< Для посетителей веб-сайта >

Жаринов В.Н.

ОПИСАНИЕ ДЕЯТЕЛЬНОСТИ НА ОСНОВЕ МЕТОДОЛОГИИ ДРАКОН

Вводный цикл (извлечение)

Версия 09.1

ВВЕДЕНИЕ В ДОКУМЕНТ

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 имён 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 приемы 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 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 машинный 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 несёт 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.

Назначение, сведения о версиях, языковые соглашения

1. Документ предназначен для представления содержания страницы гипертекста (либо ряда страниц, на которые делится его содержание) в виде обычного инфордока.

2. Версии документа выпускаются по мере обновления содержания цикла.

3. В тексте 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);
пред.	предыдущий;
след.	следующий;

Сведения о терминологии, обозначениях и сокращениях данного документа см. в п/р 1.3.

Оглавление

1. ВВЕДЕНИЕ

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

1.1. Общие указания

1.1.1. В **тексте** документа применяются следующие приемы оформления.

1.1.1.1. Чтобы у***порядочить работу с материалом***, каждая новая мысль начинается с красной строки, а кроме того, для наглядности часть абзацев имеет особые стили:

- элементы перечисления удобно оформлять 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 мысли, 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]

– так выделяется пункт перечня, требующий особого внимания;

Так выделяется формула обычным текстом, требующая особого внимания

Эти сведения даются сразу перед тем (после того) основного текста, к которому относятся.

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.

Так выделяется формула обычным текстом в примере, примечании

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.

1.1.1.2. 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 сылки 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.

1.1.2. В **графической части** документаиспользуются стандартные стили текста и условные обозначения, приведённые далее (см. п. 1.3.1).

Иллюстрации упорядочены по тексту и снабжены подписями вида:

[Подпись к иллюстрации (абзац уровня 5)]

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.

Иллюстрации оформлены с применением метода КогниСтиль (см. /1, Гл.19/ и п.1.4.3).

1.2. Общие положения

1.2.1. В документе сжато излагается содержание современной отечественной методологии ДРАКОН, предназначенной для описания человеком чьей-либо (или собственной) деятельности, т.е. ***формализации*** (или ***автоформализации***) ***технологических знаний***[4].

Развёрнуто методология изложена в книге её создателя /1/.

1.2.2. Документ предназначен для сведения о графическом (схемном) представлении содержания (логики) трудовых (бизнес-) процессов на базе методологии ДРАКОН в целях инжиниринга (реинжиниринга) бизнеса, управления качеством (в т.ч. в рамках СМК ИСО900Х).

Документ адресован носителям знаний о процессах в организациях различных сфер деятельности, а также специалистам по формализации указанных знаний.

1.2.3. Документ подготовлен с использованием источников информации, указанных в Разд.6. Необходимая информация содержится также в приложениях к документу:

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

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

1.3. Необходимые определения

1.3.1. Основные текстовые и графические термины, обозначения и сокращенияданного документа введены в Приложении 1, а другие необходимые — в тексте п/р 1.4, а также определяются визуально на рисунках. Дополнительные термины вводятся в Приложении 2.

1.3.2. Здесь расшифрованы сокращения, часто употребляемые в тексте курса.

> Сокращения, употребляемые лишь в отдельных местах текста, расшифровываются там же.
> Сокращения, означающие 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
ОС	операционная система
САПР	система автоматизации проектных работ
SADT	Structured Analyses and Design of Technologies — структурный анализ и проектирование технологий

[4] Данные понятия объясняются далее в п.1.4.1, а также в работах /1,Гл.3/ и /2,Разд.6/.

1.4. Введение в предмет

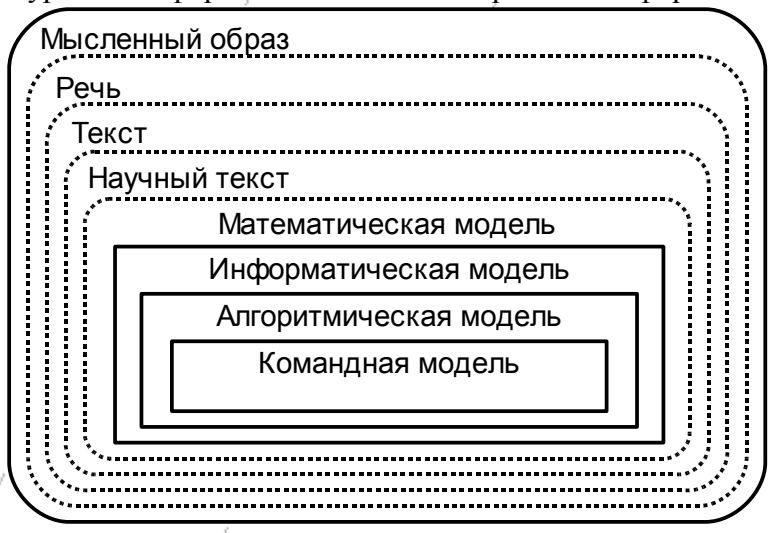
1.4.1. Формализация: процесс и результат

Центральная идея техноязыка – ***когнитивная формализация знаний*** (см. /1,с.32/). ***Когнитивная*** означает приспособленная к познавательным возможностям человека, что обсуждается в /1, Вместо предисловия, Гл.1...5/. Стоит рассмотреть и ***формализацию***, тем более, что в последнее время это понятие уточняется (как и понятие ***знания***).

Формализация тесно связана с моделированием, но эти два процесса – не одно и то же. По Фридланду /4, п. 9.1/ *моделирование* – это разработка моделей как объектов-носителей некоторого смысла, а *формализация* – придание модели свойства быть более или менее однозначно интерпретируемой (понимаемой) кем-либо, кроме её создателя. Вводится понятие *уровня формализации* как степени необходимого участия создателя модели в процессе её интерпретации; если модель недостаточно формализована, то создатель должен разъяснять её содержание другому человеку, уточняя интерпретацию. Возможно, что разъяснением вместо создателя другой человек (транслятор). Нас интересует не всякое моделирование, а информационное.

> *Информационная модель* – это описание объекта моделирования в виде знаков по правилам некоторой знаковой системы (языка), т.е. текст в некотором алфавите (в широком смысле, когда знаки есть любые различимые объекты, о значении которых договорено между сторонами, а в тексте они упорядочены не обязательно линейно).

Чем же обладает создатель (транслятор) модели для её точной интерпретации? Очевидно, представлением о том, как выражать свои знания на некотором языке. Такое представление является в той или иной степени индивидуальным (личностно значимым), и другой человек может вкладывать в элементы того же языка другие значения. Поэтому вводится вертикальная иерархия уровней последовательного уточнения модели. Фридланд в /4, п. 9.1/ выделяет следующие иерархические уровни информационного моделирования и формализации (см. схему):



Уровни моделирования и формализации

Опишем каждый уровень с некоторыми комментариями и уточнениями; главным образом они будут касаться типизации процессов и содержания методов информационной деятельности.

Первым уровнем формализации считается появление мысленного образа объекта моделирования. Уточним, что объект этот чаще всего реальный, воспринятый органами чувств; однако возможен и воображаемый объект (в частности – языковая конструкция).

Отметим, что возможность вообразить себе нечто нереальное подразумевает и возможность представить в уме некий объект абстрактно, мысленно выделив в нем то, что для аппарата мышления существенно в данном случае и отвлекаясь от несущественного. Далее можно обобщить ряд объектов, выделив в них совпадающие черты (свойства) и установив различия. Именно способности к абстрагированию обобщению важны для формализации; они позволяют перейти от личностно значимого смысла к общепонятному для ряда носителей языка (обыденного и/или профессионального).

Обычное (развитое) сознание дает возможность оперировать как с реальными, так и воображаемыми объектами, представляя их как конкретно, так и абстрактно. Однако возможно и ограниченное мышление, в т.ч. примитивное, не дающее возможности абстрагирования и/или обобщения, а иногда и воображения чего-то; оно оперирует только с объектами, «данными нам в ощущениях».

На втором уровне мысленный образ последовательно переходит в речь (в т.ч. внутреннюю). Для этого нужно владеть речью, что достигается погружением в среду общения носителей какого-то языка (который становится родным) по крайней мере с момента рождения.

Укажем на важные обстоятельства. Во-первых, если человек лишен среды языкового общения в течение определённого времени после рождения, то возникает известный «феномен Маугли»: человек не в состоянии мыслить по-человечески и ограничен животным поведением; если среда общения примитивна, то возможен тот же феномен в более или менее смягчённом виде – т.н. эффект сужения кода мышления. Современные исследования показывают, что в утробе матери человек уже способен воспринимать речь на уровне эмоциональных ощущений; т.о. среда развитого общения вокруг человеческого существа необходима еще до появления его на свет.

Во-вторых, возможно владение более чем одним языком как родным; при этом возникают интересные эффекты мышления.

> **Пример**. По воспоминаниям советского российского писателя А.Мамедова[5] в период его детства и юности в г. Баку (1960-е и 70-е годы) существовала реально полиэтническая (и соответственно многоязычная) среда общения; в результате «...дети рано начинали говорить на двух-трех языках и понимать, что об одном, например, лучше сказать на русском, о другом – на азербайджанском, а третье идеально выражается на армянском языке <...> что существуют вещи, непереводимые в принципе...»

На третьем уровне отбрасываются эмоции, свойственные речи; в результате получается текст. Конечно, чтобы записать (а вначале – хотя бы вообразить) текст на естественном (родном для человека) языке, нужно быть грамотным:), но не только.

О формализации от речи к тексту хорошо сказал в свое время американский писатель Марк Твен в своих «Записных книжках»: «…авторучка для писателя служит своеобразным фильтром; не все, о чем думаешь, поддается записи на бумаге.»[6] Нужно уметь отчуждать текст (извлекать данные из речевого представления мыслей) как точный, сжатый и целостный образ объекта формализации.

На четвёртом уровне текст записывается по законам соответствующей науки. Уточним, что здесь используется как система обозначений (напр. химических), так и правила интерпретации знаков и конструкций естественного языка, построения текста.

Отметим, что хотя язык каждой науки обладает своими особенностями, можно выделить и общие свойства всех научных языков. В частности, Фридланд в /1, с.11/ указывает «четыре золотых правила научного изложения и восприятия научного произведения», предложенные акад. А.М. Новиковым и популяризируемые методологом В.В. Краевским:

«1-е правило. Основные понятия, утверждения (теория в целом) должны быть явно и ясно определены независимо от знания их реципиентом (получателем).

2-е правило. При оценке истинности суждений пользоваться только определениями, которые дал пропонент (источник), не подменять их своими представлениями.

3-е правило. Явное определение не принимается, если оно не согласуется с контекстом.

4-е правило. Выбор подходящего определения опирается на специфику задачи, которая решается с помощью данного определения.»

На пятом уровне составляется абстрактная (логико-математическая) модель, которую не просто можно точно записать, но и проверить на правильность, используя математические способы доказательства.

При формализации на этом уровне используется понятие конструктивного объекта.

Конструктивный объект – понятие неопределяемое. По А.П. Ершову, «…в математике часто выделяют объекты, построения и рассуждения, которые называют конструктивными... для такого

[5] Программа "Разночтения" // Телеканал "Культура", эфир от 16.11.2007 г.

[6] Твен М. Собрание сочинений. – М.: Правда, 1980. – Т.8.

рода объектов... дополнительное допущение состоит в том, что эти объекты можно взять <в уме> и рассматривать, построения можно выполнить, а рассуждения можно провести "эффективно"... в реальном мире... осуществление любого из указанных действий д.б. выполнимо в конечное время и с использованием конечного количества материальных средств.»[7] К таким объектам в математике относятся числа, матрицы, графы, таблицы и др., описанные конечным текстом в конечном алфавите.

В математике выделяются сферы, изучающие различные классы конструктивных объектов: теория чисел, теория графов, теория функций и пр.

На шестом уровне строится *информатическая модель*, где ставится задача, которую нужно решить. При этом выделяются данные (в т.ч. отображающие реальные объекты) и операции по обработке данных, описывается последовательность операций. При этом используется некий формальный язык и задаётся система правил порождения (вывода), разрешающих переход от одних конструктивных объектов к другим. Этот уровень считается самым высоким при формализации – предполагается, что информатическую модель однозначно понимает любой подготовленный человек.

На данном уровне важным является вопрос о базовых элементах структуры операций.

Седьмой уровень детализирует модель предыдущего уровня для конкретного исполнителя (по Ершову – «пользователя материальных средств»).

Здесь уточним, что исполнитель формально характеризуется:

- набором выполняемых операций, или множеством предписаний исполнителя (МПИ); будем для краткости называть его *репертуаром*[8];
- набором, или множеством типов (МТ) объектов (напр. типов данных), воспринимаемых исполнителем (разрешённых); будем далее называть его *багажом* исполнителя;
- множеством механизмов, выполняющих конкретные операции над конкретными типами объектов; будем называть это множество *реквизитом*;
- *языком общения*, на котором отчуждаются описания операций, объектов и механизмов.

Изначальным исполнителем является человек, который использует естественный язык общения. При формализации этот язык тоже д.б. так или иначе формализован – в противном случае модель не будет однозначной.

Информатическая модель «переводится» на язык исполнителя; в результате получается алгоритмическая модель – процесс решения задачи данным исполнителем. Часто язык абстрагирован от исполнителей; при этом задаётся базовый набор операций и типов данных.

Под *алгоритмом* понимают конечный набор правил (указаний, предписаний), который предназначен определённому исполнителю и удовлетворяет следующим основным требованиям:

- *Массовость.* Алгоритм должен обеспечить выполнение не единственной процедуры, но класса однородных процедур. Начальная система величин может выбираться из некоторого потенциально бесконечного множества.
- *Детерминированность.* Указания, образующие алгоритм, должны однозначно пониматься исполнителем. Реализация алгоритма, т.о., никак не зависит от желаний исполнителя.
- *Результативность.* При выполнении за конечное число шагов должен быть получен результат либо указание на неприменимость данного алгоритма для решения интересующей нас задачи.

Алгоритм описывается на некотором формальном алгоритмическом языке (алгоязыке).

На восьмом, завершающем, уровне, создаётся *командная модель* – последовательность шагов, решающих задачу. Этот уровень завершает иерархию моделирования и формализации.

Отметим, что Фридланд называет эту модель *компьютерной*, предполагая, что на этом уровне обязательно использование информашины; однако точно так же может действовать и человек (допустим, "школьная" запись решения математической задачи "по действиям" есть командная модель этой задачи; а запись, скажем, операции умножения (деления) "в столбик" – этой операции).

---

[7] Ершов А.П. Введение в теоретическое программирование. Беседы о методе. – М.: Наука, 1977, с. 59-60.

[8] Термин введен В.Д. Пароджановым.

Еще одно представление модели решения – маршрутное – введем, исходя из того, что алгопроцесс есть цепочка смен одного объекта другим. На этом уровне все действия линейно упорядочены, т.е. командная модель «развернута» в цепь команд – м а р ш р у т от начала к концу алгоритма. Понятно, что это можно сделать только для конкретных значений исходных данных; тогда известны все логические условия, определяющие выбор конкретного маршрута из разнообразия возможных. Это и нужно реальному исполнителю – человеку или машине.

> Итак, в конечном итоге информатическая модель – это сведение текста в широком смысле к набору линейных прочтений.

Модели уровней 1...4 можно считать слабо формализованными, при переходе сверху вниз теряется часть информации об описываемом объекте; соответственно на схеме их контуры изображены штриховыми линиями. На последующих уровнях происходит формальное преобразование одних моделей в другие; контуры этих уровней мы изобразили сплошными линиями. На всех этапах, кроме первого, прообразом служит не действительность, а образ её – модель предыдущего уровня.

В частном случае, если прообразом является математический объект (т.е., начиная с уровня 5), возможно взаимно однозначное соответствие объекта и модели (изоморфизм). Соответствие м.б. и односторонним (гомоморфным), когда каждому элементу модели можно указать соответствующий элемент в исходном объекте, но не наоборот. Эти вопросы подробно изучает математическая логика.

Индивидуальный характер моделей каждого уровня определяется особенностями мышления конкретного человека. Они формируются в ходе предыдущего развития сознания личности.

> Следуя Фридланду, мы понимаем ***знание*** как компоненту ***информации***, неотъемлемую от человека, а ***данные*** – как информативную (для сознания) компоненту окружающего мира, в т.ч. отчуждаемую человеком часть знания (см. /4, Гл.4/ и п.1.1.2 Приложения 1).

Итак, под ***информатическим*** мы понимаем предельно формализованное, не требующее обязательного участия человека для своей интерпретации. Оно есть часть ***информационного*** наряду с ***интеллектуальным***, относящимся только к человеческому мышлению.

> С позиций Фридланда, становится неоправданным также эпитет «интеллектуальный» по отношению к искусственным системам. Мы можем говорить лишь об и н т е л л е к т н о й системе, если при её проектировании использованы все возможности (в т.ч. знания о процессах мышления) по адаптации алгоритмической обстановки к меняющейся реальности.

Комплексируя выделенные выше уровни, можно выделить ***качественную*** (по Фридланду – уровни 1…4), ***математическую*** (уровень 5) и ***информатическую*** (уровни 6…8) стадии моделирования и формализации.

> С данной точки зрения существенно, что решение любой задачи проходит описанные стадии; однако очевидно, что в тех или иных случаях (напр., при решении повседневных задач) невозможно говорить о решении средствами математики и информатики в привычном смысле – явном («на бумаге») и точном. Поэтому следует понимать, что существует (и активно используется людьми) также нестрогая (неформальная, интуитивная) математика и информатика. Подходы к её выделению и описанию сформированы в последние годы[9].

Используя тернарную иерархию стадий, в принципе можно «развернуть» иерархию уровней формализации и по-иному; для данного документа в этом не необходимости.

Процесс формализации (как строгой, так и нестрогой) итеративен в целом и в пределах каждого уровня: этапы качественной формализации можно повторять до получения удовлетворительного письменного описания предмета, этапы математической – до подходящей формулировки задачи, информатической – до командной модели, реализуемой исполнителем.

К средствам и способам формализации знаний можно и нужно предъявлять теоретически и практически обоснованные требования качества, как и в других сферах деятельности человека. Говоря просто, форма, цвет, расположение на диосцене элементов данных (включая часть знаний, ***отчуждаемых*** от носителя-человека) д.б. удобны, эстетичны также как, скажем, у элементов операторского пульта машины, инструмента, мебели.

> Традиционная эргономика, научно изучая системы «человек-техника», согласует

[9] См., напр., работу: Герасименко В.А. Защита информации в АСОД. – М.: Энергоатомиздат, 1994.3, Гл.3.

свойства техники с человеческими для повышения гарантоспособности этих систем. ***Когнитивная эргономика***, изучая системы «человек-знания», аналогично видоизменяет представление данных (в первую очередь на диосцене), исходя из свойств мышления.

Обсудим ещё одну проблему – как понимать **автоформализацию знаний**. Фридланд в /4, п. 9.3/ дает следующее определение: это формализация, которую человек (имеется в виду носитель знаний) выполняет сам. Далее приводится конкретный пример, который, по мысли автора, должен исчерпывать собой это понятие.

> **Пример**. В начале 1990-х годов появился автокран со встроенным управляющим компьютером, который фиксировал манипуляции крановщика на пульте управления в виде сценария и затем мог «прокручивать» этот сценарий столько раз, сколько надо. Пусть нужно поднять на пятый этаж 20 поддонов с кирпичами; с помощью такого крана эта задача решается следующим образом. Крановщик, включив машину, выполняет процедуру один раз очень тщательно; компьютер запоминает все его действия. Оставшиеся 19 раз человек просто отдает команду вызова сценария, а кран выполняет все автоматически; однако каждый раз перед этим кто-то должен подтаскивать очередной поддон к тому месту, с которого был взят первый.

Здесь автоформализация протекает путём записи данных о действиях человека при решении к.-л. задачи. Исходя из этого, Фридланд определяет автоформализацию профессиональных знаний как процесс автоматического оформления действий пользователя в виде программы для информашины с помощью специальных средств. Однако это следует считать лишь частным случаем, рассчитанным на человека с низким уровнем информатической культуры; этот человек, используя свое профессиональное знание (уровень которого, напротив м.б. весьма высок), фактически ставит «натурный эксперимент» по выработке данных о техпроцессе на своем рабочем месте, содержащем средства автооформления. В дальнейшем результат этого эксперимента (возможно, наиболее удачного из нескольких повторов) использует уже автоматизированная система для работы без участия человека. Очевидно, что такая автоформализация возможна как средство решения повторяющихся задач управления; иначе оператору нет смысла вести запись, тем более тщательно следя за своими действиями. Кроме того, она носит частный характер; чтобы решить даже ту же самую задачу, но по-другому, эксперимент с участием человека следует провести вновь.

По сути, мы видим ***метод проб и ошибок***, или «информатического тыка», который, конечно, не является единственно возможным (по крайней мере, с тех пор, как человек освоил научный метод познания). А каким ещё образом может происходить автоформализация?

Общеизвестен ***логико-математический метод*** (применительно к программированию изучаемый в школьной информатике как основной), когда носитель знаний описывает свою задачу на некоем формальном ЯПЗ и по полученной модели формальным методом строит гарантированно работоспособный и оптимальный в некотором смысле алгоритм решения (который далее кодируется на доступном языке программирования, также «по всем правилам» оптимизации). В этом случае пользователь формирует технологию решения на базе имеющихся инструментальных средств подобно тому, как это делал бы квалифицированный разработчик. И этот путь ничто не мешает считать автоформализацией (правда, годится он для носителей профессиональных знаний, обладающих в придачу достаточной математической и информатической культурой, чтобы создать модель и алгоритм решения своей задачи). В то же время и он не исчерпывает всех возможностей; существует по крайней мере ещё одно направление.

> В свое время ещё Дж. фон Нейман указывал по поводу программирования и вообще информатического описания решений сложных проблем: «…у нас не может быть уверенности в том, что в этой области объект <программа> не является простейшим описанием самого себя, т.е. что описание его любым словесным или формально-логическим методом не приведёт к чему-то более сложному, запутанному и трудно выполнимому»[10].

Из этой посылки вытекает ***метод прототипирования*** инфопрограммных изделий, при котором носитель знаний, владея некоторым языком программирования и имея знания (опыт)

[10]Громов Г.Р. Национальные информационные ресурсы. – М.:Наука, 1985. – С. 216.

его использования, создает сразу программу, делающую «примерно то, что надо» для решения задачи; далее он её улучшает (часто в ходе эксплуатации), используя и как «техзадание».

Т.о. обладая знанием о некотором инструменте формализации, пользователь может создавать средства решения своих задач интуитивно, без построения формальной модели, но и избегая «тыка»; оптимизация проводится сразу на работающем изделии (напр., тексте программы).

Таким путём обычно появляются программы (и инфорсимы, и вообще искусственные системы) «для себя» (т.н. утилитарные); его тоже следует рассматривать как автоформализацию знаний.

Итак, автоформализация (как, впрочем, и формализация «чужих» задач) возможна тремя различными классами методов, которые составляют опять-таки системную триаду; методы «тыка» можно относить к эмоциональным, логико-математические – к рациональным и прототипные – к интуитивным. Выбор класса зависит от способностей и опыта разработчика решения, его предпочтений, возможностей имеющегося инструментария. К настоящему моменту уже назрело понимание различия классов и стали оформляться соответствующие технологии.

> **Пример: логико-математические методы**. В этом классе наиболее широко распространены упомянутые CASE-стандарты семейства IDEF как часть более широкого семейства методологий SADT. Все они не строго формальны; поэтому в настоящее время предпринимаются усилия по более строгой математической интерпретации их[11].
>
> **Пример: методы «тыка»**. Популярные инфопроги (напр. пакет MS Office) содержат встроенное средство автозаписи всех действий пользователя с элементами (объектами) приложения, активного в данный момент. Скажем, человек может создавать форму (шаблон) к.-л. документа, а информашина будет автоматически записывать последовательность его действий для этой цели (конечно, где начинать и где завершать запись – определяет сам человек). Далее пользователь может отдать команду сохранения записи как макрокоманды (новой функции) текущего приложения; потом он же или другой человек может выполнить эту команду (как в примере с автокраном).
>
> **Пример: методы прототипирования**. В настоящее время активно развивается семейство т.н. RAD-методологий[12] программирования. Они основаны на создании приложения-прототипа и его усовершенствовании; при этом прототип регулярно демонстрируется будущему пользователю для оценки соответствия его требованиям.

Рассмотренный процесс формализации является частью более широкого процесса ***структуризации***[13] предметных областей. Методология структуризации сложных систем позволяет формировать их адекватные модели (сохраняющие все значимые характеристики). Как можно структуризовать техноязык, показано далее.

[11] См.: Костров А.В. и др. Уроки информационного менеджмента. – М.:Финансы и статистика, 2005. – Разд.7;П.8.1.

[12] От англ. Rapid Applications Development – быстрая разработка приложений.

[13] На это указывалось, в частности, в: Герасименко В.А. ..., пп. 3.6, 3.7.

1.4.2. Введение в методологию ДРАКОН

Структуризация в познании предшествует формализации; в самом деле, для начала мы стремимся «разложить по полочкам» незнакомую предметную область, определить устойчивые взаимосвязи между сущностями «с разных полочек», а уже потом стремимся по необходимости (и возможности) облечь найденные структуры в строгие математические формы.

Структуризацию деятельности, как и любого объективного явления, с материалистической т. зр. можно провести в категориях пространства, времени и движения. Нам удобнее придерживаться обратного порядка.

Цель изучения цикла - стать начинающими разработчиками (мы будем говорить *сочинителями*, как в /2/) алгоритмов. В связи с этим рассмотрим некоторые общие вопросы.

1.4.2.1. Сколько голов у ДРАКОНа?

Вообще-то на этот вопрос можно ответить по-разному. В мифологии бывают драконы и одноглавые, и стоглавые. Если обратиться к нашему предмету, то создатель техноязыка указывает, что «...язык ДРАКОН выполняет две принципиально разные функции. Для большинства работников он является новым средством повышения эффективности интеллектуального труда, причем у этого средства практически нет аналогов в мировой практике. В этом качестве ДРАКОН не имеет ни малейшего отношения к программированию. ...Вторая функция состоит в том, что для программистов ДРАКОН действительно является языком программирования. Таким образом, ДРАКОН имеет две головы, обращённые к совершенно разным аудиториям. Причем каждая голова пытается угадать сокровенные потребности своей аудитории и по возможности удовлетворить их наилучшим образом.» /1, с.26/.

Казалось бы, ответ получен. Но здесь вспомним, что цель создания языка — формализация знаний, да не простая, а когнитивная, т.е. опирающаяся на особенности информационной деятельности человека. По сути, под названием ДРАКОН скрывается новая методология формализации (автоформализации) знаний, прежде всего профессиональных. И «головы», т.е. базовые элементы структуры, имеет смысл выделять применительно к ней.

Сколько базовых элементов брать при структуризации? Вроде бы это должно проистекать исключительно из знаний о рассматриваемой системе. Однако имеется и иной подход, заявленный, напр., в /3/, при котором предлагается образовывать структуру системы из триад элементов. В целом на сегодня можно говорить о тринитарном мышлении, при котором исходят из тернарных представлений об объектах познания. Вкратце изложим его основы[14].

> Базовой структурой объекта является *триада*, т.е. комплекс из трех элементов, так или иначе связанных между собой. Сложившиеся ранее бинарные структуры (диады) достаточны для анализа, но не для синтеза. По виду связи различают триады трех типов: ***линейные***, ***переходные***, ***системные***. Первые одномерны в семантике (упорядочены на одной смысловой оси); третий элемент может представлять промежуточное значение между двумя другими либо иметь смысл центра, «золотой середины» между крайностями (как в китайском видении мира), однако критерий оптимального центра в единственном измерении не находим. Во вторых триадах один элемент поднят на более высокий уровень, где должно быть разрешено противоречие, в котором находятся два других элемента; однако механизм снятия противоречий в этой структуре не раскрывается. Наконец, триады третьего вида образуются равноправными элементами, находящимися на одном уровне общности. По смыслу это может трактоваться как треугольник «рацио-эмоцио-интуицио» (в категориях мышления – «понятие-образ-символ»). Любые два элемента триады находятся в противоречии; каждый элемент может участвовать в разрешении противоречий между двумя другими как фактор их сосуществования в системе. Особо важной считается не полнота описания, а его целостность.

Какими будут элементы методологии формализации знаний? Предлагается выделять в ней язык представления знаний (ЯПЗ), технологию формализации (ТФЗ) и реализацию. Для наглядности определим сущность этих элементов на вполне вещественном инструменте челове-

[14] Более подробно см., напр. /3, Разд.1/.

ческой деятельности – топоре. Большинство читателей, пожалуй, удивятся этому, но вспомним, что топор – по сути, древнейшее орудие труда, в пользовании которым начинали создаваться более совершенные, а также и добываться те самые профессиональные знания, которые мы формализуем. Кроме того, хороший топор в умелых руках – инструмент многофункциональный – квалифицированный плотник мог с его помощью не только, скажем, заготовить дрова и соорудить загон для скота (а то и ловушку для крупного зверя), но и построить дом, и лодку-долбленку, и даже вытесать посуду, столовые приборы, вырезать простейшие украшения и игрушки; по сути, один инструмент удовлетворял все базовые потребности человека. Не правда ли, хорошая методология формализации знаний должна быть столь же совершенной?

> Не забудем, что ДРАКОН формализует импер-знания (технологические), т.е. состоящие из действий (движений, операций). На любом уровне формализации ***алгоритм***, следуя Паронжанову, считаем частным случаем ***техпроцесса***, когда предмет труда – данные.

Итак, определим содержание выделенных элементов. Начнем с **языка**. «Язык топора» – это возможные элементарные действия, такие как рубящее, строгающее, вырезающее движения. Они элементарны потому, что освоены исполнителем и совершаются также естественно, как, например, дыхание, удержание равновесия и т.п. Кроме того, должны существовать операции выбора того или иного движения, варьирования его параметров (усилия, направленности и пр.). Из действий и актов выбора слагается процесс получения любого результата труда.

Язык формализации импер-знаний – это элементарные (неделимые) операторы, однозначно понимаемые исполнителем, а также базовые структурные конструкции. При формализации из них составляется описание существующей или проектируемой деятельности (понятно, что при автоформализации описание составляет тот, кому работать). Главная характеристика языка – это его применимость к описанию того или иного класса процессов. Когнитивно качественный язык максимально учитывает особенности восприятия человеком данных на диосцене.

Языков формализации техзнаний существует множество; по сути, любой алгоязык есть более или менее специализированный императивный ЯПЗ. Почему же ДРАКОН претендует на универсальность? Чтобы понять это, нужно рассмотреть структуру самих формализуемых знаний. Паронджанов сделал это в /1, с.191-193/, а мы здесь приведем несколько уточнённую классификацию (см. след. стр.), где формы представления исходят из когнитивной эргономичности.

Итак, полный ЯПЗ должен распадаться на четыре подъязыка; два из них определены Паронджановым, а два других представляют детализацию того, что он определил как декларативные знания и на этом остановился. Это вполне понятно; ДРАКОН предназначен для формализации импер-знаний, и в его рамках в структуру деклар-знаний вдаваться необязательно.

В императивных (импер-языках) объекты (предметы и результаты труда) лишь указываются (как именованные величины визуалов). Для полноты описания нужно формализовать знания об объектах (деталях, материалах, документах и пр.) на соответствующих языках описания объектов, т.н. *декларативных (деклар-языках)*. Звучит несколько устрашающе, но на деле все просто; скажем, деклар-описание детали – это совокупность чертежа (структурной модели) и спецификации (атрибутивной). Такое же описание документа (скажем, накладной) может состоять из шаблона, отражающего состав и расположение реквизитов, их логические связи (напр., расчетные соотношения, условия заполнения) и спецификации, описывающей свойства каждого уникального вида реквизитов.

В результате формализации и декларативных, и технологических знаний получается детальное описание. Однако необходимо также обеспечить целостность модели.

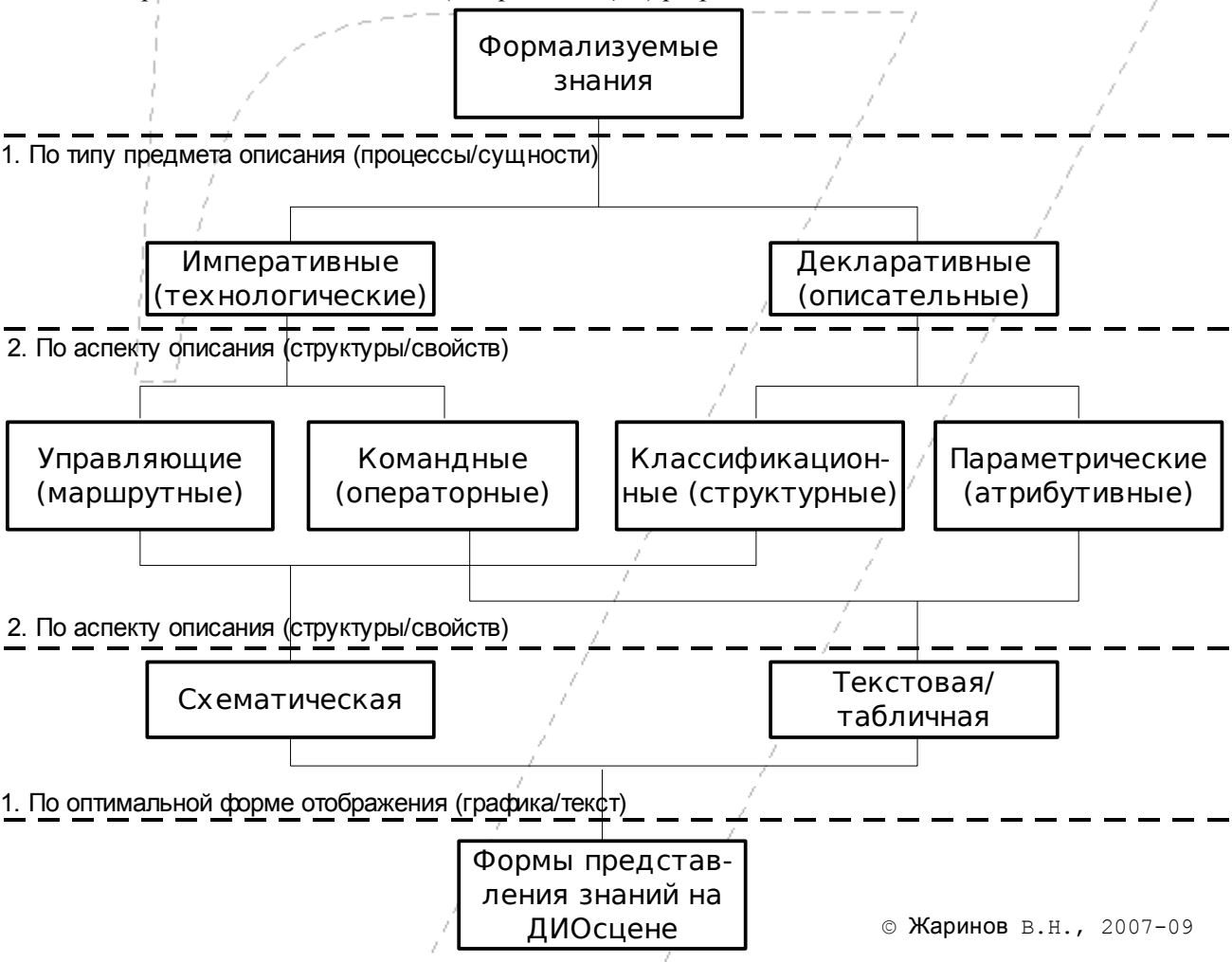
> Модель будет целостной, если включает обобщённое описание, где императивная и декларативная составляющие совмещены; оно создаётся предварительно.

Для обобщённого описания рекомендуется методология т.н. функционального моделирования IDEF0; её стандарт опубликован на русском языке (см. /5/, а также пособия для CA, напр. /6/).

В Приложении 2 описан набор деклар-языков, пригодных для визуализации бизнес-процессов.

Конечной целью алгоритмизации является алгоритм деятельности, который исполняется человеком и/или машиной последовательно шаг за шагом.

В информатике процесс формального описания состава и структур объектов (т.н. *датаматизация*) хорошо раскрыт и формализован (прежде всего в связи с технологией баз данных); аналогичный же процесс для деятельности (*алгоритмизация*) разработан недостаточно.



Классификация форм представления знаний (уточнённая)

Напомним, что деклар-языки специфичны для предметных областей. В то же время импер-языки в принципе универсальны; любая деятельность формализуется как совокупность маршрутов. Вопрос лишь в том, какой способ представления этой совокупности выбрать.

Идея техноязыка в том, что маршрутный подъязык имеет графическую основу и унифицированный синтаксис, тогда как остальные подъязыки выбираются относительно свободно. Практически в процессе формализации можно употреблять ряд последовательно сменяющих друг друга командных подъязыков: от близкого к естественному до алгоритмического. Исходный подъязык – тот, на котором описывает свою деятельность человек-носитель знаний.

Математически любое графическое описание структур лежит среди конструктивных объектов особого рода – графов (граф-схем)[15]; при этом графическое представление маршрутов относят к классу граф-схем алгоритмов (ГСА). Процесс составления ГСА по правилам ДРАКОНа именуется *визуализацией* деятельности (импер-знаний), а ГСА в техноязыке – *дракон-схемами*.

Перейдем к **реализации**, сиречь тому орудию, которым совершаются операции. «Для топора» это сам топор и есть; более широко имеем систему «человек-топор». Понятно, что топоры бывают разные, и не каждый удобен (и даже пригоден) для конкретной работы. Наиболее универсален как раз плотницкий топор; можно вспомнить ещё альпинистский топорик-ледоруб, топор-колун, геологический топор (обычно называемый молотком) и его родственника – шахтерское кайло.

[15] Понятие о графах см. курс математики.

Реализация МФЗ – это информационная система поддержки употребляемого в ней языка. Вообще-то она человеко-машинная, но специально разрабатывается только машинная часть. Эту часть следует называть ***информатической системой поддержки (ИСП)***, следуя уточняемой в настоящее время терминологии информатики (подробнее см. /4, Гл.6/). Крайними случаями реализации считаем *ручную* – формализация полностью силами человека; *автоматическую* – полностью силами машины (от человека требуется только подготовить описание задачи). Практически возможна лишь в той или иной степени *автоматизированная* формализация знаний, когда ИСП выступает партнером человека; в первую очередь это вытекает из математического запрета на алгоритмизацию процесса построения алгоритма в общем случае.

> Программно-аппаратный комплекс средств автоматизации (косавт) в основе имеет ***машину для переработки данных*** (традиционно называемую вычислительной, или компьютером, что одно и то же). Именно в контексте формализации знаний наиболее очевидно, что эта машина языковая, а вычисления есть лишь частная её функция. Следуя терминообразованию в /4/, будем говорить об *информатической машине* (*информашине*).

Идея реализации техноязыка – охватить любые случаи максимально удобно для неспециалиста в ИТ. Поэтому как реализация рассматривается система «человек-инфортехника», где техника варьируется от простейших чертежно-письменных принадлежностей до авторамов САПР типа I-CASE, и учитывается опыт конструирования физических систем (см. /1, Гл.5/).

Наконец, **технология**. Понятно, что это процесс применения выбранного орудия. «Для топора» это последовательность элементарных операций, причем из множества, заданного в «топорном языке»; варьируются только количественные характеристики действия (скажем, твёрдый материал надо рубить или строгать с большим усилием). Понятно, что разными топорами одно и то же действие придется совершать по-своему и с различной эффективностью.

Технология формализации импер-знаний – это процесс описания деятельности на основе данного ЯПЗ средствами конкретной ИСП. Если язык и/или реализация неадекватны сфере деятельности, то этот процесс может напоминать «варку супа из топора»:–). ТФЗ зависит и от возможностей реализации как информатической системы визуализации, например:

- «в карандаше», когда все моделирование ведет человек (в крайнем случае рисуя на экране);
- в виде машинных описаний, где автоматически организуются модельные данные;
- в среде автоматизации построения моделей, где также формализуются правила языка.

В целом ТФЗ – вещь неформальная, она должна согласовать формализм языка и ограничения реализации для достижения цели применения МФЗ – создания требуемого описания.

> Итак, формальный язык представления знаний (допустим, техноязык ДРАКОН) применяется для представления содержания деятельности по определённой (но неформальной, неалгоритмической) технологии; эта технология, в свою очередь, зависит от возможностей реализации как информатической системы. Таковы основные взаимосвязи в системной триаде методологии формализации знаний.

А как насчет двух голов – «для программистов» и «для большинства», – с которых мы начали? Вместо этого можно говорить о двух «обличьях» нашего «трехглавого» ДРАКОНа. Они меняются в зависимости именно от немаршрутных подъязыков (командного и атрибутивного), употребляемых при визуализации, образуя:

- неформальный язык – для профессионалов своего дела (носителей знаний предметной области), определен как версия ДРАКОН-1 и называется п с е в д о я з ы к о м;
- формальный язык – для профессионалов информатизации того или иного дела (ИТ-специалистов, таких как программисты, системные аналитики), это в и з у а л ь н ы й я з ы к п р о г р а м м и р о в а н и я (версия ДРАКОН-2).

Более точно, следует разделять различия в формальности языка и разнообразие его специализаций; это две разных точки зрения.

В результате имеем базис представлений (пространство выбора) 2x2. Его измерения: степень формальности {неформальный; формальный}; специализация {предметная; информатическая}.

С т. зр. специализаций ДРАКОН, образно говоря, имеет два «гардероба»:

- прикладной – содержит «костюмы» (наборы немаршрутных подъязыков), которые «пошиты» на носителей, т.е. формулируют текст икон в терминах рода их занятий (напр., педагогики, биологии, нефтехимии, механообработки);
- информатический – содержит «костюмы», «пошитые на заказ» ИТ-специалистов; здесь текст икон записывается в терминах переработки данных по некоторой технологии информатизации (напр., разработки баз данных, веб-приложений, обработки текстов, графики).

Сегодня чаще всего под формализацией явно или неявно понимается лишь употребление информатического гардероба, когда командное и декларативное описания подгоняются под шаблон языков программирования. Кроме того, традиционно абсолютизируется текст как единственная форма представления данных (включая отчуждённые знания), что неэргономично (об этом подробнее в /1, Гл.5,17/). Однако из целостного определения формализации, данного в /4/ и обсуждённого нами в п. 1.4.1, следует, что изначально описание деятельности «облачается из предметного гардероба». Именно здесь проявляется достоинство унифицированного и строгого визуального синтаксиса ДРАКОНа: по мере уточнения представления о деятельности правила описания её структуры не меняются, меняется лишь правила описания содержания элементов структуры.

С т. зр. формальности, возможны наборы языков разной строгости для одной и той же специализации. Для первого «гардероба» их происхождение очевидно; некий процесс (допустим, педагогический) можно вначале описывать «своими словами», а по мере уяснения переходить к «более научной» форме. Во втором «гардеробе» можно выделить гибридные техноязыки, построенные на базе языков программирования (ДРАКОН-Си и т.п.), вплоть до языка кодирования икон на Ассемблере конкретной модели информашины.

Формализуют техзадания на языках спецификаций задач, обычно менее строгих; их набор можно рассматривать и как отдельный гардероб-посредник между двумя другими, тем самым расширив базис представлений по одной из координат.

В нашей стране основным языком спецификаций, пожалуй, следует считать т.н. алгоритмическую нотацию (русский псевдокод, или «школьный» алгоритмический язык), разработанную в СССР под руководством А.П. Ершова; именно на основе её немаршрутных подъязыков можно построить русский специфтехноязык (что продемонстрировано в /1, Рис.92...95/).

Если носитель знаний знает формальный язык и главное – понимает, как интерпретировать на нем свой исходный язык, то он может автоформализовать ту или иную деятельность достаточно строго и без ИТ-специалиста, вплоть до разработки программы автоматизации «для себя» (утилитарной). Другой вопрос, всегда ли нужно обходиться своими силами.

Итак, ДРАКОН – это целое семейство техноязыков, употребляемых на всех этапах формализации императивных знаний. При этом неформальные техноязыки (псевдоязыки) образуют подмножество (продолжая биологические аналогии – «отряд») ДРАКОН-1; формальные – «отряд» ДРАКОН-2. Члены семейства объединены визуальным синтаксисом; по каким принципам он образуется, рассмотрим далее.

1.4.2.2. Основы структуризации импер-языков

В основе методологии ДРАКОН лежит ряд идей, обеспечивающих удобство и точность описания алгоритмов в форме, пригодной для человеческого восприятия; в /1, с.337/ она названа *знаковой*, но мы будем говорить о *символической* (понятие «знак» уж очень широко используется в других значениях); в первую очередь преследовалась цель строго формализовать описание структуры деятельности, чтобы её можно было впоследствии однозначно переводить в форму, представимую в машинах – *предметную* (практически – в программный код).

Особенности ДРАКОНа удобно рассмотреть, оперируя категориями материи – пространства, времени, движения – применительно к деятельности в информатике. Напомним, что она моделируется как алгоритмический (технологический) процесс.

Информатическое время протекает дискретно, по шагам алгопроцесса; на каждом шаге исполнитель совершает целостное действие, изменяющее объекты процесса и обстановку в целом. Его категории можно выделить так. Положим, что имеется конечный набор «обстоятельств времени» алгопроцесса, а при алгоритмизации конкретной задачи учитывается бо́льшая или ме́ньшая их часть. Тогда можно построить следующую иерархию информатического времени:

- *абстрактное* – временны*е* соотношения в описании алгопроцесса игнорируются (предполагая, что процесс и каждый его шаг в любом случае начнется и закончится тогда, когда нам это нужно, исходя из цели и условий обстановки);
- *условное* – принимается во внимание, что шаги процесса имеют конечную длительность, а среда взаимодействия исполнителя процесса – определённую динамику; поэтому нужно контролировать время выполнения процесса и привязывать начало операторов взаимодействия (ввода-вывода) к определённым моментам;
- *реальное* – также учитывается, что процесс может исполняться не сам по себе, а во взаимодействии с другими процессами, а исполнитель может одновременно выполнять не один, а несколько процессов, ведущих к различным целям (в общем случае ряд систем процессов распределяются между рядом координированных исполнителей); поэтому нужно обеспечивать межпроцессное взаимодействие и распределение работ между исполнителями.

Как правило, «школьные» информатические задачи составляются для абстрактного времени.

> **Пример.** Хорошая иллюстрация деятельности в реальном времени – сцена из известного фильма Чаплина «Великий диктатор», где герой-парикмахер бреет клиента под музыку Брамса, привязывая определённые фазы бритья к музыкальным фразам:))

Здесь мы обсуждали, конечно, логическое время; существует и физическое. В информатике оно отражается через состояния т.н. датчиков времени — объектов данных, значения которых формируются периодическими физическими процессами (колебаниями в генераторе синхронизации, моментами прохождения процессами во внешней среде определённых состояний). Эти значения математически образуют ось времени - числовой ряд (обычно целый) и при необходимости пересчитываются по определённой системе счёта времени (календарю); дискретность алгопроцесса ведёт к тому, что можно сопоставить его шагам отрезки на оси времени. За качественным пониманием физического времени, естественно, нужно обратиться к физике.

Информатическое движение – это изменение обстановки в ходе деятельности. Движение в алгопроцессе складывается из выполнения операторов и переходов.

В рамках ДРАКОНа ход выполнения визуала можно отображать положением т.н. ***рабочей точки*** (в терминах /2/ – «дракон-поезда») на той или иной вершине дракон-схемы; в этой точке как бы находится исполнитель алгоритма при выполнении конкретной иконы; в результате выполнения изменяется его состояние, т.е. значения некоторых переменных величин алгоритма и служебных переменных самого исполнителя, а затем происходит переход по линии (при этом состояние считается неизменным).

Состояние является важным компонентом алгоритмической обстановки; в неё также входят текущая координата рабочей точки, величины, постоянные при исполнении данного алгоритма (константы), и сформулированные при математической формализации процесса соотношения между величинами (константами и/или переменными) – начальные и граничные условия.

Можно выделить следующие классы операторов:

Рабочий – отражает совершение действия исполнителем.

Холостой – отражает бездействие исполнителя в течение некоторого времени.

Служебный (системный) – отражает изменение исполнителем конфигурации (режима работы) для настройки на оптимальное продолжение выполнения алгоритма.

Рабочий оператор (в терминах блок-схем – функциональный) тоже имеет типы. Их можно выделить по характеру совершаемого действия. На этот счёт есть различные точки зрения; так, в техноязыке фактически предлагаются два типа работ – действие и ввод-вывод.

Мы можем усовершенствовать это представление, как показано далее в п/п 2.3.1.8.

На уровне алгоритмического моделирования допускаются также т.н. *псевдооператоры объявления*, отражающие декларативное знание. В техноязыке для объявления существует икона *Полка*.

Холостой оператор имеет следующие разновидности:

- выдержка заданного интервала (по единому времени исполнителя);
- задержка до заданного момента времени (единого либо локального для процесса);
- пассивное ожидание заданного события (изменения алгоритмической обстановки).

Локальное время отсчитывается по счётчику (таймеру) заведённому в данном визуале (до оператора задержки). Таймеров м.б. много; тем самым ведётся отсчёт от ряда точек процесса.

При ожидании события исполнитель проверяет условие его наступления в цикле. Конечно, на это затрачивается некоторое время, поэтому чтобы реакция на событие была своевременной, нужно соответствующее быстродействие.

Служебные операторы могут иметь разные типы в зависимости от заложенных возможностей алгоритмической настройки (адаптации) исполнителя. Они изменяют значения не алгоритмических, а специальных служебных переменных, управляющих конфигурацией.

В любом случае у формального исполнителя (информашины или её модели) задаётся некое начальное состояние, с которого всякий раз начинается его работа; обычно в репертуаре существует и операция т.н. ***начального пуска*** (программной перезагрузки, жарг. «тёплого старта»), устанавливающая это состояние. Далее, исполнитель может иметь две и более типовых конфигурации ресурсов, и тогда существуют команды для переходов от одной из них к другой (возможно, некоторые переходы исполняются по внешним сигналам управления и не представлены операторами).

К служебным относятся и т.н. псеводооператоры организации данных (величин).

Переходы в алгоритме бывают следующих классов:

Естественный переход – применяемый исполнителем, когда текущий оператор не указывает переход иного рода; реализуется средствами исполнителя автоматически (напр. в информашине – логикой исполнения любой команды в процессоре).

Безусловный переход – изменяющий порядок следования по сравнению с естественным независимо от алгоритмической обстановки (без к.-л. логического выбора).

Условный переход – изменяющий (или нет) естественный порядок следования в зависимости от алгоритмической обстановки, складывающейся на момент перехода; отражает выбор (принятие решения) исполнителем одной из возможностей продолжения алгоритма (техпроцесса), логически обусловлен значениями к.-л. величин алгоритма.

Если для **естественного** перехода возможен только один тип – к следующему оператору, то **безусловный** м.б. уже различных типов по направлению:

- *прямой* – вперед с обходом части последующих (в естественном порядке) операторов;
- *обратный* – назад с обходом предыдущих (в естественном порядке) операторов;
- *с возвратом* – на начало другого алгоритма, логически связанного с текущим; по достижении его конца возобновляется естественный порядок движения с места перехода.

Для возврата сохраняется состояние выполнения текущего алгоритма на момент перехода.

Формально безусловный переход с возвратом разделяется на следующие особые переходы между визуалами:

Вызов вставки – переход, изменяющий естественный порядок следования с сохранением текущего состояния вызывающего алгоритма и установлением начального состояния алгоритма-вставки (с учётом указаний в основном алгоритме).

Возврат из вставки – переход, восстанавливающий естественный порядок следования с установлением текущего состояния вызывающего алгоритма (с учётом результатов исполнения алгоритма-вставки) и (если нужно) сохранением конечного состояния вставки.

Можно сказать, как в /2/, что «дракон-поезд» совершает «акробатический прыжок», т.е. рабочая точка перемещается из шампура вызывающего визуала (со входа иконы *Вставка*) на начало шампура вызываемого. При возврате совершается «акробатический прыжок» обратно в вызывающий визуал (на выход иконы *Вставка*).

Условный переход, кроме направления (***прямой***, формирующий ветвления и ***обратный***, образующий циклы), можно подразделить по мощности выбора на:

- *дихотомический* (бинарный) – условие которого представляет собой вопрос, на который можно дать утвердительный либо отрицательный ответ (в терминах ДРАКОНА – *да-нетный*); логически речь идет о том, совпадает ли суждение в вопросе о некоторых условиях обстановки алгоритма (техпроцесса) с реальной ситуацией;
- *множественный* – условие которого есть т.н. *тематический* вопрос, на который можно дать определённый ответ из конечного множества; логически это суждение о совпадении условий обстановки (представленных значением переменной/выражения выбора) с тем или иным элементом множества (утвердительный ответ) или о несовпадении ни с каким (отрицательный).

В конечном счёте выбирается то или иное продолжение маршрута. Т.о. имеем четыре типа условных переходов.

Цикл формально есть возвращение на уже пройденный участок маршрута, но в новой алгоритмической обстановке.

Переход можно представлять как прыжок «дракон-поезда» на один «перегон» по линии, соединяющей выход текущей иконы (для развилок – один из выходов) со входом следующей.

Все ли аспекты движения мы рассмотрели? Если исходить из тернарности, то кроме операторов и переходов должно существовать ещё что-то равноправное; однако это «что-то» не должно вводиться искусственно, «ради троичности» структуры. Пожалуй, такое понятие есть — это состояние алгопроцесса, т.е. конкретный набор значений величин, входящих в алгоритмическую обстановку (т.е. по сути — в модель мира, созданную для данной задачи).

Формально предполагается, что исполнение оператора меняет состояние (хотя бы координату рабочей точки алгоритма), а при переходе состояние удерживается неизменным. Фактически мы имеем установившиеся состояния (в том виде, как они формально определены), а между ними — т.н. переходные (не соответствующие формальному определению), поскольку скорость физического движения (в т.ч. процессов в схемах информашины, в организме оператора, в объектах предметной среды) всегда конечна.

Только правильность устройства (организации) и принципа действия носителей состояний гарантирует, что из одного установившегося состояния мы придём к другому установившемуся (и именно к тому, которое подразумевалось при формальном определении движения как алгопроцесса).

Информатическое пространство можно понимать физически – как место представления алгоритма в виде объекта данных (при визуализации – в виде граф-схемы) и логически – как множество сущностей, связанных с описываемой деятельностью.

Известно, что пространство мы воспринимаем в трёх измерениях; поэтому теоретически полная символическая форма алгоритма есть трёхмерное изображение его структуры (практически – объёмный макет). Однако человеку свойственно воспринимать объёмные данные визуально на плоскости (двумерной информационно-оптической сцене, кратко – *диосцене*); поэтому удобно свести трёхмерное изображение к двумерному, напр., описать алгоритм перспективным чертежом. Практически используют не чертежи, а *граф-схемы алгоритмов (ГСА)*; типич-

ные стандарты ГСА, т.н. блок-схемы, не отвечают ряду требований информатического качества (так, множественный выбор изображают как цепочку бинарных).

> Поясним термин ***диосцена***. В общем случае это любой вид, открывающийся человеческому зрению. В более узком смысле это часть наблюдаемого вида, выделенная в естественном окружении (или созданная искусственно) для конкретной цели наблюдателя, в частности – поле существования данных в информатической системе: лист документа, пульт оператора; так мы и будем обычно понимать этот термин.

Математически при переходе к двумерному изображению говорят о т.н. укладке графа на плоскости. Однако на этом «приключения визуала в пространстве» не заканчиваются.

Если ставить задачу автоматизации визуализированного алгоритма, то нужно учесть следующее. Для исполнения алгоритма человеку можно представить данные на диосцене (напр. бумажном документе). Машина же для переработки данных (*информатическая*, кратко – *информашина*, в обиходе называемая «компьютером») не только действует последовательно, но и представляет любые данные одномерно (как содержимое «ленты памяти» из адресуемых ячеек). Поэтому в конце концов структура деятельности сводится к линейной, и требуется *линейная информационно-оптическая сцена* (*лиосцена*) как логическое завершение ряда пространственных форм представления знаний о деятельности.

> Здесь понятие «линейная» относится только к логической мерности изображаемой структуры; физически изображение, конечно, получается по-прежнему в двух измерениях.

Модель машинной записи визуала (т.е. программы) получается переводом его граф-схемы в лиоформу (будем говорить о выкладке двумерного графа в линию) по определённым правилам; в таком виде возможна т.н. «прокрутка» программы (имитация её исполнения человеком, как говорят, «пеше-по-машинному»). Именно из лиоформы возможно машинное кодирование (ассемблирование) визуала; при этом шампур-метод требует модификации, поскольку при выкладке без разрывов (или пересечений) не обойтись.

Отметим, что в ветвлениях каждый горизонтальный участок от излома вертикали до точки слияния фактически визуализирует неявный БП-обход вперёд содержимого главной вертикали, а в множественном – также и других вертикалей (предшествующих данной, если предполагается, что в лиоформе они записаны после неё); в некоторых ТЯП этот переход иногда делался явным, напр. в классическом Си – по ключевому слову "break" в конструкции множественного ветвления на базе "switch-case"[16]; в бинарном ветвлении данный БП обычно неявный, через правило связывания then- и else-частей в if-операторе. В циклах с разветвлённым телом возможен досрочный выход (БП вперёд за пределы тела в конце некоторой ветви, влекущий за собой прекращение выполнения цикла); в Си это было возможно также по "break" с переходом всегда на оператор, следующий за заканчиваемым циклом[17].

В основе цикла неявно лежит БП-обход назад содержимого главной вертикали, визуализируемый как петля цикла; в ТЯП его обычно обозначает закрывающая операторная скобка тела цикла (слово "end" с различными суффиксами или без таковых, правая фигурная скобка); в русской алгонотации употребляется ключевое слово "КЦ"; также возможен досрочный обход (пропуск части тела, следующей за БП); в Си – по ключевому слову "contihue" в циклах while, do и for[18].

На уровне командных моделей указанные БП обязательны для представления соответствующих структур (вставляются при генерации кода конструкции).

1.4.2.3. О структуризации техноязыка

С учётом сказанного выше, выделим основные структурные черты ДРАКОНа по намеченному плану.

Можно выделить следующие ключевые идеи представления времени в ДРАКОНе.

---

[16] См. напр.: Болски М.И. Язык программирования Си. Справочник. – М.:Радио и связь, 1988. – С.36.

[17] Болски М.И. – С.33.

[18] Болски М.И. – С.34.

1. Зависимость версий языка от учёта как содержательного, так и временного аспекта. ДРАКОН-1 предполагает абстрактное время процессов. ДРАКОН-2 не только строго описывает сущность деятельности, но и полностью формализует информатическое время. Для этого он содержит операторы хронизации (*Пауза*, *Период*, *Пуск таймера*, *Синхронизатор*) и взаимодействия (*Параллельный процесс*).

2. Разные возможности учёта времени. Хронизация возможна по принципу паузы либо таймера; в разных практических применениях тот или иной принцип более удобен.

3. Исключение вопросов выполнения визуалов реального времени из языка. И хронизацию, и взаимодействие в принципе также можно визуализировать, но это задачи системного программирования. В автоматизированной системе службу времени (единого и локального) и распределение работ обеспечивает среда исполнения дракон-программ вместе с ОС.Ключевые идеи представления движения в ДРАКОНе:

1. Различные способы формализации холостого оператора:

- паузы – с выдержкой интервала, обозначенного в тексте икон *Пауза*, *Период*;
- таймера – с задержкой до момента по таймеру, указанному в тексте иконы *Синхронизатор*; таймер предварительно определяется (и запускается) иконой *Пуск таймера*;
- ожидания – посредством макроиконы *Цикл ЖДАТЬ*, в котором проверяется наступление заданного события с выдержкой интервала (после выполнения тела цикла, если оно есть).

Для разных способов в ДРАКОНе есть свои реализации. Практически принципы паузы и таймера по-разному соотносятся с тем, что любая икона реально выполняется за конечное время; поэтому в разных ситуациях реального времени тот или иной принцип более удобен.

2. Введение разных форм рабочих операторов, чтобы облегчить визуальное различение разных видов действий, а значит – и понимание визуала в целом (в минимальных блок-схемах используют один тип блока, соответствующий иконе *Действие*). К действиям относятся и служебные операторы (их нужно показывать только в системных процессах, напр. при визуализации управляющих программ информашин).

3. Формализация всех типов перехода, причём графически они представлены (визуализированы) следующим образом:

- естественный – как «выполняемые» соединительные линии, направленные по вертикали от начала схемы к концу и лежащие на одной оси (в терминах ДРАКОНа – по шампуру).
- условный с дихотомическим выбором – как макроикона *Развилка*, где изменение порядка следования от естественного отражено смещением от *главной* вертикали (проходящей через икону *Вопрос*) всегда вправо через горизонтальный участок на новую (*побочную*) вертикаль и последующим смещением обратно к точке слияния с главной; со множественным – как макроикона *Переключатель*, составленная из икон *Выбор* и *Вариант* и имеющая ряд побочных вертикалей и точек слияния (бинарных, т.е. с двух входных направлений); обратный – как икона *Петля цикла*, играющая роль точки слияния перед развилкой.
- безусловный – посредством икон *Адрес* и *Имя ветки*, включённых в икону *Петля силуэта*, а БП с возвратом (вызовом другого алгоритма) – как икона *Вставка*.

Принцип главной вертикали проясняет разветвлённую структуру и создает условия для её неформального упорядочения (т.н. эквивалентных и равносильных преобразований визуалов).

Переключатель логически – это развилка со многими выходами, и графика между выходом иконы *Выбор* и входами икон *Вариант* невыполняемая (и недоступная для изменений); представляя дракон-схему как мультиграф, эти иконы можно считать единой вершиной.

Употребление безусловных переходов в ДРАКОНе строго формализовано петлёй силуэта и ограничено; в частности, не допускается БП назад и с пересечением пути другого БП. Логически пространство деятельности в ДРАКОНе – это множество визуальных операторов, т.н.

икон, образующих ***алфавит*** техноязыка. Некоторые иконы употребляются не самостоятельно, а лишь в составе конструкций-*макроикон* (входящих как подсхемы в дракон-схему); макроиконы образуют исходный ***словарь*** техноязыка. Иконы и макроиконы описаны в п.1.2.2 Приложения 1 в виде толкового словаря (с определениями).

Для конкретного назначения также строятся свои пространства имён действий и объектов деятельности (для подъязыков, образующих «костюм»).

Создатель техноязыка указывает на необходимость единых правил построения объектных имён, информативных и удобных для восприятия; для этого нужна и достаточная длина имени: «...множество 32-символьных идентификаторов образует весьма выразительный, хотя и своеобразный, язык, законы и правила оптимизации которого ещё предстоит открыть, обсудить и подвергнуть экспериментальной проверке.» /1, с.163/.

Ключевые пространственные идеи ДРАКОНа.

1. Визуализация любой мыслимой структуры деятельности на плоскости как дракон-схемы (возможно – как ряда логически связанных дракон-схем). Для этой цели создан формальный шампур-метод построения дракон-схем, при следовании которому исключены неточности, двусмысленности описания структуры на диосцене. В этом методе кроме алфавита языка (икон) и словаря (макроикон) заранее даны исходные формы дракон-схем, т.е. допустимые структуры предложений на техноязыке и введены определённые правила, по которым строятся (логически выводятся) любые возможные структуры визуалов.

2. Эргономизация изображения алгоритма. С этой целью вводятся некоторые базовые правила изображения структуры:

- соединительные (объединяющие) вершины не показываются явно, а заменяются простым слиянием линий;
- соединительные линии естественного перехода расположены на одной оси-вертикали, направленной сверху вниз;
- направление линий по шампуру (естественный порядок исполнения операторов) совпадает с естественным порядком чтения и потому специально не указывается, что упрощает восприятие схемы; явно указывается (стрелками на конце связей) только направление против шампура (при разрешённом безусловном переходе и условном переходе в начало цикла).

Они же служат общими правилами перехода от блок-схем к дракон-схемам (вывода вторых из первых). Для отдельных конструкций справедливы также частные правила перехода (вывода).

Также введены ***топологические запреты*** на:

- пересечения и обрывы связей на дракон-схемах .
- обратное (снизу вверх) направление следования;

Пересечения преодолеваются за счёт найденного Пароджановым способа «укладки» произвольных ГСА на плоскости (в форме силуэта) с введением в дракон-схему безусловных переходов, причём по строго формальным правилам, в отличие от текстовых языков.

Разрывы исключаются использованием вставок, т.е. фактически иерархической декомпозицией алгоритма.

Обратное следование исключается перекомпоновкой схемы (смещением фрагментов друг относительно друга так, чтобы исчезли участки, следующие снизу вверх, и связанные с ними горизонтали). В более широком смысле на дракон-схеме вообще не должно быть т.н. неоправданных изломов, т.е. лишних «ступенек» в побочных вертикалях и петлях циклов.

На практике все компоненты импер-знаний было принято описывать посредством текста; поэтому существуют в основном текстовые языки программирования (ТЯП). При этом структуры управления в ТЯП описываются нестрого из-за неформальных правил употребления безусловных переходов. В техноязыке же введены однозначные ограничения на такие перехо-

ды, что органично совмещается с графическим представлением структур алгоритмов. Именно поэтому ДРАКОН позволяет алгоритмизовать формально (так, чтобы получить однозначную структуру) и в то же время эргономично, т.е. удобно для любого человека. Более подробно о текстовом и графическом структурном кодировании алгоритмов см /1, Гл.16/.

3. Эргономизация изображения визуальных операторов. Для этого создателем языка:

- ряд икон компонуется из частей-фигур, что полнее визуализирует их логику (так, в параллельном процессе наложение блоков действий отражает взаимосвязь пары процессов).
- применены только такие формы икон и их частей, которые и наглядны сами по себе, и удобно и экономично вмещают текст; поэтому по сравнению с блок-схемами некоторые формы блоков получают новые значения (к примеру, форма-трапеция – как основа хронизаторов реального времени), а другие (скажем, ромб) не используются.
- из графики маршрутов максимально исключены элементы, представляющие интуитивно ясные (из закономерностей восприятия данных) вещи, чтобы не загромождать диосцену.

Т.к. графика икон составлена почти полностью на основе графоэлементов стандартных блок-схем, можно рисовать вручную (или в программном редакторе) «по привычным трафаретам».

Очевидно, эргономизацию мы должны провести и для текста икон; принятый подход будет очевиден из иллюстраций, поэтому укажем лишь его основные положения:

- выбор разной гарнитуры шрифта для действий, объектов, а также текстов пояснений (элементов КогниСтиль);
- единый формат длинных имён, отличающийся от предложенного в /1, Гл.11/.

Изменённый формат имён выбран по следующим соображениям. Если строго следовать единству представления деятельности на разных уровнях формализации, то объекты должны иметь один и тот же формат имён от первоначальной визуализации до исходного текста программы. Однако в современных объектных языках программирования точка используется для иных целей, чем просто разделитель частей сложносокращённого имени: обычно она отделяет имена свойств объекта друг от друга и от его имени. Поэтому части имени объекта (свойства) мы пишем слитно, а для разделения используем их капитализацию. Это облегчит и восприятие подстановки в тексты объектных дракон-программ кратких имён (алиасов) вместо длинных, как предложено в /1, с.185/.

На этом предварительные рассуждения о методологии ДРАКОН можно завершить. Структуризация позволила нам одновременно раскрыть семантику (значение) ряда операторов техноязыка, поэтому далее мы можем сосредоточиться на описании его синтаксиса.

Здесь мы ввели базовые термины формализации знаний и ДРАКОНа; остальные будут вводиться по ходу документа, преимущественно на рисунках. Кроме «взрослых» терминов, мы также пользуемся «детскими», которые создатель ДРАКОНа ввел в пособии /2/.

Данный курс ориентирует на визуализацию в среде машинного документооборота (офисных пакетов); поэтому он в основном состоит из иллюстраций, подготовленных в редакторе рисунков. Кроме того, как можно видеть, предпринята эргономизация текста на базе возможностей форматирования в используемом офисном пакете (правда, во многом интуитивная).

Следуя положениям когнитивной эргономики, а также преследуя цель краткого описания ДРАКОНа, мы будем по возможности шире использовать графическое представление и максимально включать в иллюстрации текст, активно применяя метод КогниСтиль, более подробно о котором здесь.

1.4.3. Визуализация знаний: эргономичность и формальность

1.4.3.1. Метод Когнитивный Стиль: визуальное оформление

Для эргономичной визуализации произвольных описаний (математических и пр.) создатель техноязыка предложил метод «Когнитивный стиль», независимый от степени абстрактности и когнитивного качества основного языка описания.

КогниСтиль служит средством визуализации произвольных (неформальных и неалгоритмических) знаний. Метод основан на применении вспомогательных блоков. Основные приемы описаны в /1, Гл.19/. Анализируя их, ключевые идеи КогниСтиля можно определить как:

- зонирование диосцены путём разного оформления её частей и/или наложения на неё ограничителей зоны; в результате содержание сцены оказывается структурированным в соответствии с некоторой *информационной моделью*, и сцена легко "читается", подобно хорошо оформленному пульту оператора;
- введение дополнительных пояснений и оценок основного содержания диосцены, которые максимально полно передают *концептуальную модель*, т.е. представление автора сцены о её интерпретации и применении (в т.ч. другими людьми);
- установление дополнительных связей между зонами – служит для выражения как информационной, так и концептуальной модели, позволяя проследить содержание диосцены в динамике (например, ход математических преобразований).

Применение Когни Стиль организует документ как систему графики и текстов.

> Очевидно, чем лучше структурирована информация и чем яснее изложена концепция диосцены, тем эффективнее передаётся то, что хотел сказать её автор. Это значит, что другой человек, владеющий данным методом, сможет быстрее и точнее извлечь из данных диосцены знания, т.е. понять её смысл.

По материалам из /1/ составлен алфавит КогниСтиль, приводимый в п. 1.2.1 Приложения 1 (также с краткими описаниями элементов).

В данном курсе КогниСтиль развит и допускает *диофильмы*, или последовательности диосцен-кадров, имеющих графические элементы указания взаимосвязи. Диофильмы можно использовать для удобства последовательного изложения материала либо для представления по частям содержания, не входящего целиком на один носитель данного формата. Последнее, как справедливо указано в /1/, эргономически неоправданно, однако в случаях, когда неудобно увеличивать формат носителя, а содержание не структурируется иерархически, приемлемо.

1.4.3.2. Графит-метод: визуальное конструирование

Чтобы отразить динамику состояния произвольных объектов (и, в частности, визуализировать операции сочинения визуалов), здесь предлагается графит-метод (от ГРАФика И Текст). Центральная идея метода – показать формально сочинение произвольных структур из заранее заданных элементов, подобно тому, как визуализируется сборка физических систем из деталей и узлов. Она раскрывается через две ключевых идеи:

1. Определять содержание текстоэлементов визуальных описаний на одном из известных метаязыков. В предлагаемой версии метода выбран РБНФ-язык в редакции, описанной в п/п 1.1.1.1 Приложения 1.

2. Визуализировать базовый набор операций конструирования. Предлагается два способа для этого. Во-первых, можно непосредственно показывать операции из базового набора; так делается на сборочных чертежах и схемах в конструкторской документации уже много лет. Во-вторых, можно рассматривать элементы как лексические единицы некоего языка и выразить синтаксис этого языка визуализацией одного из известных метаязыков описания синтаксиса.

По некоторым представлениям, базовые операции можно свести к прибавлению/вычитанию элементов и преобразованию отношений (взаимосвязей) между ними.

Итак, знаки и конструкции графит-метода управляют связанными знаками и конструкциями визуальных моделей.

При любом способе важную роль играет т.н. эллипсирование - выделение повторяющихся частей конструкции, для которых можно подробно показывать одну часть-образец, а остальные вхождения обозначать сокращённо. Это также давно используется в конструкторской документации. Графические сокращения в настоящее время определены в п. 1.2.3 Приложения 1.

По первому способу предлагается визуализировать операции как диоструктурные операторы (диостропы), приведённые в п/п 1.2.1.3 Приложения 1. Основные правила :

- Объекты диостропов располагаются как внутри контура последних, так и вне его; в последнем случае показывается их участие в той или иной операции связями-отношениями.
- Приложение операций показывается направляющими линиями (обычно кривыми со стрелкой); они идут от выбранных на объекте реперных точек к точкам стыка на структуре.
- Возврат направляющих в некоторые места самого объекта означает последовательное повторение операции (прибавления того же элемента после уже прибавленных); если число повторов ограничено, то это специально указывается пояснениями.

Как правило, для абстрактных систем аксиоматически даются начальные формы-заготовки для создания тех или иных классов (подмножеств) структур.

По второму способу используются расширенные БНФ над графоэлементами и подсхемами, визуализируемые с участием знаков РБНФ-графит (см. п/п 1.3.1.4 Приложения 2).

Эллипсирование оформляется знаками-ограничителями. Текстом ограничителей могут являться РБНФ-выражения, которые тогда относятся к ограничиваемым элементам (подструктурам) и тем самым заменяют знаки РБНФ-графит.

Графит-метод также используется для описания состава тех или иных сущностей предметной области.

2. ИСТОЧНИКИ

Документы для ссылок

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

2. Пароджанов В.Д. Занимательная информатика. - М.: Дрофа, 2007.

3. Баранцев Р.Г. Становление тринитарного мышления. – М.-Ижевск: НИЦ «Регулярная и хаотическая динамика», 2005.

4. Фридланд А.Я. Информатика: процессы, системы, ресурсы. - М.: БИНОМ. Лаборатория базовых знаний, 2003.

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.

Полезные ресурсы

mgopu.ru/PVU/2.2/Sprint-Inform — здесь можно изучить современную информатическую терминологию, введённую в /4/.

forum.oberoncore.ru - на форуме обсуждается среди прочего и ДРАКОН как элемент культуры и практический инструмент.

Усилители интеллекта — рассылка, посвящённая средствам облегчения умственного труда и в частности, техноязыку.