEMENT para - -  (version*,link*,)%primitive)+)>
<!ATTLIST para       %a.node;>
<!ELEMENT para-alts - -   (para+)>
<!ATTLIST para-alts        %a.node-alts;>
<!ELEMENT text - -  (link*,(#PCDATA))>
<!ATTLIST text       %a.node;>
<!ELEMENT text-alts - -  (text+)>
<!ATTLIST text-alts       %a.node-alts;>
<!ELEMENT list - - (link*,(listitem)+)>
<!ATTLIST list       %a.node;>
<!ELEMENT listitem - -   (link*, (%primitive)+)>
<!ELEMENT list-alts - -   (list)+>
<!ATTLIST list-alts          %a.node-alts;>
<!ELEMENT table - -      (link*,caption?,trow+)>
<!ATTLIST table        %a.node;>
<!ELEMENT caption - -   (text)>
<!ATTLIST caption
            id              ID       #IMPLIED>
```

```
<!ELEMENT trow - -      (thead | tdata)+>
<!ATTLIST trow
          id            ID      #IMPLIED>
<!ELEMENT (thead | tdata)   - - (%primitive)+>
<!ATTLIST (thead | tdata)
          id            ID          #IMPLIED
          colspan       NUMBER  #IMPLIED
          rowspan       NUMBER  #IMPLIED >
<!ELEMENT table-alts - -    (table)+>
<!ATTLIST table-alts              %a.node-alts;>
<!ELEMENT graphic - -   (link*,grphprim+)>
<!ATTLIST graphic         %a.node;>
<!ELEMENT grphprim - -      (link*,#PCDATA)>
<!ATTLIST grphprim        %a.node;
          x-location      NUMBER  #IMPLIED
          y-location      NUMBER  #IMPLIED
          source          ENTITY    #REQUIRED>
<!ELEMENT graphic-alts - -     (graphic)+>
<!ATTLIST graphic-alts              %a.node-alts;>
<!ELEMENT grphprim-alts - -       (grphprim+)>
<!ATTLIST grphprim-alts           %a.node-alts;>
<!ELEMENT audio   - -    ( #PCDATA ) >
<!ATTLIST audio  %a.node;
          source   ENTITY     #REQUIRED
          start   (manual | auto) 'auto'
          play          (loop | once) 'loop'>
<!ELEMENT audio-alts - -  (audio)+>
<!ATTLIST audio-alts        %a.node-alts;>
<!ELEMENT video - -     ( #PCDATA ) >
<!ATTLIST   video %a.node;
          source   ENTITY   #REQUIRED
          start    (manual | auto) 'auto'
          play          (loop | once) 'loop'>
<!ELEMENT video-alts - -   (video)+>
<!ATTLIST  video-alts             %a.node-alts;>
<!ELEMENT object          - - ( link*,#PCDATA) >
<!ATTLIST object   %a.node;
          source   ENTITY     #REQUIRED >
<!ELEMENT object-alts - -  (object)+>
<!ATTLIST object-alts       %a.node-alts;>
<!ELEMENT process  - -   (link*,parameter*) >
<!ATTLIST process             %a.node;
          source     ENTITY   #REQUIRED >
<!ELEMENT  parameter    - -   (#PCDATA) >
<!ATTLIST parameter
          source IDREF        #REQUIRED >
<!ELEMENT process-alts  - - (process)+>
<!ATTLIST process-alts        %a.node-alts;>
<!ENTITY % form "button | choice | radio | input | check | selection">
<!ELEMENT dialog - - ((%primitive; | %form;)+,process*)>
<!ATTLIST dialog %a.node;>
<!ELEMENT dialog-alts - - (dialog)+>
<!ATTLIST dialog-alts       %a.node-alts;>
<!ELEMENT button - - (grphprim?,#PCDATA)>
<!ATTLIST   button  %a.node;
          target  IDREF       #IMPLIED>
<!ELEMENT choice - - (item+)>
<!ATTLIST choice %a.node;
          default       NUMBER    #IMPLIED
          target IDREF        #IMPLIED>
```

```
<!ELEMENT item - - (#PCDATA)>
<!ELEMENT radio - - (item+)>
<!ATTLIST radio    %a.node;
           default          NUMBER    #IMPLIED>
<!ELEMENT input - - (#PCDATA)>
<!ATTLIST input %a.node; >
<!ELEMENT check - - (#PCDATA)>
<!ATTLISTcheck %a.node; >
<!ELEMENT selection - - (item+)>
<!ATTLIST selection %a.node; >
```

ПРИЛОЖЕНИЕ Г
(рекомендуемое)

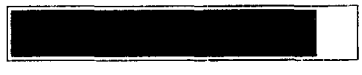
Методика разработки DTD контекстного типа

Г.1 Представление модели содержания в виде дерева

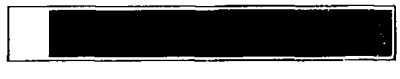
При создании видовых DTD применяются различные визуальные методы представления взаимосвязи объектов. Наиболее распространенным является метод представления иерархии объектов в виде дерева, основанный на декомпозиции модели содержания (приложение А). Для визуального представления используются следующие обозначения:


— недекомпозируемый (терминальный) объект;

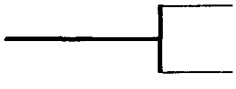
— макрос SGML;

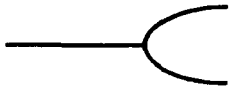

— декомпозируемый объект;

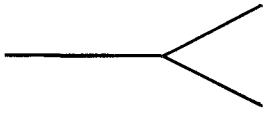

— объект с дополнительными атрибутами;


— корневой объект базы данных.

Соединители описывают последовательное вхождение, дизъюнкцию и конъюнкцию объектов:


— последовательное вхождение;


— конъюнкция объектов;


— дизъюнкция объектов.

Индикаторы задаются следующими обозначениями:

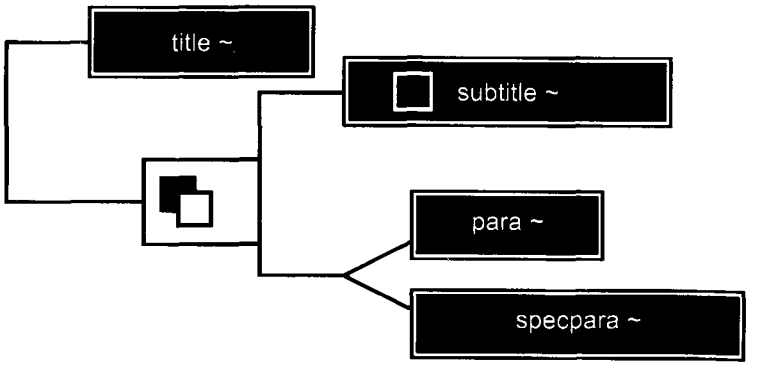
— простое вхождение (нет индикаторов);

— один или более раз (+);

— необязательный объект (?);

— ноль или более раз (*).

Пример использования обозначений



Данное представление эквивалентно следующей записи модели содержания:
<!ELEMENT имя_объекта - - (title,(subtitle? , para | specpara)+)>

Г.2 Пример DTD контекстного типа

Ниже приведен пример DTD контекстного типа, содержащий декларации объектов, необходимых для представления технического описания изделия/подсистемы/узла:

<!-- Для декларации объектов в DTD применяются ниже перечисленные макроподстановки. Используются простые шаблоны, шаблоны альтернатив и шаблоны последовательности -->

```
<!ENTITY % system          "system | system-alts" >
<!ENTITY % descinfo        "descinfo | descinfo-alts" >
<!ENTITY % task            "task | task-alts" >
<!ENTITY % reqcond         "reqcond | reqcond-alts" >
<!ENTITY % inputs          "inputs | inputs-alts" >
<!ENTITY % person          "person | person-alts" >
<!ENTITY % equip           "equip |equip-alts" >
<!ENTITY % refmat          "refmat | refmat-alts" >
<!ENTITY % expend          "expend | expend-alts" >
<!ENTITY % consum          "consum | consum-alts" >
<!ENTITY % alert           "alert | alert-alts" >
<!ENTITY % step            "step | step-alts" >
<!ENTITY % follow-on       "follow-on | follow-on-alts" >
<!ENTITY % partinfo        "partinfo | partinfo-alts" >
<!ENTITY % partbase        "partbase | partbase-alts" >
<!ENTITY % connection      "connection | connection-alts" >
<!ENTITY % attach-part     "attach-part | attach-part-alts" >
<!ENTITY % location        "location | location-alts" >
```

```
<!ENTITY % faultinf          "faultinf | faultinf-alts" >
<!ENTITY % test              "test | test-alts" >
<!ENTITY % outcome           "outcome | outcome-alts" >
<!ENTITY % fltstate          "fltstate | fltstate-alts" >
<!ENTITY % fault             "fault | fault-alts" >
<!ENTITY % rect              "rect | rect-alts" >
<!-- Данная декларация представляет собой корневой объект технического описания -->
<!ELEMENT techinfo  - -  ( version+, (%system;)+ ) >
<!ATTLIST techinfo              %a.node;>
<!-- Объект system определяет иерархию систем, подсистем и узлов, на базе которых строится иерархия разделов ИЭТР. Объект system должен быть создан для каждого узла описываемого изделия (системы, подсистемы, блока, модуля), для которого имеется какая-либо техническая информация (описательная, процедурно-технологическая, информация по неисправностям или о деталях). -->
<!-- Объект system соответствует простому шаблону и имеет связи с другими объектами, содержащими в том числе описательную информацию (descinfo), информацию об эксплуатационных процедурах (task), составе изделия (partinfo) и информацию по поиску неисправностей (faultinf). -->
<!ELEMENT system - - (link*, (%system;)*,(%descinfo;)*, (%task;)*, (%partinfo;)*, (%faultinf;)*)>
<!ATTLIST system        %a.node;>
<!-- Объект system-alts соответствует шаблону альтернатив и содержит информацию, необходимую для контекстной фильтрации данных об изделии -->
<!ELEMENT system-alts - - ( system )+ >
<!ATTLIST   system-alts %a.node-alts; >
<!-- Объект descinfo соответствует простому шаблону и содержит информацию общего, непроцедурного характера, такую как принцип или схема действия. -->
<!ELEMENT descinfo          - - (link*, paraseq) >
<!ATTLIST descinfo              %a.node; >
<!-- Объект descinfo-alts соответствует шаблону альтернатив и содержит информацию, необходимую для контекстной фильтрации данных. -->
<!ELEMENT descinfo-alts  - - ( descinfo )+ >
<!ATTLIST   descinfo-alts  %a.node-alts; >
<!-- Объект task определяет процедуру обслуживания, например демонтажа, ремонта, замены, испытания, регулировки и т. д., ассоциированную с компонентом изделия". -->
<!ELEMENT task - - (link*, (%inputs;),step-seq, (%follow-on;)*) >
<!ATTLIST task          %a.node; >
<!-- Объект task использует простой шаблон, имеет реляционные связи с другими информационными объектами и содержит исходные условия, необходимые для начала выполнения задачи, предупредительные сообщения (предостережения, предупреждения, примечания), последовательность операций в рамках технологической процедуры и перечень последующих операций, которые должны быть выполнены после завершения работы над задачей.-->
<!ELEMENT task-alts - - ( task )+ >
<!ATTLIST task-alts  %a.node-alts; >
<!-- Данный объект соответствует шаблону альтернатив и содержит информацию, необходимую для контекстной фильтрации данных о выполняемых процедурах. -->
<!-- Объект inputs соответствует простому шаблону. Определяет все исходные условия в части подготовки и оборудования рабочего места, которые необходимо выполнить перед началом выполнения задачи. Содержит данные по персоналу, расходуемым материалам, оборудованию и условиям, необходимым для выполнения задачи. Имеет связи с другими объектами.-- >
<!ELEMENT inputs - - (link*, (%alert;)*,(%reqcond;)*, (%person;)+, (%refmat;)*, (%equip;)*, (%expend;)*,
                      (%consum;)* ) >
<!ATTLIST inputs        %a.node;>
<!-- Объект inputs-alts соответствует шаблону альтернатив и содержит информацию, необходимую для контекстной фильтрации данных о требуемых ресурсах -->
<!ELEMENT inputs-alts     - - ( inputs )+ >
<!ATTLIST inputs-alts %a.node-alts; >
<!-- Объект reqcond определяет исходные условия, необходимые для проведения технического обслуживания (например самолет безопасен для проведения ТО). Также определяет задачи, шаги или операции, которые необходимо выполнить для достижения требуемого условия. Объект reqcond соответствует простому шаблону, содержит связи и текстовые примитивы, описывающие условия проведения обслуживания, перечень задач или операций, инструкции по анализу выполнения условий проведения обслуживания -->
<!ELEMENT reqcond - - (link*, (%text;)?, ( %task; | %step; )+,assertion*) >
<!ATTLIST reqcond       %a.node; >
```

```
<! -- Объект reqcond-alts reqcond-alts соответствует шаблону альтернатив и содержит информацию необходимую для контекстной фильтрации данных об исходных условиях -->
    <!ELEMENT reqcond-alts - - ( reqcond )+ >
    <!ATTLIST   reqcond-alts %a.node-alts; >
<!-- Следующие объекты содержат справочную информацию и данные о расходных материалах, необходимых для выполнения задачи. -->
    <!ELEMENT refmat - - (link*, (%text)? ) >
    <!ATTLIST refmat   %a.node; >
    <!ELEMENT refmat-alts   - -   ( refmat )+ >
    <!ATTLIST   refmat-alts %a.node-alts; >
    <!ELEMENT expend - - (link*,(%partbase)?,(%consum;)*) >
    <!ATTLIST  expend %a.node; >
    <!ELEMENT expend-alts   - -   ( expend )+ >
    <!ATTLIST   expend-alts %a.node-alts;    >
<!-- Объект person используется для указания количества и квалификации персонала. Атрибут type простого шаблона используется для указания квалификации специалистов. Атрибут quantity определяет количество специалистов данной специальности --->
    <!ELEMENT person - -   (link*, (%text;)?) >
    <!ATTLIST   person %a.node;
                quantity  NUTOKEN  #IMPLIED >
<!-- Объект person-alts соответствует шаблону альтернатив и обеспечивает контекстную фильтрацию данных по персоналу -->
    <!ELEMENT person-alts   - -   ( person )+ >
    <!ATTLIST   person-alts  %a.node-alts; >
<!-- Объект equip указывает перечень и количество единиц оборудования, необходимых для выполнения задачи. Объект equip соответствует простому шаблону. Атрибут quantity определяет количество единиц оборудования, необходимого для выполнения задачи. -->
    <!ELEMENT equip - - (link*, (% equip;)*, (%text)?, (%partbase;) ) >
    <!ATTLIST   equip     %a.node;
                quantity      NUTOKEN          #IMPLIED >
<!-- Объект equip-alts соответствует шаблону альтернатив и обеспечивает контекстную фильтрацию данных по оборудованию -->
    <!ELEMENT equip-alts - -     ( equip )+ >
    <!ATTLIST   equip-alts %a.node-alts;    >
<!-- Объект consum соответствует простому шаблону. Объект consum содержит данные о необходимых расходных материалах и их требуемом количестве (quantity) для выполнения задачи. -->
    <!ELEMENT consum  - -  (link*, (%partbase;)?, (%consum;)*) >
    <!ATTLIST consum    %a.node;
              quantity        NUTOKEN   #REQUIRED
<!-- Объект consum-alts соответствует шаблону альтернатив и обеспечивает контекстную фильтрацию данных по расходным материалам -->
    <!ELEMENT consum-alts  - -   ( consum )+ >
    <!ATTLIST consum-alts %a.node-alts; >
<!-- Объект alert указывает, что выполнение задачи может быть связано с опасностью. Атрибут type определяет вид сообщения: предупреждение (warning), предостережение (caution), примечание (note). Объект alert соответствует простому шаблону. Объект alert содержит связи, текст сообщения и, при необходимости, графические изображения, выводимые на экран. -->
    <!ELEMENT alert    - - (link*,(%text;)+,(%graphic;)*) >
    <!ATTLIST alert    %a.node;>
<!-- Объект alert-alts соответствует шаблону альтернатив и обеспечивает контекстную фильтрацию данных -->
    <!ELEMENT alert-alts     - - ( alert )+ >
    <!ATTLIST   alert-alts %a.node-alts; >
<!-- Операции (step) являются основными компонентами последовательной технологической процедуры обслуживания. Они описывают действия, которые необходимо предпринять для того, чтобы успешно выполнить поставленную задачу. Объект step соответствует простому шаблону. Объект step содержит связи, сообщения, данные о последовательности переходов внутри операции -->
    <!ELEMENT step   - - (link*,(%alert;)*,(%primitive;)*, step-seq?) >
    <!ATTLIST step       %a.node; >
<!-- Объект step-seq соответствует шаблону последовательности. Дает возможность описывать последовательности операций.-->
    <!ELEMENT step-seq - - ( step)+ >
    <!ATTLIST step-seq    %a.node-seq;   >
```

```
<!-- Объект step-alts соответствует шаблону альтернатив. Обеспечивает контекстную фильтрацию данных.-->
    <!ELEMENT step-alts      - -   ( step)+ >
    <!ATTLIST   step-alts   %a.node-alts;   >
<!-- Перечень завершающих действий является необходимым условием технического обслуживания, которое
должно быть выполнено после выполнения задачи. Например при выполнении задачи для смены какой-то
детали было нужно снять какую-то панель. Эта панель должна быть возвращена на место в ходе завершающих
действий. Эта работа может быть выполнена сразу после завершения ремонта или обслуживания, но она же
может быть выполнена и позже, так как может возникнуть необходимость выполнения еще каких-то техноло-
гических операций в этом же самом месте по обслуживанию оборудования при снятой панели до выполнения
завершающей операции. -->
<!-- Объект follow-on соответствует простому шаблону. Содержит реляционные связи, текст, описывающий
условия выполнения завершающего действия, инструкции по выполнению завершающего действия. -->
    <!ELEMENT follow-on   - - (link*, (%text;)?, ( ( %task; | %step; ), assertion* )) >
    <!ATTLIST follow-on %a.node; >
    <!ELEMENT follow-on-alts   - - ( follow-on )+ >
    <!ATTLIST follow-on-alts %a.node-alts; >
<!-- Объект partinfo содержит информацию о деталях. Каждый объект partinfo связан с объектом partbase (дан-
ные о составе изделия). -->
<!-- Объект partinfo соответствует простому шаблону. Содержит список предусловий, реляционных связей с
объектами, описывающими соединительные, прикрепляемые или навесные детали, обозначение детали, гра-
фическое изображение, месторасположение-->
    <!ELEMENT partinfo  - -  (link*, (%partinfo;)*, (%partbase;)+,(%connection;)*, (%attach-part;)*,(%text;)?,
                             (%graphic;)*, (%location;)* ) >
    <!ATTLIST partinfo     %a.node;
<!-- Объект partinfo-alts соответствует шаблону альтернатив. Обеспечивает контекстную фильтрацию данных о
детали -->
    <!ELEMENT partinfo-alts      - -    ( partinfo )+ >
    <!ATTLIST   partinfo-alts %a.node-alts;       >
<!-- Объект partbase соответствует простому шаблону. Содержит данные о составе изделия с точки зрения
системы материально-технического обеспечения. -->
    <!ELEMENT partbase     - - (link*, (%partbase;)*, (%text;)?, (%location;)* ) >
    <!ATTLIST  partbase  %a.node;
<!-- Объект partbase-alts соответствует шаблону альтернатив. Содержит данные, необходимые для контекстной
фильтрации данных о составе изделия.. -->
    <!ELEMENT partbase-alts      - - ( partbase )+ >
    <!ATTLIST   partbase-alts %a.node-alts;    >
<!-- Объект connection использует простой шаблон. Данный объект указывает, что в системе присутствует
соединение между деталями (узлами), описанными при помощи partinfo -->
    <!ELEMENT connection  - - (link*, (%partinfo;)+ ) >
    <!ATTLIST connection    %a.node; >
    <!ELEMENT connection-alts      - -     ( connection )+ >
    <!ATTLIST   connection-alts %a.node-alts;   >
<-- Объект attach-part соответствует простому шаблону. Объект указывает перечень деталей, присоединяемых
к детали partinfo. -->
    <!ELEMENT attach-part - -    (link*, (%partinfo;)+ )>
    <!ATTLIST attach-part   %a.node; >
    <!ELEMENT attach-part-alts      - - ( attach-part )+ >
    <!ATTLIST   attach-part-alts %a.node-alts; >
<-- Объект location соответствует простому шаблону и содержит информацию о местоположении детали (узла)
в координатах x, y, z. ->
    <!ELEMENT location - -       (link* )>
    <!ATTLIST   location %a.node;
                location-x   NUTOKENS       #IMPLIED
                location-y   NUTOKENS       #IMPLIED
                location-z   NUTOKENS       #IMPLIED>
    <!ELEMENT location-alts  - -     ( location )+ >
    <!ATTLIST   location-alts %a.node-alts;      >
<!-- Объект faultinf соответствует простому шаблону и содержит информацию (перечень испытаний и провер-
ок) по отысканию неисправностей. Объект faultinf можно использовать как для динамических моделей поис-
ка неисправности, так и для «статических деревьев» поиска неисправностей -->
```

```
<!ELEMENT faultinf - - (link*, (%test;)+, (%fault;)*)>
<!ATTLIST   faultinf %a.node;>
<!ELEMENT faultinf-alts  - - ( faultinf )+ >
<!ATTLIST   faultinf-alts  %a.node-alts; >
```

<!-- Объект **test** соответствует простому шаблону и определяет последовательность действий, необходимых для получения одного из возможных результатов тестирования (outcome). -->

```
<!ELEMENT test      - - (link*,(%task;), (%outcome;)+)>
<!ATTLIST   test  %a.node;>
<!ELEMENT test-alts - -    ( test )+ >
<!ATTLIST   test-alts %a.node-alts; >
```

<!-- Объект **outcome** соответствует простому шаблону. Объект **faultstate** содержит описания предположительных неисправностей. -->

```
<!ELEMENT outcome      - - (link*,((%fltstate;) ( ( %test; | %fault; ), (%fltstate;)? ) ) )
>
<!ATTLIST   outcome  %a.node;>
<!ELEMENT outcome-alts  - -  ( outcome )+ >
<!ATTLIST   outcome-alts  %a.node-alts; >
```

<!-- Объект **fltstate** определяет набор предполагаемых неисправностей. Каждая предполагаемая неисправность имеет соответствующий вес, основанный на вероятности того, что именно данная неисправность имеет место. -->

```
<!ELEMENT fltstate - - (link*, (%fault;)+ ) >
<!ATTLIST   fltstate  %a.node;
                  weight      NUTOKENS        #IMPLIED>
```

<!-- Объект fltstate использует простой шаблон. Атрибут **type** определяет тип предполагаемой неисправности. -->

```
<!ELEMENT fltstate-alts   - - (fltstate )+ >
<!ATTLIST   fltstate-alts  %a.node-alts; >
```

<!-- Объект fault определяет причину неисправности (системы). -->
<!-- Объект fault соответствует простому шаблону. Объект **rect** содержит описание задач по устранению неисправности. Объект **system** обеспечивает обратную ссылку на отказавшую деталь.. -->

```
<!ELEMENT fault  - -  (link*, ( %rect; | %fltstate; )+, (%system;)+ ) >
<!ATTLIST fault       %a.node;>
<!ELEMENT fault-alts   - -  ( fault )+ >
<!ATTLIST   fault-alts %a.node-alts;      >
```

<!-- Объект **rect** соответствует простому шаблону. Объект **system** указывает узел(деталь), который(ая) должен(на) быть отремонтирован(а). Объект **test** содержит проверочные тесты, которые требуются для завершения работ. -->

```
<!ELEMENT rect  - -   (link*, (%task;)+, (%fault;)+, (%system;), (%test;)* ) >
<!ATTLIST   rect  %a.node;>
<!ELEMENT rect-alts   - -  ( rect )+ >
<!ATTLIST   rect-alts    %a.node-alts;  >
```

ПРИЛОЖЕНИЕ Д
(справочное)

Библиография

[1] ИСО 8879—86* Обработка информации. Текстовые и офисные системы. Стандартный обобщенный язык разметки (SGML)

[2] ИСО/МЭК 10744—98* Информационная технология. Язык структурирования гипермедийной информации на основе разметки по времени (HyTime)

[3] REC-CSS2-19980512** Cascading Style Sheets, level 2

---

* Оригиналы международных стандартов ИСО/МЭК — во ВНИИКИ Госстандарта России.

** Данный документ доступен в Интернете по адресу: **http://www.w3c.org.**

УДК 656.072:681.3:006.354 ОКС 25.040.40 П87 ОКСТУ 4002

Ключевые слова: интерактивное техническое руководство, техническое руководство, эксплуатационная документация, электронное руководство, электронный технический документ

РЕКОМЕНДАЦИИ ПО СТАНДАРТИЗАЦИИ

Информационные технологии поддержки жизненного цикла продукции

ИНТЕРАКТИВНЫЕ ЭЛЕКТРОННЫЕ ТЕХНИЧЕСКИЕ РУКОВОДСТВА

Требования к логической структуре базы данных

Р 50.1.030—2001

БЗ 12—2000/26

Редактор *В. П. Огурцов*
Технический редактор *Л. А. Гусева*
Корректор *Н. И. Гаврищук*
Компьютерная верстка *А. П. Финогеновой*

Изд. лиц. № 02354 от 14.07.2000. Сдано в набор 21.08.2001. Подписано в печать 09.10.2001. Формат 60×84$^1/_8$.
Бумага офсетная. Гарнитура Таймс. Печать офсетная. Усл. печ. л. 4,18. Уч.-изд. л. 3,85.
Тираж 526 экз. Зак. 1895. Изд. № 2766/4 С 2317.

ИПК Издательство стандартов, 107076, Москва, Колодезный пер., 14.
http://www.standards.ru e-mail: info@standards.ru
Набрано в Калужской типографии стандартов на ПЭВМ.
Калужская типография стандартов, 248021, Калуга, ул. Московская, 256.
ПЛР № 040138