Жаринов В.Н.

НЕАВТОМАТИЗИРОВАННАЯ ВИЗУАЛИЗАЦИЯ АЛГОРИТМОВ

Описание задачи (текстовая часть)

Версия 11.1

ВВЕДЕНИЕ В ДОКУМЕНТ

В1. Общие положения

1. Файл содержит выполняемый автоматизированным способом (в форме машинного оригинала МО) [беловик|черновик] целевого документа или его части (неотъемлемой), выделенной для удобства работы.

Документ в целом, кроме основного содержания, может включать приложения. Содержание документа, приложения (его выделенной части) составляют текст и/или иллюстрации (графчасть).

Конкретное наполнение файла определяется по его имени (полный формат имен см. шаблон документа)[1].

2. Содержание документа, приложения подразделено на структурные элементы по иерархии; её высшие 4 уровня стандартны. Элементы обычно имеют многоуровневую нумерацию и заголовки-абзацы, входящие в оглавление; возможны также элементы без нумерации, в т.ч. не входящие в оглавление, в т.ч. с заголовками в тексте.

В тексте применяются типовые приёмы оформления, описанные в п/р 1.1 документа|шаблона.

3. В файл части из документа, приложения выделяется элемент структуры стандартного уровня иерархии (или ряд соседних элементов одного уровня) целиком (с заголовками).

Для многофайлового МО в имени каждого файла указаны индексы входящих элементов (формат: разделы <ЧN>, подразделы <пPNN>, пункты <ПNNN>, подпункты <пПNNNN>); файл первой части является *головным*.

При наличии приложений их форму (способ выполнения) указывают в отметках о наличии в составе единственного (или головного) файла основного документа (виды способов и формат отметок см. шаблон).

Приложения в МО могут выполняться как отдельные файлы *ПрилN* (что указывается в их отметках о наличии).

При наличии иллюстраций в документе, приложении (части) они также м.б. выполнены разными способами. Подрисуночные подписи включаются в оглавление для удобства поиска рисунков в документе.

Иллюстрации в МО могут содержаться в отдельном файле графчасти *Рисунки*; тогда текст содержится в файле *Текст*, и в нём дублируется подпись к каждой иллюстрации по месту её упоминания для отсылки к графчасти.

4. Оригинал документа (части) выполнен как настоящий файл (имя см. поле внизу) и другие необходимые (детальный состав многофайлового документа см. п. 1.1.4 в <настоящем файле|головном файле *Ч.1 Введ.*>).

Текст подготовлен в среде OpenOffice.org 2.4.0 Writer или иной программы, совместимой по файлам; иллюстрации выполнены в той же программе и/или иными средствами, включая захват машобразов для МО.

Подлинник выполняется как твёрдая копия с заменой и/или добавлением листов к твёрдой копии предыдущих версий, либо как электронный образ файлов оригинала по листам, с которого делаются твёрдые дубликаты.

Информация, содержащаяся в документе, получена из открытых источников, рассматриваемых автором как надежные. Возможное наличие секретных, конфиденциальных, а равно иных сведений ограниченного доступа следует рассматривать как результат предположения на массивах открытых сведений. Имея в виду возможные человеческие и технические ошибки, автор не может гарантировать абсолютную точность и полноту приводимых сведений, и не несет ответственности за возможные последствия, связанные с их использованием.

---

[1] Переменные части текста даются как поля в '< >', заменяемые на описание; общая часть (корень) поля пишется как есть, а изменяемые части как '*'. Файлы МО с однокоренным именем относятся к одному элементу структуры.

[2] Федеральный закон № 230-ФЗ от 18 декабря 2006 г.

[3] Если цитата состоит полностью из сведений, цитирующих другой источник – дать ссылку на первоисточник.

В2. Назначение, сведения о версиях, обозначения и сокращения

В2.1. Документ содержит <указать содержание>.

Документ предназначен для самостоятельной работы по темам входящих заданий.

В2.2. Версии документа выпускаются по мере обновления содержания цикла.

В2.2.1. Текущая версия черновика используется как источник беловика документа (в виде файла Writer с именем, включающим определённый номер редакции, или части такого файла при вёрстке) в составе готовых материалов; из беловика удаляются незавершённые элементы (разделы и пр.), ограничения для черновика и настоящий пункт, после чего оглавление пересобирается.

При необходимости работа над целевым документом/приложением (выделенной частью) отражается в служебном документе – плане (МО файл с приставкой <План>).

В2.3. Об основных терминах данного документа см. п. 1.3.1.

В2.3.1. В **тексте** документа применяются следующие приёмы оформления.

В2.3.1.1. Чтобы у***порядочить работу с материалом***, часть абзацев имеет особые стили:

- элементы перечисления оформляются как пункты маркированного списка;

Без красной строки оформляются абзацы, отбиваемые в объёмном тексте мысли для удобства чтения (за первым), а также вводные положения к крупному элементу (под заголовком).

Формулы обычным текстом даются центрованно отдельных строках

> Такими абзацами оформлены фрагменты, на которые Вам следует обратить особое внимание (узловые моменты текущего пункта или наиболее важные выводы их него).
>
> **Внимание**: [особое указание, требование, необходимое условие для применения пункта]
>
> Эта информация дается сразу же после того места основного текста, к которому она относится.
>
> – так выделяется пункт перечня, требующий особого внимания;
>
> Так выделяется формула обычным текстом, требующая особого внимания

Пример. Такими абзацами выделяются примеры, иллюстрирующие текущую мысль основного текста. Так же выделяются практические рекомендации, советы, указания.

– так в тексте примера выделяется пункт перечня;

Так в тексте примера оформляется абзац продолжения текущей мысли.

Абзац с отступом и уменьшенным шрифтом выделяет в тексте документа составляющие развития, которые дополняют (уточняют, конкретизируют) содержание основного текста.

– так в тексте развития выделяется пункт перечня;

Так в тексте развития оформляется абзац продолжения текущей мысли.

Так в тексте развития выделяется формула

Информация к размышлению. [подзаголовок статьи]. Таким образом в тексте выделена познавательная составляющая, которая не является обязательной для изучения, но может расширить и углубить понимание предмета.

- так в тексте отступления выделяется пункт перечня;

Так в тексте отступления оформляется абзац продолжения текущей мысли.

В2.3.1.2. С той же целью фрагменты в тексте могут оформляться в следующих стилях:

Жирным шрифтом выделены названия отдельных пунктов, уровни классификации или комментарии в тексте к элементам схем, диаграмм.

Курсивом выделяются понятия, определяемые в тексте документа, а также предложения, содержащие важную информацию (выводы, указания и пр.). В списке литературы курсивом выделены позиции, которые имеются в [учебной|служебной] библиотеке.

Жирным курсивом выделены подуровни классификации либо понятия, о которых идет речь в окружающем тексте, или которые уже должны быть Вам известны.

Подчеркиванием обозначаются ссылки на место в данном документе или за его пределами, напр., на другие документы (кроме гиперссылок на ресурсы интернет, которые оформляются стандартно для

инфордоков). Если подчеркнута ссылка на другие дисциплины, сферы деятельности, то Вы можете обратиться за информацией к соответствующим специалистам, преподавателям, в интернет.

Разрядкой выделены места, на которые следует обратить особое внимание.

Таким начертанием (гарнитурой) шрифта и курсивом выделены наименования объектов (сущностей), описываемых в документе.

Такой гарнитурой шрифта (с уплотнением) выделены тексты процессов (алгоритмов, программ).

Так выделяется внутритекстовый заголовок части пункта (подпункта). Далее начинается собственно текст этой части. Такой заголовок не входит в оглавление документа.

В2.3.2. В **графической части** документа используются стандартные стили текста и условные обозначения, приведённые далее (см. п. 1.3.1).

Иллюстрации упорядочены по тексту и снабжены подписями вида:

Так оформляется подрисуночная подпись

В файле выделенного текста эти подписи указывают наличие и положение иллюстраций.

> **Внимание**: отдельные подписи могут размещаться не под, а над рисунком. В любом случае подпись относится к тому рисунку, к краю которого она расположена вплотную.

В2.3.3. В тексте **черновика** (**плана**) применяются также иные приёмы оформления:

Такими абзацами (уровня 6 в иерархии элементов текста) оформляется краткое содержание текста текущего элемента структуры в черновике (плане) .

[**Замечание.**]Так оформляются замечания, указания по содержанию текста.

В2.3.4. Элементы структурированного содержания документа (далее — *докэлементы*) помечены заголовками-абзацами, как правило, нумерованными; иерархия из 4-х уровней.

Для представления структур содержания с большим числом уровней в документ дополнительно вводится уровень надзаголовков докэлементов вида:

[N]<L>. <Текст надзаголовка>

Здесь: L - литер надуровня структуры; N - номер надуровня.

Для удобства чтения литеры чередуются с номерами; также отдельные уровни индекса могут браться в скобки.

В2.3.5. Фрагменты (слитные группы докэлементов) данного документа могут рассматриваться как содержание других документов (полный текст или извлечения). Тогда он является т.н. *сводным документом* – включающим докэлементы других (в своём окончательном виде - отдельных) документов как свои докэлементы – *сводные части*.

Пример. Черновой документ (план) м. б. сводным из частей для разных целевых документов, совместно разрабатываемых по единому замыслу, а затем разделяемых.

Другой пример. В беловом документе могут присутствовать части, которые исходя из назначения документа должны выделяться в самостоятельные целевые документы.

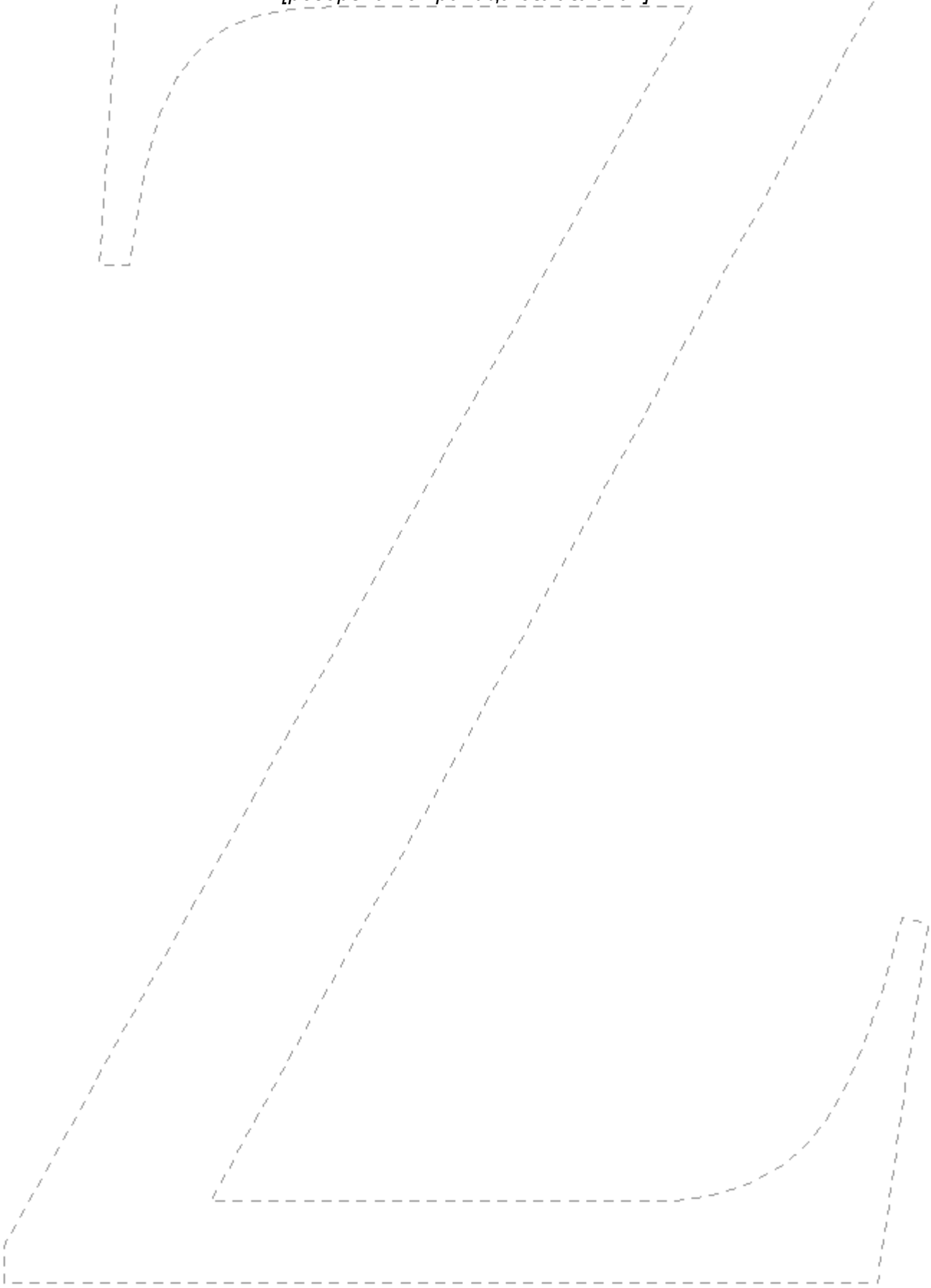
> Допускается вложение сводных частей для разных документов друг в друга.
>
> В сводную часть черновика могут входить сводные части беловика. Заголовки и надзаголовки последних находятся в границах первой.
>
> Границы сводной части в сводном документе указываются особыми абзацами-комментариями уровня 5 (входящими в оглавление); в тексте этих комментариев определяется и соответствие между частью и отдельным документом, приложением, в который она выделяется из сводного документа, приложения.

Для примера начало данного раздела оформлено как сводная часть; предполагается, что она входит в любой документ (или в его часть, приложение, используемые самостоятельно) из шаблона. Указатель конца части находится прямо под этим абзацем.

---конец служебной части (введения в документ) для документа/шаблона---

Оглавление

---[резервная страница оглавления]---



1. ВВЕДЕНИЕ

Внимание: следует хорошо изучить этот раздел, чтобы ориентироваться в документе.

1.1. Общие положения

1.1.1. В документе излагается суть произвольной задачи и процесс решения этой задачи (технология формализации знаний о решении).

Документ предназначен для включения в курс лекций по информатике.

1.1.2. Задача поставлена в конкретном виде и описана в свободной форме (как Разд. А), а техпроцесс формирования решения — как базовая ТФЗ.

1.1.3. Документ подготовлен с использованием источников информации, указанных в Разд. В. Необходимая информация содержится также в приложениях к документу:

1.1.4. Основное содержание документа в оригинале выполнено как настоящий файл.

Приложения к документу выполнены как отдельные файлы (см. отметки о наличии).

1.2. Необходимые определения

1.2.1. Основные текстовые и графические термины, обозначения и сокращения данного документа введены в п/р 1.3, а другие необходимые — в тексте п/р 1.4 цикла, а также определяются в его тексте и визуально на рисунках по ходу изложения.

1.2.2. Здесь расшифрованы сокращения, часто употребляемые в тексте документа.

Сокращения, употребляемые лишь в отдельных местах текста, расшифровываются там же.
Сокращения, значащие одно и то же (напр., на разных языках), отсылают друг к другу.

БНФ	Бэкуса-Наура формы <определения синтаксиса текстов>
ГСА	граф-схема алгоритма
ДРАКОН	Дружелюбный Русский Алгоязык, Который Обеспечивает Наглядность

В тексте документа употребляются следующие типовые обозначения и сокращения:

англ.	английский;
букв.	буквально;
в т.ч.	в том числе;
и т.д.	и так далее;
и т.п.	и тому подобное;
к.-л.	какой-либо;
напр.	например;
см.	смотри;
т.е.	то есть;
т. зр.	точка зрения;
т.о.	таким образом;
разд.	раздел (документа);
п/р	подраздел (документа);
п.	пункт (документа);
п/п	подпункт (документа);

<1>А. Введение в сущности задачи

Содержание данного раздела пополняется и изменяется в ходе жизненного цикла задачи (по мере выполнения фаз, этапов и шагов Разд.Б.).

В разделе описаны внутренние и внешние сущности (понятия, объекты, явления) предметной области задачи, а также существенные отношения между ними (логические, физические).

При компоновке (вёрстке) дракон-схем на диосцене рекомендуется следовать принципам эргономичной организации, которые изложены создателем техноязыка, в частности, в теме Эргономическая оптимизация дракон-схем конференции Оберон.

Вёрстка графоэлементов исходит из некоторых общих закономерностей, которые для дракон-схем вкратце описаны в /3, п. 2.2.1/.

Возможности графического редактирования для заданного офисного пакета описаны в шаблоне графчасти (файл *...\Шаблоны\[имя-задачи]\ГрафЧасть А2 Задача [род] [имя] (Бланк с оргметодчастью)*).

<1>Б. Жизненный цикл решения задачи

I. СОЗДАНИЕ/СОВЕРШЕНСТВОВАНИЕ РЕШЕНИЯ

1. Постановка

1.1. Определение цели решения

Исходно считаем, что решается задача оформления дракон-схемы как машинного документа в среде офисного пакета, включая получение твёрдой копии на стандартном принтере. Предполагается, что оформляемое решение задачи уже доведено до алгоритма, т.е. автор схемы предварительно выбрал диоформу визуала и знает, из каких базовых конструкций составить шампур(ы) будущей дракон-схемы); эти данные (назовём их *ПредстРешения*) считаются результатом предшествующей задачи алгоритмизации.

На этой задаче иллюстрируем как правила языка и его возможности, так и технологию формализации с помощью ДРАКОНа.

1.2. Подбор данных

Для решения задачи необходимы данные, указанные в Разд. А, а также найденные самостоятельно в иных источниках. Наряду с этим используются знания исполнителя задания, отчуждаемые им по необходимости и возможности в результирующий документ.

Правовой статус указанных данных предварительно определяется авторами, выражение воли которых можно найти, в частности, в следующих источниках:

- Пароджановым В.Д. – в сообщении относительно прав на техноязык на конференции Оберон;
- Жариновым В.Н. – в п. 5 Разд. Общие положения Введения в документ.

Помимо того, п.1 Ст.1274 Ч.4 ГК РФ и другие положения законодательства России разрешают использование содержания любых правомерно опубликованных произведений, но при определённых условиях (в частности, для учебных целей).

Т.о. предварительно эти данные можно считать общедоступными для целей данной задачи. Окончательно вопросы решаются с правообладателями. При этом для свободно распространяемых данных следует учитывать потенциальное возникновение т.н. самозваных правообладателей, использующих лазейки в юридическом статусе таких данных по тому или иному законодательству, чтобы заявить те права (прежде всего имущественные), которые принципиально не заявили авторы (подобная ситуация возникала, напр. со свободным ПО, как-то ОС семейства Линукс).

Сочинитель при необходимости вкратце описывает собственные результаты поиска данных и их включения в задачу.

1.3. Формирование структуры решения

В составе данного шага мы не будем выделять логическое и физическое структурирование; обычно это нужно для задач, где есть сложная структура объектов и процессов, требующая специфического физического воплощения.

Для начала следует конкретизировать формулировку задачи (она у нас весьма общая), определить какие-то условия достижения цели и возможные ограничения. Главная цель этапа – определить конкретное программное, информационное, методическое обеспечение задачи, основные понятия. В данном случае зададимся рядом исходных посылок в текстовой форме:

- формализация задачи проводится на языке ДРАКОН-Алго-1 (без совместного протекания алгопроцессов). Дракон-схему будем изображать на одном листе (назовем его *рабочим листом*); текст вершин излагать в свободной форме;
- используется оператор Комментарий для пояснений по содержанию дракон-схемы;
- при оформлении вводятся пояснения по методу *КогниСтиль*;
- допускается, что в процессе оформления решение уточняется, что ведёт к корректировке состава вершин и/или структуры дракон-схемы;
- методическим обеспечением решения задачи является содержание разрабатываемой дракон-схемы, включая комментарии и пояснения КогниСтиль (т.е. мы фактически делаем алгоритм, определяющий сам себя);
- программное обеспечение создания машинного документа – эледок-процессор *Writer*, а графоэлементов схем – элеграф-редактор *Draw* офисного пакета *OpenOffice.org*;
- чтобы просматривать дракон-схему как одно целое, она рисуется на рабочем листе нужного формата, определяемого с учётом ограничений (в пакете можно задать любой формат);
- для рисования используем заготовки вершин, изображённые в Приложении 1; по необходимости их формат корректируется;
- свойства силуэта используются не только для смыслового структурирования задачи, но и для "нарезки" дракон-схемы на ветки, гарантированно входящие на заданный формат диосцены. Поэтому крупные (по занимаемому месту на листе) смысловые шампуры будем делить на две и более веток; также иногда можно объединять в одной ветке мелкие шампуры (без существенного ущерба для понимания), но мы этого делать не будем;
- при вертикальной компоновке схемы можно смещать положение главных и побочных вертикалей относительно осей симметрии вершин, чтобы полнее использовать площадь диосцены; относительная ширина вершин в вертикалях также может различаться;
- твёрдая копия выполняется сразу по завершении машинного документа. Под *печатным листом* понимается наибольший формат листа, поддерживаемый имеющимся принтером (плоттером);
- считаем, что формат печатного листа меньше, чем требуемый формат диосцены (рабочего листа); т.к. *Writer* не поддерживает автоматическую печать в таком режиме, то надо определить способ вывода дракон-схемы по печатным листам. Заранее принимаем, что формат печатного листа машинописный (А4), т.е. ширина 210 х высота 297 мм. Схема большего формата вписывается в конструктивную сетку с ячейками А4 так, чтобы вершины находились внутри ячеек; перед печатью области изнутри каждой ячейки переносятся на листы промежуточного документа печати.
- информационным обеспечением является формуляр-образец документа (файл шаблона *Writer*) в виде рабочего листа максимально возможного чертежного формата. Примем формат листа А2 (597х420 мм); предполагаем, что тело ветки будет относительно простым, а значит, высота шампура небольшой: поэтому на листе можно размещать не одну схему;

В редакторе Word пакета MS Office сторона листа ограничена 558,7 мм, следовательно максимальный формат близок к А2. Поэтому предельный формат рабочего листа А3

(420x297 мм), для нас он состоит из двух ячеек сетки – машинописных листов (А4х2); возможно также использовать промежуточный формат, близкий к А4х3 (558х297 мм);

- шаблон содержит заготовку-силуэт дракон-схемы, корректируемую по реальному набору веток. Для ориентировки пользователя на лист также наносятся линии конструктивной сетки (будем называть их *линиями стыка*) и связующие реквизиты в колонтитулах. Для текста в вершинах задаётся фиксированное оформление, т.к. *Writer* (и другие приложения пакета) не поддерживает стили текста внутри графоэлементов;

  В Word можно определять стили текста внутри графоэлементов, рекомендуемых отличающихся в основном плотностью размещения;

- к назначению отдельных графоэлементов на шаблоне даются направлённые пояснения КогниСтиль.

Перейдём к следующему этапу фазы I.

Далее мы совместили два этапа (2 и 3) фазы I, поскольку выбор средств решения в основном предопределён постановкой задачи.

2. Эскизная визуализация и выбор средств решения

На эскизном этапе из заданных посылок выводится предварительная дракон-схема или дракон-модель решения. Определяющими факторами являются:

- выбор формы дракон-схемы (мы используем силуэт);
- тип силуэта: конкретный или алгоритм-концепция (выбираем ограниченно конкретный);
- степень обобщения содержания вершин (максимальна, т.к. мы ограничены форматом листа).
- метод визуализации (нестрогий).

Поскольку три фактора из четырёх заданы нечётко, то и весь процесс неформальный, приводящий к разным (но эквивалентным) вариантам графит-схем. В эскизе смысл вершин (заданный их текстом) м.б. обобщённым. Кроме того, мы не будем формулировать смысл информатически строго (только через арифметические и логические выражения от абстрактно типизированных операндов).

Поскольку содержание ряда вершин обобщённое, к ним следует дать разъяснения. В качестве примера это сделано для областей и некоторых виопов.

Разъяснения оформлены как графит-примечания, визуальный синтаксис которых имеет в основе «буксируемую выноску», часто используемую в ГИО геоинформационных приложений. Примечание связано со знаком-управителем, постоянно присутствующим на контуре вершины; он отображает заполненность примечания и служит для его открытия на просмотр/редактирование.

Прежде всего определимся со структурой деятельности.

Импер-моделирование проведём на техноязыке в «родном» диалекте версии ДРАКОН-Алго (ДА-Р); подробнее о них см. /3, Приложение 2, п/р 3.5/.

Характерной чертой ДР-диалекта является информатическая неформальность текста виопов, где действия м.б. сформулированы на естественном языке (структурированном, напр., как показано в /3, п/п 2.1.3.2/ для действий и в /3, п/п 2.1.3.10/ для условий), а именами операндов м.б. нестрого определённые сущности. Определения, разумеется, целесообразно давать вне текста вершин; мы будем использовать как текстовые, так и графит-определения (вторые относятся прежде всего к сущностям шампур-метода и потому даются прямо при ДР-схемах средствами КогниСтиль и графит-динамизации).

Чтобы отразить незавершённость, схемы модели имеют статус сочинения 'draft' (подробнее о статусах см. в конце /3, п/п 2.1.3.4/).

Рассмотрим принятые решения по алгоритмизации. Обобщение текста вершин означает, что нам надо укрупнять операторы построения дракон-схемы; до какого разумного предела это возможно? Ответ подсказывает теория систем, и он использован при формировании графит-метода в /3, п/п 1.4.3.2/ – минимальный базис видов операций структурирования образуют при-

бавление структурного элемента, его вычитание и преобразование структуры связей. Первые виды в нашей постановке сводятся к командам «редактирования по выделению» над фигурами из дракон-алфавита (копирование через «карман» и удаление); в качестве инфобеспечения оправданно создать образцовый файл-каталог вершин и линий, который назовём *АлфКатВиопов*.

Мы имеем дело с уже готовым описанием и потому, как правило, не требуется и удаление элементов; из каталога берётся то, что нужно, и размножается по числу вхождений в схему.

А как определить преобразования? Можно исходить из следующего.

Т.к. мы визуализируем нестрого, т.е. «просто рисуем», то в нашем распоряжении нет аксиоматически заданных формальных структур (заготовок) и подструктур (макровершин); мы просто можем воспроизвести их из графоэлементов офисного пакета (фигур). Поэтому прежде чем преобразовать какую-то схему, мы должны её образовать; встаёт вопрос: по каким принципам это делать? Ответ даёт шампур-метод: сколь бы «либеральной» ни была визуализация, всё-таки нужно следовать неким его базовым правилам, без которых дракон-схема не будет собой; ведь «если можно всё – то нельзя ничего» :).

Прежде всего сохраняются топологические запреты; но нужны и правила-разрешения. Это будут принцип шампура (как упорядочивать вершины при следовании) и принцип главной/побочных вертикалей (как упорядочивать уже цепочки следования друг относительно друга); для силуэта также принцип петли (укладки маршрутов в тела веток).

Итак, операции преобразования включают редактирование геометрии линий (реализует пересадки и заземления лиан) с последующим возможным смещением включённых в соответствующие маршруты вершин; понятно, что связь вершин с линиями существует в сознании сочинителя и никак не определена в файле дракон-схемы.

Кроме того, мы можем «подгонять по месту» графику отдельных вершин; для их текста установим форматы по видам содержания (объекты/процессы), так же, как сделано для текста, который Вы сейчас читаете, и будем считать постоянными (минимально требуемыми для читабельности) межстрочные и иные интервалы; тогда мы можем менять лишь кегль шрифта.

В офисном пакете можно форматировать одну фигуру по образцу другой; при этом переносится вид линий контура, площади, а также формат абзаца (но не знака) текста. Такие преобразования нам при наличии каталога вершин требуются, только если надо установить формат по уже изменённому.

Из посылки об уточняемости решения также следует, что по ходу оформления возможно уточнение дракон-схемы (как правило, не затрагивающее сути представления о решении; для оформления это существенно лишь постольку, поскольку меняется *ПредстРешения*).

Возможный визуал, отражающий наши соображения, приведен в графчасти.

Задача оформления дракон-схем в офисном пакете (эскизная визуализация техпроцесса решения)

Здесь мы преследовали также вспомогательную учебную цель – показать все основные конструкции техноязыка. Поэтому мы и однотипные фрагменты деятельности в разных случаях визуализируем альтернативными конструкциями (напр. оценку понятности алгоритма по полученной дракон-схеме – то переключателем, то развилкой).

Также одна из схем имеет варианты сочинения части содержания; они оформлены как графит-области (определение механизма областей см. /3, п/п 2.1.3.12/).

Также детальные определения операций даны в Разд. В.

Здесь мы положили, что каждая вершина в каталоге нарисована несколько раз разной ширины; однако обратившись к реальному каталогу, Вы обнаружите там вершины только одной ширины. Поэтому выше мы и ввели подгонку вершины; тем самым устраняется «зазор» между принятым определением величины и его реализацией (что бывает и на практике, если исходная модель расходится с её реализацией).

Отметим, что для описания сходных ситуаций использованы разные конструкции: повторное оформление примитива оформлено как веточный цикл, а силуэта – как обычный цикл внутри ветки;

это можно рассматривать как пример использования альтернативных конструкций в сходных ситуациях.

Мы видим, что главный маршрут при выводе дракон-схемы предполагает наличие дополнительных операций, тогда как при оформлении примитива и силуэта, напротив, главные маршруты исключают их. Это также пример возможностей, которые предоставляет ДРАКОН для эргономизации описания: автор схемы, пользуясь правилом главного маршрута, наглядно выражает свое представление о ходе процесса "по умолчанию".

Рассмотрим также некоторые принятые решения по структуре процессов.

Изменение состава вершин и заполнения их текстом оформлено в самостоятельную процедуру Заготовить элементы схемы. Это дало возможность «разгрузить» головной визуал, а также точнее отразить решение задачи; ведь такие же действия нужно выполнять и после прояснения алгоритма.

Вызов визуала Заготовить элементы схемы включён непосредственно в процедуру Прояснить алгоритм. Можно было бы поставить его после этой процедуры в головном визуале, но это не влияет на логику (и так, и так видно, каков порядок действий), а вот объём описания возрастает (и это приходится опять на головной визуал), что может снизить понятность модели (больше объектов — больше нужно читать).

Визуал Вывести твёрдую копию оформлен как силуэт, но особой логической необходимости в выделении веток здесь нет. Видно, что можно перейти к примитиву; он м.б. размещён в высоту рабочего поля листа, а в ширину компоноваться более рационально.

При областной декомпозиции визуала Подготовить вывод мы показали два варианта - с пустой входящей областью и с заполненной — как отдельные схемы (и группы подстановок). Поэтому получилось, что одна из областей дублируется (как входящая во вторую схему и как вынесенная для первой). Конечно, на практике сочинитель выбирает тот вариант, который ему удобнее.

> Возможен вопрос: а как различать, какой из вариантов визуала вызывать? А никак :) - мы это не определили. Строго говоря, это логическая ошибка повторного использования имени (и в автоматизированном редакторе пользователю д.б. на неё указано).
>
> Самое простое — оставить один вариант; но мы создали два, чтобы наглядно показать различие между пустой и заполненной входящими областями. Если не отказываться от этого, то устранить ошибку можно двумя способами. Вариантам можно дать разные имена (скажем, добавить в конце имени номер варианта), а в примечаниях к визуалам указать, зачем варианты нужны. В каждом вызове тогда нужно выбирать имя вставки; это не очень удобно. Напрашивается охват каждого вызова графит-областью в общей группе подстановок и определение замены части имени, определяющей вариант (тогда достаточно менять только условие замены); но это, пожалуй, «автоматизация неэлегантности».
>
> Можно применить область непосредственно ко вставкам. Тогда схема будет одна, а тело схемы м.б. охвачено новой входящей областью целиком (тогда, кстати, можно устранить дублирование виопов, не охваченных прежней областью). Группа подстановок этой области будет определять варианты (если входящая область пустая — то через две вынесенных; но в данном случае удобнее заполненная по одному из вариантов и вынесенная для другого).

Тело головного визуала в целом задаёт жёсткую связку между оформлением схемы (модели) и выводом её твёрдой копии. Чтобы избежать этого (и тем самым придать нашему решению новые функциональные возможности), введём дополнительный вход в визуал на ветку Завершение; очевидно, при входе следует передать как параметр документ с дракон-схемой.

Отметим, что в данном случае сделать это возможно сразу, т.к. нужные действия уже находятся именно в этой ветке; иначе были бы нужны дополнительные преобразования алгоритма.

Допвход условно показан пунктиром.

Состав процессов решения задачи получился небольшим, а структура их взаимодействия не вполне элементарна. Было бы удобно для лучшего понимания обозреть её, так сказать «с высоты птичьего полёта», показав только связи процессов (в нашем случае имеют только один тип — вызова одних процессов другими). Эту возможность нам даёт свёртка дракон-мо-

дели на ДМ-языке. Основные принципы её изложены в /3, п/п 2.1.5.2/; там же даны основные положения для определения языка моделирования.

Схема-свёртка строится как вспомогательная от текущего состояния дракон-модели (схемы). Поэтому она имеет статус моделирования 'auto'.

Моделирование проведено на ДМ-языке, описание которого дано в /3, Приложение 2, п/р 5.4/.

Вызываемая процедура в ДМ-языке представлена как вставка, а собственно виопы Вставка в вызывающих процедурах — совокупностью виопов-границ ветки.

Основные решения по моделированию заданы автоматическим статусом схемы; как вручную, так и автоматически отображается структура вызовов с переносом данных из виопов Вставка, Заголовок и Конец.

При **деклар-информоделировании** исходим из концепции т.н. абстрактных типов сущностей (в программировании — данных); необходимые сведения о ней изложены в /3, п/п 2.1.1.2/. Для моделирования использован АТ-язык, основные положения которого приведены в /3, п/п 2.1.5.1/.

> Запись структурирует предметную сущность как многоуровневое дерево; при этом в гарантоспособных языках принято наращение дерева, т.н. расширение записи, каждый раз только одной записью-элементом (полем). По сути, это тот же подход, что и в обыденной практике многих народов при установлении кровного родства — связь ведётся линейно по одному родителю — позволяющий избежать «комбинаторного взрыва».
>
> Произвольные структуры в информоделировании принято определять через т.н. указатели — ссылки на некоторые другие типы. В нашем случае такой необходимости нет.

Как обычно, дубли глобальных величин, используемых локально (во вставках), показаны на АТ-схемах вставок блёклыми.

Вообще говоря, в АТ-языке определены три вида подстановок: процедур (законченных АТ-схем); величин (АТ-подсхем); типов. Из них только типы могут подставляться также и активно (в смысле /3, п/п 2.1.3.3/); это визуализируется АТ-оператором Вставка типа. При этом схема типа трактуется как образец определения сущности. Пассивные подстановки служат для декомпозиции сложных АТ-определений (при этом получается АТ-модель аналогично дракон-модели в импер-визуализации).

Декомпозиция посредством всех этих вставок, помимо повышения обозримости модели (за счёт восприятия по частям и прослеживаемости многократного использования некоторых определений), позволяет рациональнее скомпоновать модель (единая АТ-схема, как нетрудно понять, занимала бы больший формат — не только из-за дублирования объектов подстановок, но и в силу «разбросанности»).

Пассивные подстановки схем и подсхем вообще-то можно рассматривать как случаи областной декомпозиции; но она в данном случае проста и регулярна, и синтаксис областей здесь был бы избыточно сложен; поэтому в АТ-язык и введены специальные виопы.

Обсудим некоторые решения по деклар-моделированию. Прежде всего отметим, что в дополнение к принципам типизации, установленным в методиках Ляховича (см. /3, Приложение 3, Обозначения и определения, п/р 3.3/), мы руководствуемся здесь ещё одним: *если структура сущности ещё неясна нам, но предполагается сложной — абстрагируем её как тип запись*. При этом точную структуру типа можем определить не сразу, а по мере углубления понимания задачи (и соответственно проработки модели).

Нетрудно видеть, что в частную деклар-информодель включены только сущности, которым можно (при текущем понимании задачи) назначить абстрактные типы. Остальные сущности определены неформально, прежде всего на схемах (средствами КогниСтиль и графит-метода). По сути, можно сказать, что они имеют «неабстрагированный тип», т.е. не типизированы информатически строго.

Часть сущностей и в деклар-информодели определены неформально; это связано с тем, что детали реализации нас не интересуют. Поэтому использованы шампур-заместители с РБНФ-определениями; где нужно, к ним даны разъяснения в форме графит-примечаний.

Для расширяемой записи в АТ-алфавите определён свой вариант графики (в виде «пачки»). Мы указываем на расширяемость не только когда это очевидно (запись имеет поле-запись, как у типа *ФайлСхемы*), но и когда предполагаем такую возможность (как для *ПредстРешения*, определение которой не полностью проработано).

Для определения допвхода головной схемы использовано ОС-расширение, допустимое в графит-методе для любого языка. Показано, что процесс под разным именем использует разные формальные параметры (и получает разные фактические). Правда, т.о. невозможно показать использование сущностей, лежащих внутри области видимости процесса (оно тоже, как правило, будет разным, и в данном случае это так); но это не является задачей деклар-схемы (для отражения этого есть имп-ер-схема).

Некоторые определения вынесены из основной схемы пассивной подстановкой. Можно подставить их обратно, но это, кроме увеличения объёма, усложнит синтаксис схемы.

Все три вида пассивных подстановок, как и вставка типа, представлены в нашей АТ-модели. Основным, как можно видеть, является подстановка подсхем (как величин *ПредстРешения* и *АлфКатВиопов*, так и типа *ФайлСхемы*), обозначенная виопом Гибкое поле. При этом подсхема начинается сразу с первой вершины тела; её заголовок как бы «делегируется» в виде гибкого поля в места вставки, определяемые по совпадению имён. Также широко использована вставка схем процедур, обозначенная виопом Процедура; здесь уже подставляемая схема имеет заголовок. Наконец, посредством активной подстановки типизированы экземпляры файлов документов задачи (в нашем случае - как имеющие один и тот же тип *ФайлСхемы*).

В принципе величины в пределах задачи глобальны; поэтому передачи переменных как параметров между процедурами не требуется. Только для визуала Подготовить вывод мы определили параметры (и соответственно их передачу при вызове из визуала Оформить дракон-схему), чтобы показать их использование (всё-таки у нас учебный пример).

Варианты визуала Подготовить вывод используют разный состав величин; поэтому в его АТ-схеме определена подстановка, взаимосвязанная с ДР-схемой (через общий индекс группы).

Видно, что вариант А предполагает отсутствие дополнительной величины; поэтому шампур входящей области дан пунктиром (реально при выполнении условия подстановки варианта он исключается из секции величин схемы; соответственно вершина-терминал секции укорачивается по правилам метрики АТ-схем).

Пожалуй, в данной АТ-модели из основных возможностей языка не использовано только приравнивание типов (виопом АТ20). В принципе можно сделать и это, но необходимости здесь нет; читатель может реализовать эту возможность сам.

Активная графит-модель описывает систему-решатель задачи. Общая концепция актив-моделирования базируется на опыте автора задания; однако можно сопоставить ей т.н. моделирование реализации в ЯВС-методологии формализации знаний (известна также как автоматное программирование).

В данном случае решатель определён как сосредоточенная СЧМ типа автоматизированного рабочего места.

В рамках графит-метода модель представлена на языке актив-силуэтов (АС). АС-язык описывает многосвязные граф-схемы (сетевой топологии) гармонизированно с принципами шампур-метода (см. в /3, п/п 2.1.3.8/).

АС-схема есть силуэт, повёрнутый (кроме заголовка) так, что АС-ветки (блоки со связями) идут сверху вниз. Блоки понимаются как области; на схеме мы видим вложение (т.е. фактически непустую входящую область *АРМ-Разр*, детализированную на вложенные области *Оператор* и *КСА*; внутри них уже выделены блоки-элементы, понимаемые как неделимые при данной степени подробности описания).

Отметим, что пространство имён едино для всего описания; поэтому большинство блоков и ряд связей именованы так же, как в визуалах и деклар-схемах, что показано стилем знака; остальным именам нет соответствия в частных описаниях, поэтому они не выделены.

Структура решателя описана укрупнённо, связи детализированы на внутренние (замыкаются через АС-петлю) и внешние (присоединяются к вершинам-туннелям), а внутренние — только на команды для КСА и данные (параметры команд и сообщений от КСА). Как принято ещё в шампур-методе, направление связей не показывается; оно естественное для чтения европейского текста (справа налево). Ему соответствует направление обхода АС-петли против часовой стрелки; оно и показано на схеме.

Не раскрыты и внутренние связи блоков сети; для наших целей это не требуется, а необходимое понимание можно извлечь из других частных моделей решения (прежде всего императивной) и собственного опыта.

Так, наши общие знания подсказывают, что общие данные из среды должны влиять на общие интеллектуальные ресурсы оператора и на частные (показанные как конкретные ЗУНы; определение интелресов см. в /3, п. 1.1.2/); но из ДР-модели можно понять, что они также влияют и на представление о решении и на оценку когнитивного качества... и на саму модель решения (в т.ч. как бы «нефор-мально/неявно-рекурсивно» - каждое применение модели может давать новый опыт оформления и потому порождать её вызов уже для оформления самой себя).

Схема представлена так, чтобы строить её путём «исчисления вершин». Для чтения иногда м.б. удобнее менее регулярный порядок блоков и связей. В этом случае можно допустить альтернативное представление АС-схемы, на котором можно перетаскивать блоки; редактор при этом перечерчивает связи, в т.ч. с пересечениями (подобно тому, как сделано в пакетах конструирования оборудования).

Возможны и разные варианты компоновки АС-силуэта с сохранением регулярности. Так, в графчасти представлен (на отдельном листе как форма 2) многоконтурный вариант, при котором разным блокам выделяются секции АС-петли, приобретающей в данном случае конфигурацию «гребёнки».

Обобщённая графит-модель описывает структуру задачи как целостного описания. Она составлена в т.н. компонентной парадигме информоделирования, широко используемой в программировании. Основные принципы её изложены в /3, п/п 2.1.5.2/; там же даны основные положения для определения языка моделирования.

Предполагается, что решение структурировано на компоненты-модули, которые в то же время есть единицы реализации.

Понятие модуля гармонично с актив-знанием, поэтому можно ввести единую идентификацию (адресацию, именование) реализации для обобщённых и частных (прежде всего актив-) моделей.

Моделирование проведено на ПК-языке, описание которого дано в /3, Приложение 2, п/р 5.3/. Оно основано прежде всего на следующих решениях.

Сущности в разных процессах могут иметь одинаковые имена, если определены как локальные (поскольку тогда их области видимости не перекрываются). В описании мы уточняем имена таких сущностей именами процессов (приписываются спереди через двоеточие).

Аналогично уточнены множественные входы визуалов (в данном случае есть только у головного). При этом и визуал, и его допвходы нумерованы для удобства восприятия (ну и также для этого сгруппированы по порядку следования).

В нашем решении мы предположили, что сочинитель и исполнитель для задачи один и тот же, т.е. задача формализуется тем же решателем, который затем применяет результат формализации. Это тражено в совпадении имён сочинителя и исполнителя на верхнем «этаже» ПК-модуля (а также в структуре актив-схемы).

И о некоторых **общих вопросах графит-моделирования**, важных для данной задачи.

Графит-схемы информатизованной части решения логически объединены в *графит-информодель* (*ГИМ*). ГИМ имеет единый индекс, все схемы нумеруются как её части; предполагается, что она не делится между графит-документами (не считая того, что вынесенные области

в принципе м.б. и в других документах, что учитывается при их индексации), а в одном документе м.б. более одной ГИМ.

> В нашем случае графит-информодель только претендует на информатизованность (из-за недоопределённости ряда схем как конечных структур и ряда операторов — как выражений и директив прогязыка). Вообще же полностью определённая ГИМ может трактоваться как представление некоего программного текста. Если транслировать нашу модель *Оформление дракон-схемы* (для чего нужно изменить статус каждой её схемы с 'draft' на 'exec'), то получим текст квазипрограммный, в котором ряд операторов и директив будут иметь атрибутивную часть на естественном языке; такая запись также практикуется в текстовом программировании по методу пошагового уточнения.
>
> Может показаться, что «квазипрограммность» нарушает требования к алгоритмической строгости. На самом деле следует учесть актив-знание о решении; строгость восстанавливается, если считать исполнителем систему «человек-машина», где человек (сочинитель и исполнитель) понимает недоопределённые части описания и постепенно приводит их к «машинным» требованиям.

Для организации как текстового, так и иного содержания на отдельных диосценах применены схемы *графит-синт-языка* (*ГС*). По смыслу это синтдиаграммы, организованные по модифицированному шампур-методу, когда расположение маршрутов изменено на горизонтальное (основные положения см. в /3, п/п 2,1.3.12/).

В отдельных ГС-схемах положение вершин Адрес строки также изменено; представляется, что этот вариант лучше указывает порядок чтения схемы.

ГС-строки использованы в первую очередь для неформального определения сущностей задачи. Кроме того, посредством их даны описания некоторых типов графит-схем.

В ГС-языке реализован также выбор маршрута движения по схеме, т.е. она приобретает смысл сценария развёртки содержания; используется синт-переключатель, имеющий здесь смысл меню маршрутов. В нашей модели так выбирается просмотр вариантов компоновки актив-схемы (оформлены на разных листах).

> Видно, что переход из строки ГС-схемы возможен на строку любой ГС-схемы, а также на иную ссылку — ручную или автоматическую. Тем самым сочинитель не ограничен только маршрутизацией по ГС-строкам (правда, без обратных ссылок извне ГС-схем он д.б. переходить на ГС-строки, листая документ).

Ссылки и их цели (показанные цветом в текстах вершин) определялись следующим образом. Автоссылки должны наводиться всегда из таблицы имён графит-информодели (которую мы не показываем, но подразумеваем её наличие); потому все имена обозначены как цели этих ссылок (выделены зелёным); предполагается, что из таблицы можно перейти к любому вхождению данного имени. Также они наводятся из операторов подстановки (и активной — вызова процедур, и пассивной — простой вставки); поэтому в этих операторах целевые имена обозначены как автоссылки (бирюзовым с подчёркиванием). Ручные ссылки м.б. установлены сочинителем на любой текст; в данном случае выделены (синим с подчёркиванием) отдельные термины, определяемые в модели неформально. Цели этих ссылок — определения терминов (показаны цветом морской волны). Наконец, перекрёстными считаются ссылки на имена в обобщённой ПК-схеме модели (от них к таблице имён и обратно); поэтому они обозначены сине-зелёным с подчёркиванием. Оранжевым обозначаются поля графит-подстановки текстов (в областях); в нашем случае они не определены.

> Вообще говоря, принцип структуризации связей графит-модели через автоссылки предполагается следующий. В таблице имён ведётся перечень автоссылок на текущие вхождения каждого имени, откуда и можно перейти к нужному. Кроме того, из любого оператора вставки любой схемы по имени вставляемого объекта можно перейти к нему (на имя в заголовке) и обратно, т.е. наводится парная перекрёстная ссылка; каждое из имён также является целью автоссылки из таблицы. То же действует для имён в ПК-схеме.
>
> Наша визуализация реализована по ручной технологии (в смысле /3, п/р 4.1/). По-

нятно, что её можно рассматривать как своего рода «макет» документа автоматизированной визуализации; тогда видны некоторые макетные решения. Так, ссылочные связи определены через совпадение текста ссылки и её цели; в машинном документе, конечно, связь по ссылке определяется служебным адресом, а ссылка и цель м.б. любыми текстовыми фрагментами (и постоянное зрительное выделение целей не обязательно — можно ввести режим показа служебных полей).

Помещение всего тела силуэта в одну петлю, как принято в шампур-методе для дракон-схем - не единственный возможный вариант организации. Схема с телом из более чем одного блока м.б. организована так, что разным блокам выделяются секции петли силуэта, приобретающей сложную конфигурацию.

Секции, в свою очередь, м.б. упорядочены вертикально - «этажеркой», или горизонтально - «гребёнкой» - а также могут содержать цепочки блоков.

При внимательном взгляде можно обнаружить и **недостатки** данной модели задачи:

- неявно предполагается, что оформление модели выполняется всегда; единственное указание, что это не обязательно так, можно найти в визуале Заготовить элементы схемы в виде формулировки «... нужных [макро]виопов и линий...» во втором действии - если определение состава и связей в первом действии визуала показало, что в существующем документе нет отличия от *ПредстРешения*, то заготавливать нечего, и можно считать, что далее оформлять тоже (если не возникает необходимость прояснить алгоритм);
- не учтено, что решение содержит не только импер-часть; изменение *ПредстРешения* может влиять также на деклар- и/или актив-знание о задаче;
- в ветках Оформление примитива и Оформление силуэта имеются повторяющиеся по структуре и почти идентичные по содержанию фрагменты;
- в визуале Подготовить вывод нечётко определены условия, требующие подгонки к линиям сгиба, а также порядок вывода листов.

Первое можно устранить, введя явный выбор, во-первых, между оформлением заново и переоформлением текущего документа (который ищется по совпадению имени в заголовке переданного *ПредстРешения* и имени документа среди существующих), а во-вторых, между заготовлением элементов для существующего документа и пропуском этой работы (по результатам сравнения схем в документе и описания в *ПредстРешения*).

В качестве практики по визуализации найдите пути устранения остальных недостатков; попробуйте сделать это прежде, чем продолжить изучение примера.

По сути, мы дали крайне общее описание процесса оформления дракон-схем, которое можно считать визуальной постановкой задачи. При этом разъяснения, данные в виде текста, являются необходимым компонентом описания решения и вместе с дракон-схемой составляют основу для последующей формализации задачи.

Как рекомендовалось по технологии, можно было ввести описания действий в дракон-схему в виде вершин комментария, но у нас получились чересчур большие и «рыхлые» тексты, поэтому мы поступили иначе.

3. Детальная визуализация (задание для самоподготовки)

3.1. Определение формы (разработка интерфейса)

Выбор параметров дизайна интерфейса (вида физических объектов): цветовых сочетаний, шрифтового дизайна, форматирования текста и графики и их отражение в описаниях Разд.А. Определение формы реализации: деление на исполняемые и служебные (конфигурационные) файлы, форматы файлов, что отражается в Разд.А.

3.2. Формализация содержания (рабочая алгоритмизация)

Разработка алгоритма решения на основе модели процесса из этапа 1. Выделение автоматизируемой части (компонентов алгоритма, выполняемых машиной) и ручной (другие компоненты, выполнение которых остаётся за человеком).

Приведённый далее текст этапа 3 используется как основа для самостоятельной работы над детальной дракон-моделью задачи и сопутствующим ей описанием этапа. Выбор параметров интерфейса здесь предопределён форматом графики ДРАКОНа и КогниСтиль в алфавитном каталоге; формой реализации служит документ описания (текстовая часть в этом файле и графчасть в отдельном файле).

На этом этапе устраняются недостатки эскизной и все содержание процесса по возможности отражается в дракон-схеме, а не за её пределами.

Одна из идей по улучшению возникает при сравнении процессов оформления примитива и силуэта. Бросается в глаза, что в них много общего; это вполне естественно, т.к. структурно тело примитива – почти то же самое, что тело ветки силуэта. Разница в том, что если ветка многоадресная, то она имеет не один выход; можно сказать, что ей соответствует «примитив с несколькими концами» – вершинами Адрес. Вопрос: можно ли всё-таки учесть эту разницу и составить инвариантную процедуру, которую можно вставить и в ту, и в другую ветки?

Попробуйте самостоятельно сформулировать такую процедуру.

Из посылки об уточняемости решения можно вывести иное следствие, чем при эскизной визуализации, а именно: уточнение дракон-схемы может приводить к уточнению представления решения. Тогда процесс оформления нужно визуализировать совместно с процессом алгоритмизации; посмотрим, как это можно сделать.

В состав визуалов задачи включаем алгоритм, описывающий процесс алгоритмизации; конечно, в силу неалгоритмизуемости данного процесса для общего случая его содержание можно отобразить лишь в общем виде (эскизно).

> В самом деле, записав укрупнённое действие «построить алгоритм решения поставленной задачи», мы могли бы сначала детализировать его на основе соображений из ТФЗ-ДРАКОН, выбрав род методов построения; далее можно выбрать, напр., среди формальных методов описанный в Приложении 3 комплекс методик Ляховича; затем можно визуализировать содержание этих методик и т.д... но мы всё время будем обнаруживать, что условия выбора маршрутов нестрогие, а ряд ключевых понятий не определяются так строго, как нужно для их использования в алгоритме – поэтому формулировки действий и условий так и останутся «на словах».

Здесь на основе представления исполнителя (человека-алгоритмиста) о решении задачи (определим их как объект *ПредстРеш*) формируется представление алгоритма, входное для оформления (объект *ВхПредстАлг*), и логически структурируется задача, в результате чего определяется структура визуала (определим их как объект *ФормаВиз*, имеющий значения "примитив" и "силуэт"). Т.о. имеем дракон-модель по крайней мере из двух визуалов.

Взаимодействие процессов формализуется так. Визуал Алгоритмизация задачи имеет формальный параметр *ПредстРеш* и вызывается первым с фактическим значением этого параметра, определённым за пределами модели. По завершении работы он вызывает как вставку визуал Оформление дракон-схемы с фактическими параметрами *ВхПредстАлг* и *ФормаВиз*. В процессе оформления исполнитель отвечает на да-нетный вопрос "Изменилось ли <моё> представление о задаче?"; если да, то считается, что порождён объект *УтПредстРеш* и он передаётся как фактический параметр вновь вызываемому визуалу Алгоритмизация задачи; в результате снова порождаются *ВхПредстАлг* и *ФормаВиз*, с которыми вновь вызывается визуал Оформление дракон-схемы, и так пока исполнитель не даст на вопрос ответ "нет", после чего выполняются заключительные действия по оформлению и этот визуал завершается (а с ним и всё решение).

Т.о. образуется ***взаимная рекурсия*** («Алгоритмизация кивает на Оформление, а Оформление — на Алгоритмизацию»), которая разрешается, исходя из мнения исполнителя.

> Конечно, реально представление о решении может измениться в любой момент выполнения – и гораздо раньше места расположения указанного вопроса – тогда формально испол-

нитель идет по визуалу, уже понимая ненужность выполняемых действий, и даже позже – тогда формально исполнитель вообще должен выполнить заключительные действия и начать новый сеанс решения задачи – но это и есть ограничения алгоритмической формализации деятельности.

Процесс оформления структурируем на:

⇒ первоначальный – не зависящий от содержания *ПредстАлг*;

⇒ конкретный – связанный с оформляемым содержанием;

⇒ вывод твёрдой копии дракон-схемы.

Конкретное оформление зависит от значения *ФормаВиз*, т.е. выполняется в вариантах для примитива и силуэта; в ходе его определяется необходимость разделить тот или иной шампур на "физические" ветки (если изначально оформляется примитив, то в этом случае переходим к варианту силуэта).

В нашем случае возможным результатом будет дракон-модель, представленная далее (см. графчасть).

Эту модель нужно составить самостоятельно.

Рабочая визуализация техпроцесса оформления дракон-схем

У данного решения всё равно есть недостаток: мы предполагаем, что входное представление состоит из единственного алгоритма, тогда как в ходе алгоритмизации может получиться и дракон-модель, и нужно будет оформлять два и более визуалов.

В качестве самостоятельной работы попробуйте устранить и этот недостаток.

4. Оформление (выпуск, кодирование) решения

Кодирование машинной части для конкретного искусственного исполнителя в выбранной инструментальной системе с одновременным формированием инструкций оператору из ручной части.

Машинную часть этого решения составляет сам офисный пакет, поэтому кодирование не требуется. Оно нужно, если появится желание автоматизировать некоторые операции, используя встроенный прогязык пакета (или средства оформления макросов). Тогда нужно визуализировать также и алгоритмы, подлежащие кодированию. Инструментальной системой поддержки этапа по-прежнему будет сам офисный пакет; поэтому выбор внешней системы (среды программирования) не требуется.

5. Тестирование, оптимизация и разработка документации

Проверка на правильность (верификация). Оценка машинной эффективности кода и его эргономической эффективности. Формирование требуемого состава эксплуатационной документации на искусственную систему решения.

Для такой задачи тестирование заключается в мозговой проверке полученного описания с участием предметника (того, чьи знания визуализируются).

В данном случае (описывается та же задача, что будет решаться) описание будет служить и инструкцией пользователю.

Проверьте самостоятельно предложенный Вам эскизный визуал и опишите здесь результаты. После сочинения рабочего визуала предложите оценить его кому-либо и также опишите здесь результаты. Предложите оптимизацию.

6. Ввод в эксплуатацию

6.1. Размещение (развёртывание) решения

Переключение (перенос) системы из тестового контура (окружения) в рабочий.

Перенос сводится к копированию файлов описания на ту машину, где предполагается оформлять дракон-схемы. Если там отсутствует офисный пакет, его сначала нужно установить.

Поясните, как устанавливать офисный пакет, какие при этом могут возникнуть проблемы и как они решаются.

6.2. Запуск в работу

Запуск системы в рабочем контуре. Опробование работы в реальном окружении (опытная эксплуатация). Решение о приёмке системы (начале промышленной эксплуатации).

В данном случае этот шаг тривиален.

II. ЭКСПЛУАТАЦИЯ РЕШЕНИЯ

Для задач такого рода (когда результат получается прямым воздействием на систему) фаза в основном состоит в сопровождении решения, в ходе которого собираются замечания о содержании и форме описания. По мере необходимости задача формализуется заново (возвращаемся к первой фазе).

1. Подготовка к работе

Определение условий решения, подготовка рабочего места (в т.ч. исходных данных для информатической задачи), по необходимости - установка конфигурации (наладка, настройка).

Опишите, с чего начинается графическое редактирование в офисном пакете.

2. Использование решения

2.1. Получение результатов

Запуск решения (цикла), слежение за процессом и управление машинной частью.

Поясните, что нужно для начала оформления дракон-схемы в офисном пакете.

2.2. Обработка результатов

Анализ результатов, определение необходимости нового решения (следующего цикла).

Объясните, чем Вы будете руководствоваться, решая переделать ранее оформленную дракон-схему (дракон-модель).

3. Сопровождение решения

Анализ разработчиком данных обратной связи с пользователями, положения дел в предметной области задачи, определение необходимости совершенствования решения.

Предложите своё описание процесса сопровождения (основываясь на опыте оформления дракон-схем при выполнении последующих заданий).

III. УТИЛИЗАЦИЯ РЕШЕНИЯ

1. Перенос ресурсов

Передача ресурсов, запасённых для использования утилизируемой системой (данных, материальных средств, энергии) в новую путем переноса объектов (носителей) и/или пересылки со старых КТС на новые по каналам передачи. При различиях представления (кодировок, струк-

туры и т.п.) дополнительно проводится конвертация данных посредством специальных программ и/или аппаратуры.

Оцените необходимость конвертации описания задачи для платформ, отличных от Intel-IBM-MS (допустим, если ОС заменена на Линукс).

2. Адаптация технологий

Выделение среди инфопрограммных компонентов (структур данных и алгоритмов в исходном и машинном коде) т.н. повторно используемые, которые можно употреблять для разных задач и/или при смене постановки той же задачи; создание обобщённых компонентов, инвариантных к разным задачам; при необходимости - перетрансляция этих компонентов для новой платформы (модели) информашин.

Выделение инварианта ручной части техпроцесса и/или адаптация старой модели процесса к новой задаче, новым средствам и/или условиям решения той же задачи.

Здесь содержание решения используется для формирования решения других задач и/или той же самой, но в кардинально иной постановке (напр. создать дракон-редактор для среды визуализации).

Предложите иную постановку данной задачи и опишите утилизацию как использование в её решении ранее полученного решения (структур данных и элементов дракон-модели).

При программировании данной задачи, очевидно, нужно заложить в приложение правила техноязыка; так мы фактически придем к собственному редактору дракон-схем. Можно пойти иным путём; оставаясь в среде офисного пакета, запрограммировать отдельные операции на встроенном макроязыке.

3. Фондирование оборудования

Определение среди технических средств повторно используемых подсистем, комплексов, узлов, предназначенные для переноса из утилизируемой системы во вновь создаваемые (для данных — частично совместно с их использованием на копиях, для аппаратуры - лишь по мере снятия её с эксплуатации).

Для данного решениия не требуется, т.к. оно не аппаратное.

<1>В. Детализация укрупнённых операций решения

В разделе содержатся описания техопераций, употреблённых в Разд. Б без детализации. Определения могут пополняться и уточняться по мере изменения состава операций и накопления опыта их выполнения.

Самостоятельно пополняйте этот раздел при оформлении дракон-схем в офисном пакете.

Процесс **уточнения первоначального решения** косвенно отражают следующие операторы:

Изучить начальную постановку – подразумевается, что оформляется решение задачи, уже сложившееся у разработчика в результате осмысления первоначальной её формулировки. Именно этот блок неалгоритмизуем, что определяет неполную формализацию нашей задачи в целом.

Задаче следует дать два имени: полное – для заголовка схемы и краткое (но осмысленное) – для файлов; лучше задаться некоторым форматом имён документов.

Использовать альтернативные конструкции – иногда часть задачи можно описать разными способами (например, тем или другим видом цикла, с операторами реального времени или без них). При этом визуальные алгоритмы равносильны, однако не равно понятны (и, возможно, кодируются на формальном/машинном языке с различной эффективностью).

Структурировать задачу на ветки… – значит разделить алгоритм на относительно самостоятельные части. При этом сложившееся у разработчика "примитивное" представление о задаче сменяется на "силуэтное", и он выделяет части, логически (по смыслу) "напрашивающиеся" в отдельные

ветки. Однако кроме "логики", приходится учитывать и "физику" – при заданном формате диосцены не каждая ветка уместится в высоту и ширину. Тогда понадобится "физически" разносить смысл одной части по ряду веток либо сменить формат рабочего листа; если ни то, ни другое невозможно, остаётся активно использовать вставки, изображая визуальные алгоритмы-вставки на отдельных рабочих листах.

Здесь также даются имена веткам; они образуют массив ЛогИмВеток[n], n≤16 – число веток. Мы используем бит-массив (тип множество) *ШампДляВеток* для определения номеров веток; предполагается, что его можно дополнить именами (перейдя к массиву записей) либо ввести массив имён (элементы в котором размещаются в том же порядке).

Процесс **первоначального оформления** решения задачи в основном отражают вершины:

Выбрать формат диосцены... – реально выбирается файл шаблона, имеющий нужный формат рабочего листа. Его следует пересохранить под именем схемы. В процессе оформления следует периодически сохранять файл вручную (или установить автосохранение в пакете).

Ввести – каждая вершина заполняется текстовой частью, которая определена автором схемы; одновременно корректируется высота и ширина вершины. При необходимости меняется стиль текста на более убористый.

Структура алгоритма сложна? – определяется самостоятельно, например, по критерию автора языка (см. /1, с.91/.

Процессы собственно **конкретного оформления** главным образом отражаются в вершинах:

Упорядочить вершины по алгоритму – значит попробовать составить на рабочем листе дракон-схему из размноженных и заполненных текстом вершин, распределяя шампур-вершины по вертикалям, заданным структурой вершин ветвления, с соблюдением читабельных промежутков. В результате станет очевидно, где нужна подгонка форматов вершин.

Подогнать положение... – значит определить оптимальную ширину вершин для каждой ветки, высоту каждой вершины и положение оси каждой вертикали. Размеры и положение выбранных вершин изменяются посредством буксировки мышью. При этом можно использовать групповые операции по каждой ветке или её части отдельно для контуров вершин и для звеньев вертикали (вершину, составленную из нескольких фигур, следует вначале разгруппировать или изменять после входа в группу). После этого в режиме редактирования текста фигуры выбирается оформление текста (из набора рекомендованных стилей) и кегль шрифта.

Пересадить лианы... – значит подвести линию каждой лианы, исходное положение которой не обеспечивает выбранного продолжения маршрута, к новой точке слияния (не нарушая топологических запретов).

Перезамкнуть петли циклов – значит подвести линию каждой петли к точке возврата (середине намеченного звена вертикали) с учётом свободного места по пути. Практически это осуществляется буксированием линии петли мышью; если этого недостаточно, то используется команда Изменение геометрии панели *Рисование*, после чего формируются необходимые изломы линии и подгоняется их положение по месту.

Заземлить лианы – специфичная для оформления силуэта операция. Означает, что некоторые лианы подводятся не к вертикалям (соседним) тела ветки, а к нижней горизонтали силуэта. Если при этом возникают пересечения, необходимо их разрешить; в рамках строгого шампур-метода для этого надо найти возможность либо заземлить и мешающие лианы, либо попытаться эквивалентно преобразовать часть схемы, в которую они включены так, чтобы ни одна лиана, подлежащая заземлению, не охватывалась другими.

Остальные вершины, думается, особых пояснений не требуют.

<2>. Документы для ссылок

Пополняется сочинителем по мере использования новых источников

Документы для ссылок

1. Пароннджанов В.Д. Как улучшить работу ума. Алгоритмы без программистов – это очень просто! – М.: Дело, 2001.

2. Пароннджанов В.Д. Занимательная информатика. – М.: Дрофа, 2007.

3. Жаринов В.Н. Описание деятельности на основе методологии ДРАКОН. Вводный цикл. – 2009.

4. Тышов Г. Интегрированная среда ДРАКОН. Справка.

5. Функциональное моделирование. Методология IDEF0: Стандарт/русская редакция. – М.: МетаТехнология, 1993.

6. С.В. Черемных, И.О. Семенов, В.С. Ручкин. Структурный анализ систем: IDEF-технологии. – М.: Финансы и статистика, 2001.

7. Герасименко В.А. Защита информации в автоматизированных системах обработки данных.– М.: Энергоатомиздат, 1994.

8. Визуальный язык ДРАКОН: веб-форум. – Конференция OberonCore.

Полезные ресурсы

Ты-среда – здесь можно загрузить файлы установочного комплекта.

Драконография – данный веб-ресурс содержит извлечение из документа /3/ в объёме, достаточном для целей данного документа, и цитаты из некоторых других источников.