УДК 381.3.06

ПРОСТРАНСТВЕННАЯ МОДЕЛЬ ОЦЕНКИ ЭВОЛЮЦИИ МЕТОДОВ ВИЗУАЛЬНОГО ПРОЕКТИРОВАНИЯ СЛОЖНЫХ СИСТЕМ

А. И. Власов

Рассмотрены результаты системного анализа применимости визуальных методов моделирования сложных систем при использовании различных проектных методик. Предложена пространственная модель для оценки эволюции и эффективности широко распространенных методов разработки визуальных схем сложных технических систем. Отмечено, что известные подходы визуального анализа характеризуются изолированностью при проектировании на разных этапах жизненного цикла изделий электронной техники. Как правило, они обеспечивают описание отдельных модулей сложных систем и не имеют средств миграции данных моделей между различными уровнями проектных процедур. Даны оценки важности учета оси "регламент" в общей пространственной модели эволюционного проектирования сложных систем с использованием визуальных методов.

Ключевые слова: визуальное проектирование, сложные системы, конструкторско-технологическая информатика.

ВВЕДЕНИЕ

При решении задач конструкторско-технологического проектирования изделий электронной техники на первый план выходит необходимость эффективного управления знаниевой информацией и системный подход. Эффективное управление знаниями о новых системных, конструктивных реализациях, новых материалах и способах их синтеза требует минимизации количества операций, необходимого для их обработки, что, в свою очередь, приводит к необходимости компактного описания их формализованного представления. Научное направление "Конструкторско-технологическая информатика в радиоэлектронике" охватывает исследования и разработку методов и способов решения системотехнических, конструкторских и технологических задач по синтезу сложных радиоэлектронных систем в условиях комплексной информационной поддержки жизненного цикла изделий радиоэлектроники. В его основе лежат три основных понятия [1]:

— *конструкция* (лат. constructio — строение, устройство, построение, план, взаимное расположение частей; англ. — construction; нем. — die Konstruktion; фр. — construction);

— *технология* (греч. techne — исскусство, мастерство + logos — понятие, учение; англ. — technology; нем. — die technologie; фр. — technologie);

— *информатика* (ср. нем. Informatik; англ. Information science; фр. Informatique; англ. computer science — компьютерная наука — в США; англ. computing science — вычислительная наука — в Великобритании) — наука о способах получения, накопления, хранения, преобразования, передачи, защиты и использования информации (в нашем случае — конструкторско-технологической).

Под конструкцией электронных средств (ЭС) понимается совокупность элементов и деталей с различными физическими свойствами и формами, находящимися в определенной пространственной, механической, тепловой, электромагнитной и энергетической взаимосвязи [1]. Эта взаимосвязь определяется системотехнической, схемотехнической, конструкторской и технологической документацией и обеспечивает выполнение электронной аппаратурой (ЭА) заданных функций с необходимой точностью и надежностью в условиях воздействия на нее различных факторов: эксплуатационных, производственных, социальных.

Технология производства (или технологический процесс) — основная часть производственного процесса, заключающаяся в выполнении определенных действий, направленных на изменение исходных свойств объекта производства (в нашем случае ЭА) и достижения им определенного состояния, соответствующего технической документации [1]. Конструирование и технология производства являются, с одной стороны, отдельными частями сложного процесса создания ЭА, а с другой, не могут выполняться в отдельности, без учета взаимосвязей между собой и с другими этапами разработки, являясь этапами более общего процесса "разработка — производство — эксплуатация — утилизация" (жизненного цикла изделия). Как конструирование, так и технология определяют в конечном итоге общие потребительские свойства ЭА [1].

Информатика решает задачи обработки информации с использованием вычислительных систем и сетей. Термин информатика возник в 1960-х годах во Франции для названия области, занимающейся автоматизированной переработ-

кой информации, как слияние французских слов *information* и *automatique* (F. Dreyfus, 1972). Тематика исследований в информатике обширна и постоянно расширяется: теория вычислимости и искусственный интеллект, теория сложности вычислений, информационные структуры и базы данных, социальный аспект развития информационных систем, языки программирования, представление знаний и т. п.

С широким распространением встраиваемых электронных систем важной задачей становится правильное применение методов проектирования сложных систем, характеризующихся своими интеллектуальными ресурсами, методами и средствами формализации, хранением обработки и передачи информации в них. Что касается понятия сложной системы, то до последнего времени единого определения этого термина нет. Известны различные подходы и предложены различные формальные признаки его определения. Так, Г. Н. Поворов предлагает относить к сложным системы, имеющие 104—107 элементов, к ультросложным — системы, состоящие из 107—1030 элементов и к суперсистемам — системы из 1030—10 200 элементов. Основной недостаток такого подхода в том, что данное определение сложности является относительным, а не абсолютным. Английский ученый С. Бир предлагает к сложным системам относить системы, описываемые на языке теоретико-вероятностных методов (мозг, экономика, форма и т. п.) [2]. Наиболее четким определением сложных систем является определение, данное, в [3]: сложной системой называется система, в модели которой недостаточно информации для эффективного управления этой системой.

Таким образом, признаком простоты системы является достаточность информации для ее управления. Если же результат управления, полученный с помощью модели, возможно, будет неожиданным, то такую систему относят к сложной. Для перевода системы в разряд простой необходимо получение недостающей информации о ней и включение ее в модель. Одним из инструментов такого преобразования "сложная система" ⇔ "простая система" являются методы визуального моделирования [4—9].

Визуальное представление предметной области, в том числе и описание сложных систем в частности, строится на принципах когнитивности, конвергенции (от английского convergence — схождение в одной точке) и инкапсуляции [4]. Широко распространенные методы разработки визуальных схем сложных технических систем предоставляют удобный инструментарий для создания, изменения и редактирования визуальных моделей в цепочке "модель—диаграмма—компонент" [4—12]. Под визуальным моделированием будем понимать совокупность методов, которые используют метафоры визуализации, предлагают представлять объект с разных точек зрения и могут применяться для разработки и эволюции объекта моделирования [1, 2]. Метафоры визуализации — это сопоставление абстрактных или реальных объектов зрительно воспринимаемым образам. Языки визуального моделирования в свою очередь образованы фиксированными наборами метафор и правилами построения из них визуальных моделей.

Следуя классическому унифицированному процессу проектирования [5—7], на каждом из этапов проектирования применяются визуальные модели с соответствующим уровнем абстракции и детализации экспертизы. Так, на начальных стадиях проектирования используется абстрактное визуальное моделирование, представляющее рассматриваемые системы в обобщенном виде (концептуальное моделирование). На этом этапе используются менее формализованные методы, так как его основная цель — создать самую простую обобщенную модель предметной области.

Следующий этап включает в себя разработку моделей с точки зрения структурно-функционального и операционного подходов. На этом этапе чаще всего используются IDEF модели [6—8].

Моделирование информационных потоков сводится к построению моделей объектов и потоков данных, а также структуры информационной модели и структуры базы данных и, если необходимо, базы знаний.

На заключительном этапе, как правило, разрабатывают объектно-ориентированные модели системы с помощью процедур RUP, на основе которой создают каркас системы, соответствующей реализуемым процессам [5, 7].

В соответствии с описанной выше обобщенной последовательностью системного анализа процессов и систем на текущий момент используются различные инструментальные средства и методики, при этом средства взаимодействия, миграции данных и знаний моделей различных типов от одних инструментариев к другим практически отсутствуют. Можно утверждать, что известные визуальные походы характеризуются высокой степенью изолированности при проектировании разных этапов жизненного цикла (рис. 1). Следовательно, не решенной до сих пор, остается *проблема "семантического разрыва"*, которая заключается в необходимости обеспечения взаимной

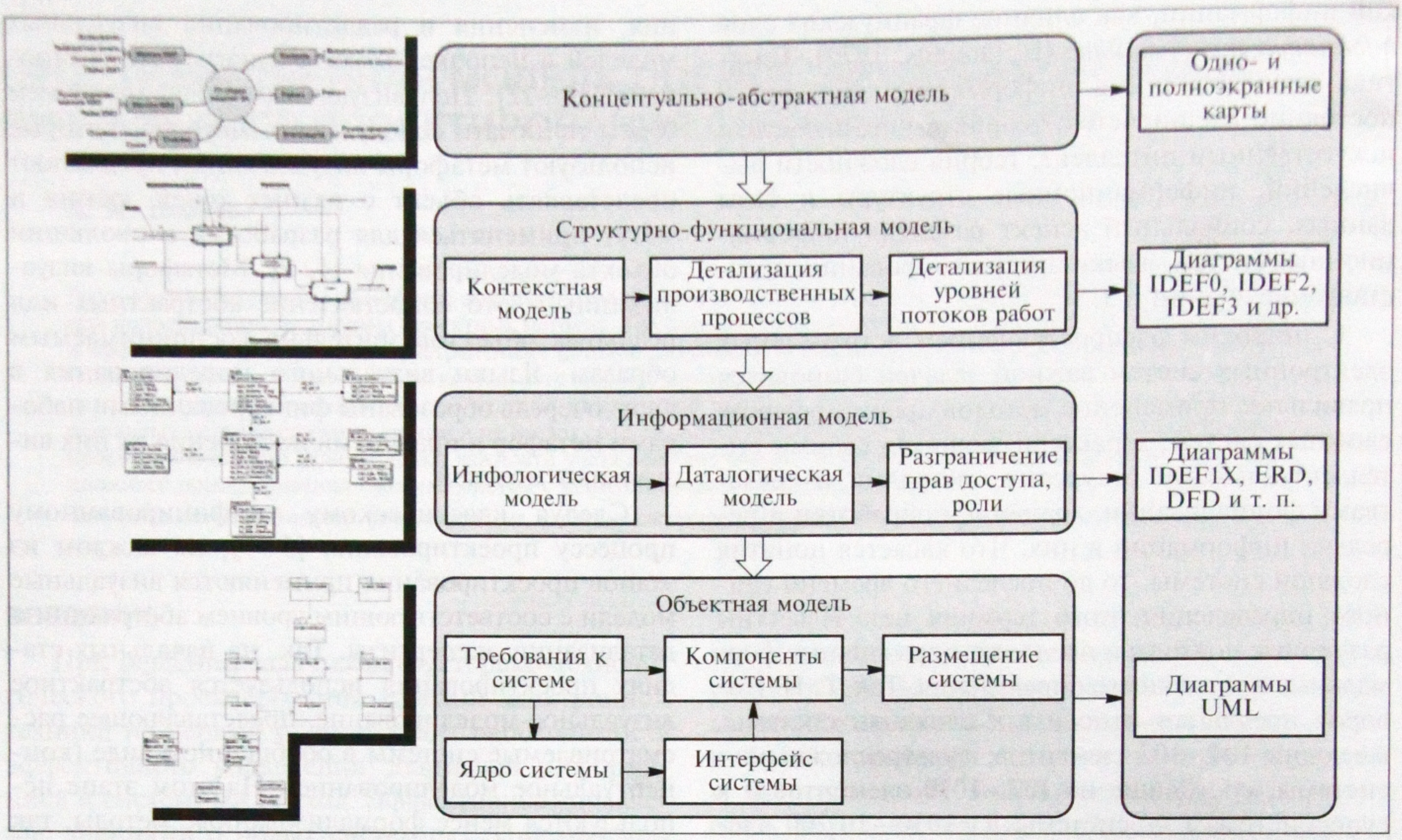

Рис. 1. Эволюция моделей предметной области по различным уровням экспертизы

миграции данных из визуальных моделей разного уровня абстракции (экспертизы) при выполнении условия однократности ввода характеристик рассматриваемых объектов, систем и процессов [4]. Также следует отметить проблему синхронизации моделей при их групповой разработке. Это приводит к снижению эффективности применения визуальных схем, в частности, в процессе анализа отдельных компонентов предметной области, сложно сочетающихся между собой.

Постановка проблемы. Основным недостатком существующих подходов к созданию визуальных схем сложных систем является их фрагментация, изолированность применения визуального анализа и проектирования на разных этапах жизненного цикла изделий. Это приводит к очаговой эффективности применения визуальных схем. В качестве главной причины данной проблемы можно указать наличие так называемого семантического разрыва, т. е. когда описания отдельных областей предметной области сложно сочетаются между собой.

Учитывая, что объекты радиоэлектроники имеют высокий уровень взаимного влияния всех этапов жизненного цикла изделия, то для создания охвата всех его этапов требуется использование простого и эффективного подхода с применением интуитивно понятных инструментов проектирования. Легкость восприятия разработчиком визуальных схем (“один рисунок заменяет тысячу слов, а одна схема тысячу рисунков”) выводит на первый план необходимость решения проблемы формирования эффективных способов описания, хранения и передачи знаний о сложных системах между уровнями моделирования (экспертизы) в виде визуальных схем со встроенным внутренним универсальным метаописанием [5—7].

Целью настоящей работы является исследование пространственной (3D) модели оценки эволюции методов и средств визуального моделирования, обеспечивающей замкнутый жизненный цикл процесса разработки визуальных схем и базирующейся на иерархически связанной эволюционной компонентной модели, включающей упорядоченную совокупность метаобъектов.

При этом следует разделять методы, методики и методологию визуального проектирования. Методология визуального проектирования как понятие в целом обозначает учение о структурах, логическую организацию, принципы построения, формы и способы научного познания посредством визуальных схем. Метод (греч. — путь исследования, теория, учение) — это решение конкретной задачи, совокупность приемов или операций прак-

тического или теоретического освоения действительности. Методика — это последовательность выполнения определенных действий для решения поставленной задачи в рамках определенного метода. Методология визуального проектирования — это отрасль научного знания, изучающая средства, предпосылки и принципы организации познавательной и генерационной деятельности посредством визуальных средств, приемов, принципов и подходов, используемых в различных видах проектной деятельности [2, 3, 7, 8, 12—14]. В работе предложен комплексный подход к оценке эффективности визуальных методов проектирования, включая средства онтологической связи схем разного уровня экспертизы, редактирования схем, трансформации визуальных моделей и их регламента. В рамках работы предложен способ описания визуальных моделей сложных технических систем на основе метаязыка ViXML, который позволяет представлять в едином формате визуальные схемы на всех этапах исследования предметной области [4, 11].

Практическая ценность работы заключается в возможности использования ее положений в конструкторско-технологической деятельности, характеризующейся применением визуальных схем на всех этапах процесса проектирования. Результаты работы могут быть использованы при разработке перспективных CASE средств, обеспечивающих разработку визуальных схем на всех необходимых этапах конструкторско-технологической деятельности, что позволяет реализовать проектные процедуры в полном объеме, без потери каких-либо важных элементов проекта и без превышения установленных сроков.

ОБОСНОВАНИЕ КРИТЕРИЕВ ОЦЕНКИ ЭФФЕКТИВНОСТИ ВИЗУАЛЬНЫХ МЕТОДОВ ПРОЕКТИРОВАНИЯ

Современный характер исследования сложных систем определяет проблемы анализа протекающих процессов, трудоемкости построения формального описания, хранения и обработки знаний об их объектах и процессах. Поиск решения данных проблем в настоящий момент построен вокруг использования визуального моделирования в качестве основного инструмента генерации, хранения и обработки знаний в рассматриваемой предметной области. Под инструментарием *визуального моделирования* будем понимать комплекс методов и средств, сочетающих графическую нотацию (библиотеку элементов визуального языка, графический редактор с репозиторием, навигатором модели и т. п.) и текстовую нотацию, связанную с компонентами визуального языка, в которой представлено описание атрибутов данных компонентов. Сочетание графической и текстовой нотации должно обеспечивать синтез полнотекстовых документов по разработанным моделям. В последнее время популярность приобретает DSM-подход (Domain-Specific Modeling — предметно-ориентированного моделирования). В настоящее время на рынке существует ряд средств для разработки графических редакторов с возможностью задать собственную графическую нотацию (создать в автоматизированном режиме сам графический редактор с репозиторием и навигатором модели). К таким DSM-пакетам можно отнести: Eclipse/GMF, Microsoft DSL Tools и т. п. Основной недостаток данных решений — практически полное отсутствие текстовой нотации модели.

Наиболее распространенными методами визуального моделирования являются: абстрактно-концептуальные, структурные — IDEF (Integrated DEFinition), ARIS (Architecture of Integrated Information Systems), объектно-ориентированный анализ и проектирование (ООАП) (RUP — Rational Unified Process), DSM-подход (Domain-Specific Modeling — предметно-ориентированного моделирования) и MSF-подход (Microsoft Solutions Framework). На данный момент создано множество качественных, удобных и уже широко распространенных визуальных методов, основным недостатком которых остается слабая связь методов между собой и проблемы с экспортом/импортом данных. Обобщенная классификация методов визуального моделирования приведена на рис. 2. Вопросами оценки их эффективности посвящен ряд работ [5—7, 10—17].

При выборе методов визуального проектирования возникает вопрос критериев оценки и сравнения возможностей, удобства и функциональности различных методов. Так, в работе [14] Алексеем Закисом предложена 2D-методика анализа визуальных методов проектирования по оценке степени формализации и итеративности процесса разработки. За счет правильного выбора показателей возможно существенное снижение стоимости и/или сроков разработки при гарантии необходимого качества создаваемых систем [14], однако в явном виде в 2D-варианте не дается оценка показателей регламента проектирования.

Наиболее конкурентоспособным будет метод — простой по восприятию, интерпретации, обучению, средствам реализации и имеющий развитые механизмы экспорта/импорта. В настоящее время наиболее значимыми и удобными для сравнения характеристиками следует считать: итеративность

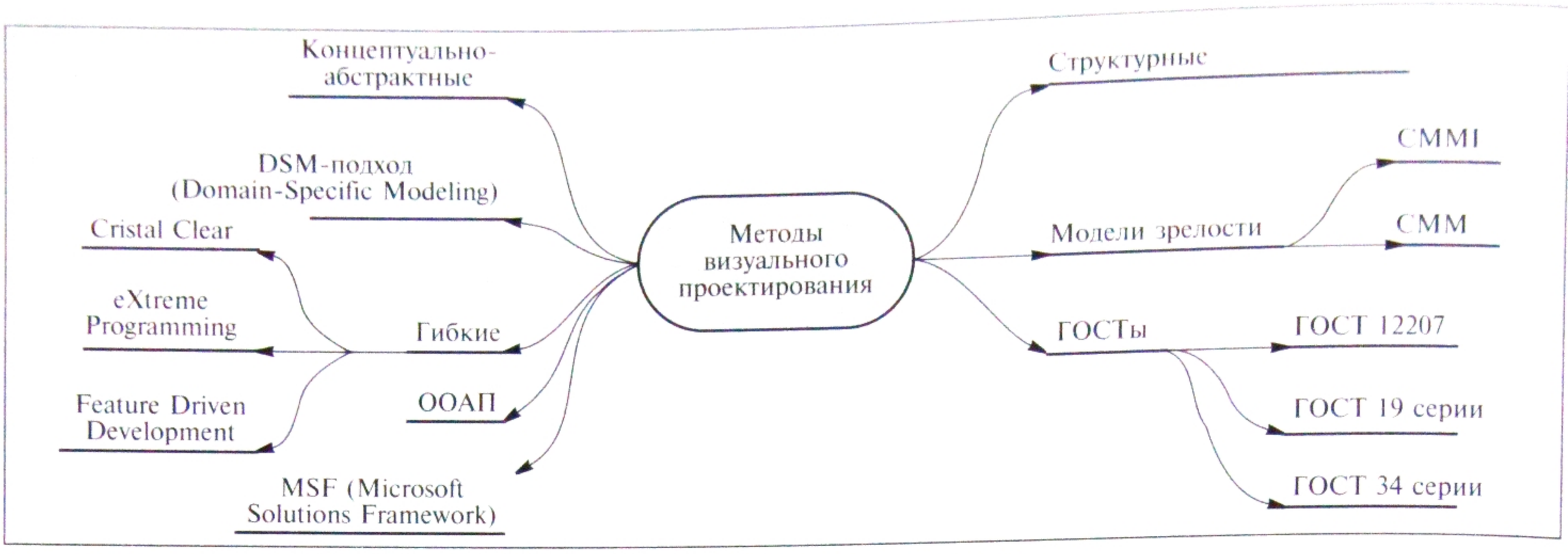

Рис. 2. Основные методы визуального проектирования сложных систем

разработки, гибкость степени формализма и регламент проведения разработки [10, 14]. Для сравнения методов можно использовать трехмерное пространство свойств с отложенными по осям вышеприведенными ключевыми характеристиками (рис. 3), представляющее абстрактное пространство свойств рассматриваемых методов. Соответствие осей и характеристик приведено в табл. 1.

Точка в пространстве свойств отражает набор характеристик конкретного метода в пространстве свойств, которое определяет направления его возможной дальнейшей эволюции. Все методы предоставляют некоторую в большей или меньшей степени свободу варьирования ключевых характеристик процесса моделирования в зависимости от различных параметров проекта (масштаб, критичность, количество участников, долговечность, требования заказчика и др.).

Итерационность разработки. Выделяют каскадные и итерационные методы проектирования [14]. Каскадные методы проектирования исходят из того, что разработка делится на фазы, каждая из которых характеризуется своим набором операций. В отличие от них итеративный подход разбивает разработку на несколько итераций, в ходе каждой из которых выполняется практически весь набор операций и создается реальная работающая система (прототип) с все более развитыми функциональными возможностями.

При каскадном подходе сначала происходит выявление всех требований к проекту и их анализ. Затем проектная группа приступает к проектированию системы (чаще всего, "сверху вниз", разбив создаваемую систему на подсистемы и далее детализируя их до атомарного уровня) [10].

При итерационном подходе разработка разбивается на относительно короткие итерации [10, 12—14]. Так, в самой первой итерации, еще до выявления всех требований, может начаться разработка прототипа, на котором проверяются основные архитектурные решения. По мере детализации требований на отдельные подсистемы или компоненты на последующих итерациях начинается их проектирование. Разработанные "начерно" подсистемы и компоненты собираются в единую систему (не дожидаясь завершения разработки всех подсистем) и тут же начинается их системное тестирование.

Преимущества итерационного подхода — возможность учета дополнительных требований (либо измененных старых), выявление возможных ограничений (связанных с принятыми техническими решениями) на ранних стадиях проектирования. При таком подходе руководство наиболее полно готово к изменениям [12—14].

Недостатком итеративного подхода является увеличение сроков проектирования в связи с проведением всех работ в каждой итерации.

Соответственно, преимущество каскадного подхода в более коротких сроках проектирования (что справедливо только для маленьких и средних проектов).

Недостатками каскадного подхода являются невозможность полного учета всех пожеланий за-

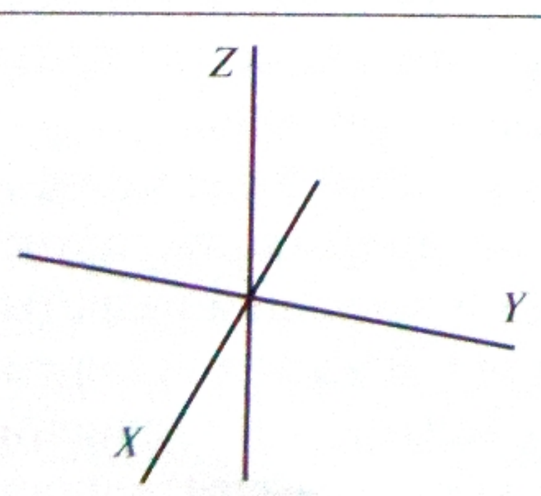

Рис. 3. Трехмерное пространство свойств методик визуального моделирования:

X — уменьшение формализма; *Y* — увеличение регламента; *Z* — приближение к каскадному подходу

казчика, многие из которых могут возникнуть уже после начала проектирования, а также риск появления критической ошибки на ранней стадии проектирования, что приведет к краху проекта в целом.

Уровень формализации. Методы проектирования различаются не только составом и характером моделей, которые разрабатываются в ходе проекта, они различаются тем, насколько формализованно ведется разработка. Каждая модель визуального проекта включает, как правило, текстовую и графическую нотации. На каждом этапе проектирования требования к составу, структуре и полноте графической и текстовых нотаций должны быть сформированы исходя из принципа "минимальной достаточности", так как это определяет скорость и трудоемкость разработки.

Немаловажным аспектом является выбор эффективного способа (формата) представления информации, обеспечивающего сохранность информации о рассматриваемых системах и переносимость (экспорт/импорт) этой информации. Хранение информации в БД не обеспечивает переносимость в явном виде на физических носителях и требует знаний языков манипулирования данными (ЯМД) для извлечения информации. Бинарные файлы не могут быть прочитаны без соответствующего ПО или же требуют знания специальных алгоритмов их записи/чтения. Текстовое представление информации лишено подобных недостатков.

Таблица 1

Оси трехмерного пространства свойств [10, 13, 14]

Ось	Характеристика	Описание
X	Уровень формализации	Включает состав разрабатываемой документации, аккуратность и структурность ее оформления. Положительное направление оси соответствует уменьшению уровня формализации
Y	Жесткость регламента	Указывает на жесткость используемого регламента проведения процедур рецензирования проектной документации в соответствии с принятыми правилами, ее одобрения и передачи. Положительному направлению оси соответствует усиление регламента, вплоть до точного указания правил и последовательности проведения проектирования. Отрицательному направлению соответствует ослабление регламента, вплоть до полного отсутствия каких либо правил и предоставления свободы действия разработчику
Z	Итерационность подхода	Положительному направлению оси соответствует приближение подхода к каскадному, который подразумевает разбиение разработки на фазы. Отрицательному направлению соответствует увеличение итерационности подхода, что означает циклическую разработку в несколько итераций, каждая из которых проходит через все фазы

Проанализируем основные текстовые форматы хранения данных: CSV (от англ. Comma Separated Values — значения, разделенные запятыми), RTF (от англ. Rich Text Format — формат обогащенного текста), XML (от англ. eXtensible Markup Language — расширяемый язык разметки) [15].

- CSV — текстовый формат, предназначенный для представления табличных данных. Каждая строка файла — это одна строка таблицы. Значения отдельных колонок разделяются разделительным символом (delimiter) — запятая (,) или точка с запятой (;). Используемый символ разделителя зависит от установленной в системе настройки. В США это запятая, а в России — точка с запятой, так как запятая используется для дробных чисел (в отличие от США, где это — точка). Значения, содержащие зарезервированные символы, такие как: запятая, точка с запятой или новая строка обрамляются символом двойные кавычки ("); если в значении встречаются кавычки, они представляются в файле в виде двух кавычек подряд.
- RTF — проприетарный межплатформенный формат хранения размеченных текстовых документов, предложенный Microsoft и Adobe в 1982 г. как метатэговский формат. С тех пор спецификация формата несколько раз изменялась. RTF-документы поддерживаются всеми современными текстовыми процессорами.
- XML — рекомендованный Консорциумом W3C (W3C, http://www.w3c.org) язык разметки, предназначенный для хранения структурированных данных и обмена информацией между программами, являющийся упрощенным подмножеством языка SGML (Standard Generalized Markup Language).

Результаты сравнения перечисленных форматов сведены в табл. 2.

Использование формата XML несет в себе больше преимуществ, чем RTF или CSV. Единственным недостатком является избыточный синтаксис, но это компенсируется достоинствами. XML является предпочтительным, так как обеспечивает представление как табличных данных (такие как реляционные данные из базы данных или больших таблиц), так и псевдоструктурированных данных (такие как Web-страницы или полнотекстовые документы). XML фактически пред-

ставляет собой свод общих синтаксических правил описания структур документов с помощью тэгов разметки (дескрипторов, взятых в угловые скобки '<' и '>'). При этом сами теги изначально не определены и должны быть созданы разработчиком. Отсюда одно из главных преимуществ XML — он неограниченно расширяем. Это привело к созданию огромного числа подмножеств XML, например XHTML (семейство языков разметки веб-страниц, (http://www.w3.org/TR/xhtml11/)), XMCD (этот язык разметки используется для описания данных программного пакета Mathcad, (http://www.ptc.com/products/mathcad/)) [15].

Как подмножество XML разработан и язык описания знаний о сложных технических системах VIXML [4, 11], который обеспечивает гибкие возможности по настройке уровня формализма анализируемых объектов и процессов. Преимуществом минимально формализованного подхода является значительное сокращение времени проектирования за счет отсутствия временных затрат на разработку какой-либо дополнительной документации. Недостатками минимально формализованного подхода являются риски, связанные с возможностью возникновения необходимости передачи проекта другой проектной команде, сложности в поддержке и переработке разработанной системы, высокая вероятность возникновения несогласованностей и прочих критических ошибок при проектировании. Напротив, преимуществами высоко формализованного подхода являются согласованность принятия решений по разработке, минимизация ошибок проектирования, простота в обслуживании объекта разработки. Недостатком высоко формализованного подхода, соответственно, являются огромные затраты сил и времени на разработку документации [10—14].

Таблица 2

Сравнение текстовых форматов хранения данных XML, RTF и CSV

Критерий	XML	RTF	CSV
Не зависит от платформы	Да	Да	Да
Имеет реализации парсеров для всех современных языков программирования	Да	Нет	Нет
Подходит для описания любых типов данных	Да, но сетевые модели описываются с трудом	Нет, только текст	Нет, только таблицы
Открыт, т. е. не защищен патентами и может внедряться без всяких финансовых отчислений третьим лицам	Да	Нет	Да
Самодокументируемый формат	Да, описывает структуру и имена полей, так же как и значения полей	Нет	Нет
Избыточный синтаксис	Да, одну структуру можно описать несколькими равноправными способами	Нет	Нет
Расширяемость	Да, позволяет определять и использовать собственные словари	Нет	Нет

Жесткость регламента. Каждый метод подразумевает набор правил, некоторый план действий для достижения конечного результата. В зависимости от подхода к разработке эти правила могут быть либо жестко регламентированы, либо, наоборот, предоставлять команде разработчика широкое поле деятельности.

Высоко регламентированные методы предъявляют жесткие правила к процессу проектирования, устанавливают достаточно четко определенную последовательность действий, не позволяют импровизировать, отступать от плана.

Низко регламентированные методы характеризуются отсутствием либо установленных правил, либо последовательности действий. При использовании таких методов процесс проектирования планируется лишь на очень абстрактном уровне.

Преимуществом низко регламентированных методов является отсутствие ограничений на методики и средства достижения конечного результата, что позволяет находить принципиально новые, наилучшие подходы к решению конкретных задач, их недостатком — то, что их использование не позволяет проводить сколько-нибудь серьезные проекты ввиду невозможности определения заранее сроков и масштабов работы.

Преимущество высоко регламентированных методов состоит в том, что они используют проверенные рабочие процессы проектирования, позволяют хорошо контролировать проектную группу, недостатки — в ограниченности поля деятельности.

Методы разработки визуальных схем сложных систем предписывают правила применения визуальных языков. Методы реализованы в программных инструментах, позволяющих удобно работать с визуальными языками. В состав подобных инструментов входят графические и текстовые ре-

Рис. 4. Карта противоречий при визуальном проектировании [8]:

а — логическое предметное противоречие задается через *P* (значение оцениваемого параметра) и не-*P* (противоположное значение оцениваемого параметра); *б* — операционное противоречие: *ПЭ* — положительный эффект, положительное влияние; *НЭ* — нежелательный эффект, нежелательное влияние; *В*М — визуальные методы проектирования

дакторы, а также средства валидации моделей, генераторы исходного кода по диаграммам и т. д.

Проблемы визуального моделирования позволяют построить карту противоречий, показанную на рис. 4. Для разрешения данных противоречий предлагается комплекс методов с единым унифицированным XML метаописанием, применяемым на каждом этапе создания визуальных схем и описывающим связи между ними. Такой подход позволит соединять базовые модели визуального проектирования в единую замкнутую цепь, сопутствующую как анализу процессов, так и построению

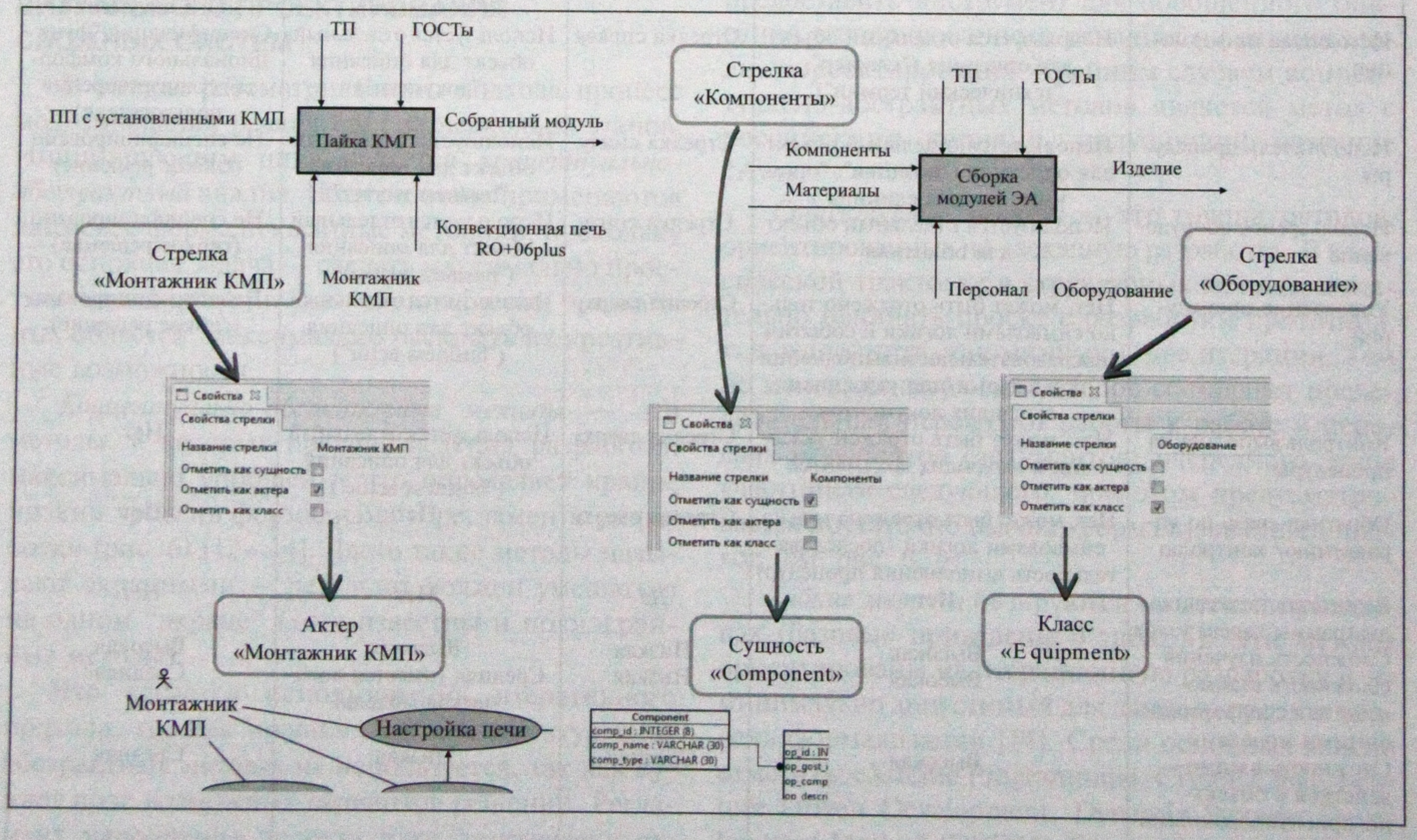

Рис. 5. Конвертор элементов при экспорте/импорте моделей

информационных и управляющих модулей. Существенным для реализации данного подхода является разработка программного средства, реализующего все выбранные методики на базе единой платформы. Это в свою очередь позволит не только использовать результаты моделирования одних этапов при построении моделей другого уровня, но и расширит жизненный цикл модели таким образом, что, проходя каждый новый виток эволюции, будет развиваться созданная в предыдущей итерации модель, и таким образом будет обеспечено соответствие модели в любой момент времени описываемой системе.

1. Необходимо грамотно, надежно и без нареканий со стороны заказчика сопроводить проект на всех циклах проектирования.

2. Для этого нужно подробно прописать регламенты всех этапов работы, т. е создать формальное описание процедур (большое и подробное на все случаи жизни).

3. Но из-за большого объема документации необходимо прорабатывать каждое изменение проекта и фиксировать его в соответствии с разработанным формальным описанием.

4. На все это нужно время и это приведет к увеличению сроков проектирования.

Таблица 3

Сравнение методов ARIS, IDEF0, RUP(UML), MSF

Критерии сравнения	ARIS (ARIS eEPC)	IDEF0	RUP(UML)	MSF
Принцип построения диаграммы / логика процесса	Временная последовательность выполнения процедур	Принцип доминирования	Временная последовательность выполнения процедур	Управление компромиссами (ресурсы-время-возможности)
Описание процедуры / процесса	Объект на диаграмме	Объект на диаграмме	Объект на диаграмме	Компонент шаблона проектирования
Входящий документ	Используется отдельный объект для описания ("документ")	Стрелка слева, стрелка сверху	Используется отдельный объект для описания ("документ")	Используется отдельный объект для описания ("документ")
Входящая информация	Используется отдельный объект для описания ("кластер", "технический термин")	Стрелка слева, стрелка сверху	Используется отдельный объект для описания ("документ")	Матрица компромиссов
Исходящий документ	Используется отдельный объект для описания ("документ")	Стрелка справа	Используется отдельный объект для описания ("документ")	Используется отдельный объект для описания ("документ")
Исходящая информация	Используется отдельный объект для описания ("кластер", "технический термин")	Стрелка справа	Используется отдельный объект для описания ("документ")	Спецификация "функционального комфорта" (удовлетворения пользователя)
Исполнитель процедуры	Используется отдельный объект для описания ("позиция", "организационная единица")	Стрелка снизу	Используется отдельный объект для описания ("business worker")	Не специфицировано (гибкое решение)
Используемое оборудование	Используется отдельный объект для описания	Стрелка снизу	Используется отдельный объект для описания ("business worker")	Не специфицировано (гибкое решение)
Управление процедурой	Нет, может быть отражено только символами логики и событий (последовательность выполнения процедур) и/или указанием входящих документов	Стрелка сверху	Используется отдельный объект для описания ("business actor")	Не специфицировано (гибкое решение)
Контроль выполнения процедуры	Нет, может быть отражен указанием входящих документов	Стрелка сверху	Используется отдельный объект для описания ("business actor")	Нет
Обратная связь по управлению/ контролю	Нет, может быть отражена только символами логики (последовательность выполнения процедур)	Стрелка сверху	Нет	Нет
Возможность создания диаграммы дерева узлов	Нет	Да	Нет	Нет
Сложность изучения	Высокая	Низкая	Высокая	Высокая
Сложность взаимодействия с информационными моделями	Высокая	Низкая	Средняя (имеется логическая модель)	Средняя
Сложность взаимодействия с объектно-ориентированными моделями	Высокая	Средняя	Низкая	Средняя

Необходимо найти такое решение, чтобы при малых сроках проектирования обеспечивалась бы высокая эффективность проекта. Следовательно, современные средства создания визуальных схем должны представлять собой сочетание графической и текстовой нотаций, связанных с компонентами визуального языка, в которых представлено описание атрибутов данных компонентов. Сочетание графической и текстовой нотации должно обеспечивать синтез полнотекстовых документов по разработанным моделям и миграцию компонентов между моделями различного уровня экспертизы (рис. 5).

В качестве критериев сравнения была выбрана возможность реализации наиболее важных инструментов, сложность изучения методов, а также возможность организации взаимодействия между моделями. Сравнение приведено в табл. 3 [12, 13].

Как видно из таблицы, наиболее подходящей для моделирования технологических процессов является метод IDEF, при этом гарантируется возможность описания как предметных, так и операционных задач. При создании встраиваемых систем выбор метода следует осуществлять с учетом используемой программной платформы (Windows, Unix, MacOs и др.).

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ ВИЗУАЛЬНОГО ПРОЕКТИРОВАНИЯ СЛОЖНЫХ СИСТЕМ

В рамках рассматриваемого подхода процесс моделирования реализуется за несколько этапов. Инициирующим цикл является *концептуально-абстрактный* анализ. На этом этапе применяются наименее формализованные методики, поскольку его основная задача — сделать максимально простым моделирование для специалистов предметных областей, максимально раскрыть их креативные возможности.

Концептуально-абстрактные методы — это методы, в которых правила ведения разработки максимально упрощены, что определяет крайне низкий уровень формализма и регламента разработки (рис. 6) [12—14]. Часто такие методы называют экранными — результат должен умещаться на одном “экране” (хотя известны и полиэкранные методы).

Что касается использования итеративного подхода, то, как правило, он в концептуально-абстрактных методах не используется, так как сужает поле возможных вариантов решений. Регламент выполнения проекта тоже практически отсутствует.

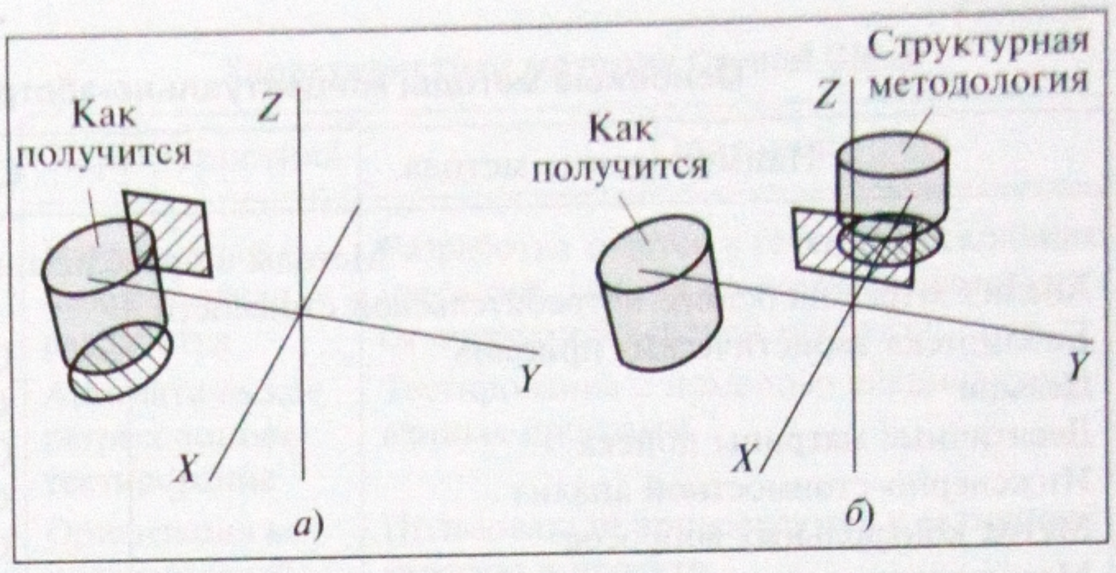

Рис. 6. Место в трехмерном пространстве

а — лабиринтовые методы; *б* — структурные методы

В настоящее время известно более сотни методов концептуально-абстрактного анализа. Каждый из них имеет свою область применения, уровень сложности и особенности, некоторые из них представлены в табл. 4 [16]. Применение в начале проектирования концептуально-абстрактного моделирования позволяет более четко поставить проблему (задачу) разработки и с наименьшими затратами перейти к более глубокому анализу.

Концептуально-абстрактные методы занимают крайнее левое нижнее положение в пространстве свойств (см. рис. 6). На них не налагается жестких ограничений по формализации, регламенту, итерационности процесса, основная их задача предоставить инструмент для обобщенного описания сгенерированного решения на ранних стадиях проектирования. Крайним случаем концептуально-абстрактных методов является метод с российскими корнями, получивший название “Avos”.

Структурные методы — это группа методов, ориентированных на каскадную разработку. В классической трактовке в структурных методах желательно начинать проект с разработки прототипа, т. е. выполнить, как минимум, две итерации. Тем не менее, основу этих методов составляет последовательный переход от работы к работе и передача результатов (документов) очередного этапа участникам следующего, при этом предусматривается достаточно высоко формализованный подход (см. рис. 6, *б*).

Гибкие методы базируются на десяти принципах (базовые приведены в табл. 5). Эти методы ориентированы на итерационную разработку и на минимально допустимый для данного проекта уровень формализации [14]. Среди основных можно отметить: eXtreme Programming, Crystal Clear, Feature Driven Development, Dynamic Systems Development Method (DSDM), Scrum, Adaptive software development, Pragmatic Programming и др.

Таблица 4

Основные методы концептуально-абстрактного синтеза технических решений [16]

Наименование метода	Страна	Год публикации	Автор
Методы с преобладанием "текстовой нотации"			
Анализ затрат на основе потребительской стоимости	ГДР	1971	К. Томас
Библиотека эвристических приемов	СССР	1969	А. Половинкин
Дельфи	США	1966	О. Хелмер
Десятичные матрицы поиска	СССР	1972	Р. Повилейко
Инженерно-стоимостной анализ	США	1961	Л. Майлз
Метод контрольных вопросов	США	1945	Дж. Пойа
Метод организующих понятий	ГДР	1953	Ф. Ханзен
Метод "Матриц открытий"	США	1954	А. Моль
Метод "Мозгового штурма"	США	1957	А. Осборн
Метод фокальных объектов	США	1958	Ч. Вайтинг
Метод направленного решения	СССР	1961	Н. Середа
Метод рационального конструирования	США	1966	Р. Мак-Крори
Метод конференции идей	ГДР	1970	В. Гильде, К. Штарке
Морфологический анализ	США	1942	Ф. Цвикки
Психоэвристическое программирование	СССР	1968	В. Чавчанидзе
Семикратный поиск	США	1964	Г. Буш
Синектика	США	1944	В. Гордон
Стандарты на решения изобретательских задач	СССР	1979	Г. Альтшуллер
Методы с преобладанием "визуальной нотации"			
Алгоритм изобретения	СССР	1956	Г. Альтшуллер
Вепольный (структурный) анализ	СССР	1973	Г. Альтшуллер
Метод каталога	Германия	1926	Ф. Кунце
Метод поэлементного анализа	СССР	1950	Ю. Соболев
Метод комплексного решения проблем	ЧСФР	1967	С. Вит
Метод гирлянд случайностей и ассоциаций	СССР	1972	Г. Буш
Метод выявления обобщенных приемов на основе анализа описаний изобретений	СССР	1978	М. Зарипов
Многоэкранный системный оператор	СССР	1979	Г. Альтшуллер
Моделирование "маленькими человечками"	СССР	1974	Г. Альтшуллер
Функциональный метод проектирования	Великобритания	1966	Е. Мэтчетт
Функционально-идеальное моделирование (свертывание) систем	СССР	1988	—
Совокупность визуальных методов поддержки интеллектуальной деятельности	—	—	Не имеют связанных методик синтеза решений, обладают только средствами визуализации

С 2001 г. распространение получил подход “Agile” (Agile software development) гибкой разработки встраиваемых программных систем, ориентированный на использование личностного взаимодействия и итеративной разработки в условиях динамики формирования и реализации требований проектирования в результате постоянного внутреннего личностного взаимодействия проектных групп. Данный подход практически минимизирует уровень формализма и нацелен на быстрейшее получение готовых реализаций, базируется на четырех базовых положениях и 12 принципах, изложенных в манифесте (Agile Manifesto, 2001).

Экстремальное программирование (eXtreme Programming) или метод XP, разработанный Kent Beck, Ward Cunningham и Ron Jeffries, является наиболее известным из гибких методов. Иногда само понятие гибкие методы явно или неявно

Таблица 5

Базовые принципы гибких методов

Принцип	Описание
Ориентация на заказчика	Главное — удовлетворить заказчика и предоставить ему продукт как можно скорее
Частые релизы	Новые выпуски продукта должны появляться раз в несколько недель, в крайнем случае, месяцев
Быстрая передача проектной информации	Наиболее эффективный способ передачи знаний участникам разработки и между ними — личное общение
Ориентация на межличностные коммуникации	Какие либо документы должны появляться только тогда, когда иными коммуникационными способами информация не передается
"Команда профессионалов"	Высокий уровень доверия, квалификации, самодисциплины и ответственности среди членов рабочей группы
Ориентация на результат	Работающая программа — лучший показатель прогресса разработки

отождествляют с XP. XP проповедует коммуникабельность, простоту, обратную связь и настойчивость. Классически XP описывается как 11 практик, представленных в табл. 6 [12—14].

При использовании XP предварительное проектирование заменяется постоянным присутствием в команде "носителя знаний о предметной области" (готового ответить на любой вопрос и принять решение) и сопровождается рефакторингом — регулярными переработками кода. Основой проектной документации считается тщательно прокомментированный код. Очень большое внимание в методологии уделяется тестированию, можно даже отметить, что создание тестов подменяет собой формирование и согласование требований к системе.

Crystal Clear — семейство методик, определяющих необходимую степень формализации процесса разработки в зависимости от количества участников и критичности задач [12—14]. Методики уступают XP по производительности, зато максимально просты в использовании. Требуют минимальных усилий для внедрения, так как ориентированы на человеческие привычки. Считается, что методика описывает тот естественный порядок разработки, который устанавливается в достаточно квалифицированных коллективах. Основные характеристики методологии приведены в табл. 7.

Таблица 6

Практики экстремального программирования

Практика	Описание
Игра в планирование	В экстремальном программировании все начинается с планирования, оно должно быть простым, быстрым и последовательным
Короткие релизы	Быстрый запуск в производство простейшей системы и затем выпуск новых версий в течение очень короткого времени
Метафоры	Простая и понятная концепция, единство интерпретации
Простой дизайн	Максимальная простота и максимальная интуитивность функциональных элементов, стремление к выполнению правила "трех кликов"
Переработки кода (refactoring)	Широкое использование библиотечных элементов и наработок
Разработка "тестами вперед"	При разработке нового модуля сначала разрабатывается тест для проверки его работоспособности (проектирование от задачи)
Парное программирование	Весь код пишется парами
Коллективное владение кодом	Любой разработчик может изменять любой код для расширения функциональности, исправления ошибок и улучшения
40-часовая рабочая неделя	Разработчик не должен работать больше 8 часов в день, возможность гибкого графика и удаленной работы
Постоянное присутствие заказчика	При разработке всегда необходимо присутствие представителя заказчика для быстрого разрешения спорных моментов
Стандарты кода	Использование единого подхода (жесткого регламента) для кодогенерации, строгое выполнение требований соглашения о разработке или иного документа, определяющего стиль

Таблица 7

Характеристики методик Crystal Clear

Характеристика	Описание
Итеративная инкрементная разработка	Разработка ведется в течение нескольких итераций, каждая из которых завершается очередной версией продукта
Автоматическое регрессионное тестирование	Тестирование с помощью специализированных программ
Ориентация на пользователей	Пользователи привлекаются к активному участию в проекте
Минимальная формализация	Состав документации определяется участниками проекта
Поддержка версионности	Как правило, используются средства контроля версий

Помимо Crystal Clear в семейство Crystal входит еще несколько методов, предназначенных для выполнения более крупных или более критических проектов. Они отличаются несколько более жесткими требованиями к объему документации и вспомогательным процедурам (управление изменениями, версиями и т. п.).

Функционально-ориентированная разработка (FDD, Feature Driven Development) — итеративная методология разработки, ориентированная на функциональность (свойства) разрабатываемой системы. В ее основе лежит понятие функции или свойства (feature) системы (близко к понятию прецедента в RUP). При этом налагается достаточно жесткое ограничение на понятие "функция": каждая функция должна допускать реализацию не более, чем за две недели, т. е. если сценарий использования достаточно мал, его можно считать функцией. Если же велик, то его надо разбить на несколько относительно независимых функций. FDD включает пять процессов, перечисленных в табл. 8. Последние два повторяются для каждой функции.

Разработчики в FDD делятся на "хозяев классов" и "главных разработчиков". Главные разработчики привлекают хозяев задействованных классов к работе над очередным свойством. Для сравнения, в XP нет персонально ответственных за классы или методы. К работе над любым классом

может привлекаться любой из участников разработки. Работа над проектом предполагает частые сборки и делится на итерации, каждая из которых предполагает реализацию определенного набора функций [12—14].

ГОСТы — группы методов, регламентированные государственными стандартами. В настоящее время в России действуют старые ГОСТы 19-ой и 34-ой серий и более новый ГОСТ Р ИСО МЭК 12207. ГОСТы 19-ой и 34-ой серий жестко ориентированы на каскадный подход к разработке (рис. 7). Разработка проводится по этапам, каждый этап предполагает выполнение строго определенных работ. Процесс завершается выпуском достаточно большого числа весьма формализованных и обширных документов. Строгое следование этим стандартам сразу приводит к каскадному подходу и к очень высокой степени формализованности разработки [12—14].

ГОСТ 12207, в отличие от стандартов 19-ой и 34-ой серий, описывает процесс разработки в виде совокупности основных и вспомогательных процессов, которые могут действовать с начала до завершения проекта (основное внимание уделяется процессному подходу). Требования стандарта не запрещают применение итеративного подхода, но и не рекомендует его использование. Также мягче ГОСТ 12207 и в части требований к формальности процесса разработки. В нем содержатся только указания на необходимость документирования основных результатов всех процессов, но нет перечней необходимых документов и указаний относительно их содержания. Таким образом, ГОСТ 12207 допускает итеративную и менее формализованную разработку [14].

Модели зрелости процесса разработки (*CMM, CMMI — Capability Maturity Model*) — модель зрелости процессов проектирования, которая предназначена для оценки уровня зрелости процесса разработки в конкретной компании [12]. В соответствии с этой моделью есть пять уровней зрелости процесса разработки. Первый уровень соответствует разработке “как получится”. Второй уровень соответствует более-менее налаженным процессам, когда можно с достаточной уверенностью надеяться на положительный исход проекта. Третий уровень соответствует наличию разработанных и хорошо описанных процессов (библиотек), используемых при разработке. Четвертый — активному использованию метрик в процессе управления для постановки целей и контроля их достижения. И, наконец, пятый уровень означает способность компании оптимизировать процесс по выбранным критериям (интегральные: стоимость, время; дифференциальные: пользовательские критерии) по мере необходимости.

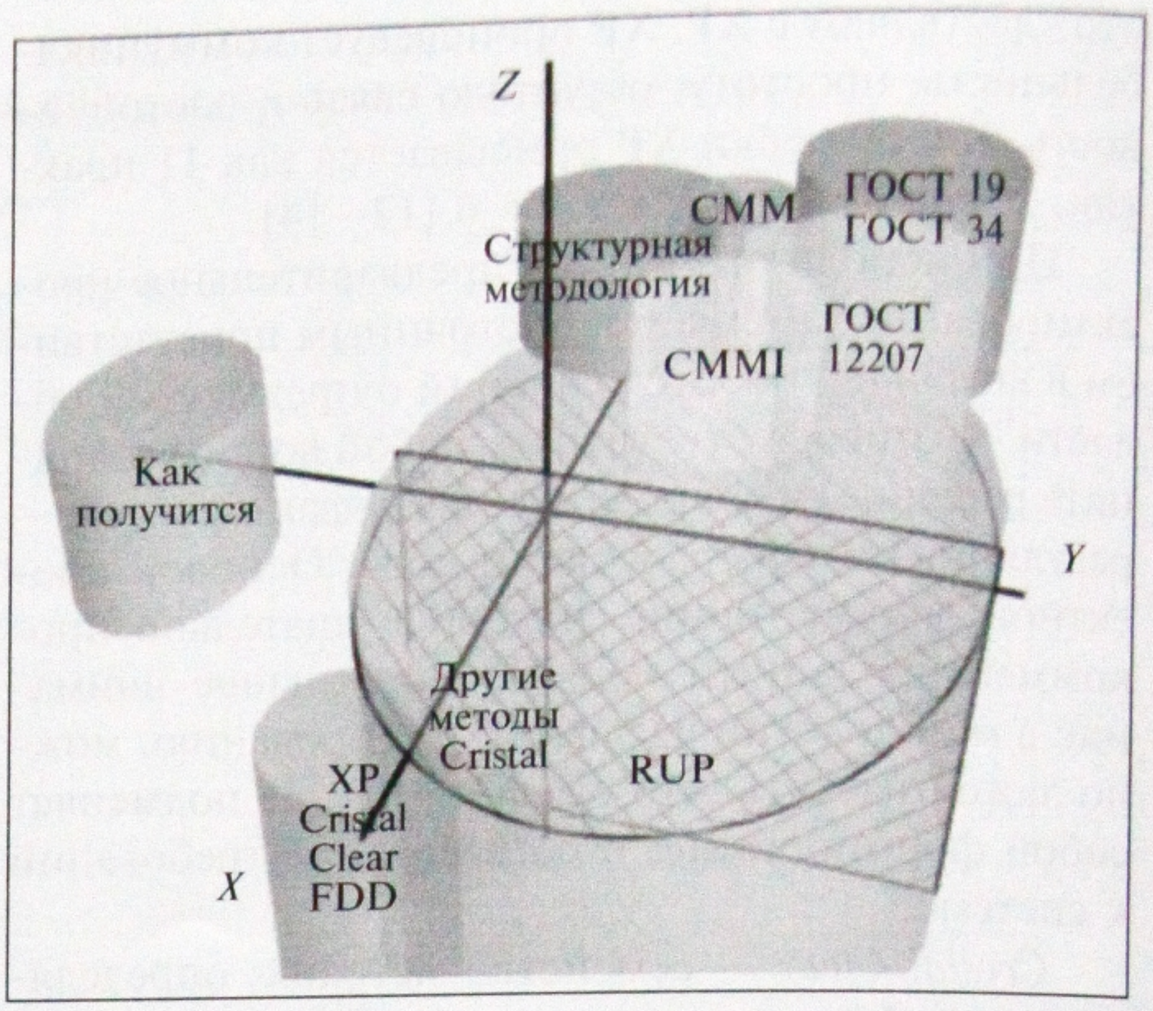

Рис. 7. Сводное распределение методов в трехмерном пространстве свойств

После появления CMM стали разрабатываться специализированные модели зрелости для разработки информационных систем, для процесса выбора поставщиков и некоторые другие. На их основе была разработана интегрированная модель CMMI (Capability Maturity Model Integration). Кроме того, в CMMI была предпринята попытка преодолеть недостатки CMM в части завышения роли формальных описаний процессов, когда наличие определенной документации оценивалось значительно выше, чем просто хорошо налаженный, но не описанный процесс.

Таким образом, основой CMM и CMMI является формализация процесса разработки. Они нацеливают разработчиков, желающих сертифицироваться на достаточно высокую степень зрелости процесса разработки, на внедрение жесткого фор-

Таблица 8

Процессы FDD

Процесс	Описание
Разработка общей модели	Структурирование объекта разработки
Определение функциональности	Составление списка необходимых функций системы
Составление плана	Планирование работы над каждой функцией
Проектирование функции	Разработка необходимой функции
Конструирование функции	Реализация необходимой функции

мализованного процесса. На максимальных уровнях зрелости предполагается, что этот процесс настраивается для каждого конкретного проекта. Однако, в рамках проекта процесс остается очень жестким.

Связь CMM и CMMI с итеративной разработкой более опосредованная. Формально и та, и другая не выдвигают конкретных требований к тому, чтобы придерживаться каскадного или итеративного подхода. Однако, по мнению ряда специалистов, CMM в большей степени совместим с каскадным подходом, в то время как CMMI допускает также и применение итеративного подхода [14].

Рациональный Унифицированный Процесс (RUP) — можно отнести к гибридным методам, в которых сочетается итерационный подход и возможность применения в "каскадных" проектах (см. рис. 7).

Итерационная разработка программного обеспечения в RUP [5—7, 12—14] предполагает разбиение проекта на конечное количество итераций, которые выполняются последовательно, и каждая итерация разработки четко определена набором целей, которые должны быть достигнуты в конце итерации, и критериями оценки достигнутого результата. В результате выполнения каждой итерации появляется новая версия разрабатываемой системы. Конечная итерация предполагает, что набор целей итерации должен в точности совпадать с набором целей, указанных заказчиком продукта, т. е. все требования должны быть выполнены. Следовательно, RUP можно позиционировать, как итеративную, с гибким диапазоном формализации процесса разработки в зависимости от уровня экспертизы. В части регламента RUP не задает жесткие рамки на длительность итерации. Применять RUP можно и в том случае, когда заказчик требует строго соответствия ГОСТам 19 и 34, максимально расширив стадии предпроектных исследований, согласования требований и сдачи продукции.

Что касается уровня формализма, то RUP представляет пользователю весьма широкий диапазон возможностей [14]. Если выполнять все работы и задачи, создавать все артефакты и достаточно формально проводить все рецензирования, RUP представляет собой крайне формальную, тяжеловесную методологию. С другой стороны, RUP позволяет разрабатывать только те артефакты и выполнять только те работы и задачи, которые необходимы в конкретном проекте. Основными являются пять базовых моделей: модель вариантов использования, деятельности, логическая, модели компонентов и развертывания. Каждая из моделей включает набор соответствующих диаграмм, представляемых средствами языка UML [5—7, 12—14]. При разработке вводится понятие "уровня экспертизы": если принятое решение по возникшему вопросу касается конкретного компонента разработки, то оно принимается на уровне владельца (создателя) компонента и описывается со степенью детализации, установленной им. Несмотря на то что в последних версиях UML появились средства по расширению и конфигурированию языка (например, profile-механизм), одним из существенных его недостатков остается его "тяжеловесность" и средства по его настройке и динамической настройки под локальные задачи пока еще не достаточно сильно развиты.

MSF (Microsoft Solution Framework) — подход компании Microsoft по итерационной разработке, базирующийся на наборе методик организации процесса разработки, который может быть адаптирован под потребности коллектива разработчиков [17]. MSF содержит не только рекомендации общего характера, но и предлагает адаптируемую модель коллектива разработчиков, определяющую взаимоотношения внутри коллектива, гибкую модель проектного планирования, основанного на управлении проектными группами, а также набор методик для оценки рисков.

MSF базируется на модели проектной группы и модели процессов. В MSF нет роли "менеджер проекта" и иерархии руководства, управление разработкой распределено между руководителями отдельных проектных групп с выделением трех базовых фаз: управление проектами, управление рисками и управление подготовкой. Жизненный цикл процессов в MSF сочетает водопадную и спиральную модели разработки: проект реализуется поэтапно, с наличием соответствующих контрольных точек, а сама последовательность этапов может повторяться по спирали. Треугольник приоритетов является основой для матрицы компромиссов (ресурсы—время—возможности), заранее утвержденных представлений о том, какие аспекты процесса разработки будут четко заданы, а какие будут согласовываться или приниматься как есть. Впервые среда разработки, в полной мере поддерживающей основные идеи MSF, реализована в Microsoft Visual Studio 2005 Team Edition. Это не только среда разработки для индивидуальных членов коллектива, а комплексное средство поддержки коллективной работы. Начиная с MSF 4.0 произошло разделение на два направления: MSF for Agile Software Development (увеличение гибкости → Agile манифесту) и MSF for

CMMI Process Improvement (увеличение формализма → RUP).

В качестве рекомендаций можно отметить, что MSF ориентирована на проектные группы малых и средних размеров, она не налагает никаких ограничений на используемый инструментарий и содержит рекомендации весьма общего характера, варьируя которыми можно сформировать конкретный процесс, соответствующий потребностям конкретного коллектива разработчиков.

DSM-подход (Domain-Specific Modeling — предметно-ориентированное моделированию) предполагает использование готовых IP модулей (Intellectual Property Cores) или библиотечных средств для разработки графических редакторов с возможностью задать собственную графическую нотацию с репозиторием, навигатором по модели и т. д. (табл. 9). В качестве примеров реализации можно отметить: Eclipse/GMF, Microsoft DSL Tools, Microsoft Visio 2005 и др.

Предметно-ориентированное моделирование направлено на улучшение технологии проектирования встраиваемых систем, обеспечивая рост производительности проектирования при увеличении сложности решений за счет применения крупных библиотечных вычислительных модулей. Эти модули должны быть надежно повторяемыми и настраиваемыми под решаемые задачи в ряде проектов. Повторное применение таких модулей (IP Core reuse), которые можно назвать вычислительными заготовками за их функциональную и технологическую адаптируемость, позволяет уменьшить трудозатраты и сроки проектирования. Вычислительные заготовки (IP модули) различаются по степени гибкости своей настройки под условия потребителя как гибкие, жесткие (логические), твердые (инсталляционные комплекты).

Гибкие заготовки обычно подстраиваются к условиям нового проекта в широких пределах и независимы от его технологии. Чтобы компонент был принят как гибкий IP модуль, он должен иметь:

— исчерпывающую и ясную документацию;

— текст описания на исходном языке в хорошем стиле для синтеза, заготовка должна быть настраиваемой под технические условия потребителя;

— хорошие средства верификации в виде исчерпывающих тестов;

— компактность (безизбыточность);

— отчуждаемость (лицензионная, логическая, физическая);

— четкую методику того, как компонент инкапсулируется в проект, включающую надежные скрипты, автоматизирующие тестирование.

Можно выделить четыре типа инструментальных средств разработки DSM-решений [5—7, 12—17]:

— полноценные среды разработки DSM-пакетов (например, Eclipse/GMF, Microsoft DSL Tools и др.);

— конфигурируемые и надстраиваемые графические пакеты (например, Microsoft Visio, MetaEdit+, AutoCAD и др.);

— различные библиотеки для создания отдельных компонент DSM-пакетов — графические библиотеки, библиотеки для работы с данными (например, Eclipse/EMF и др.);

— CASE-средства, имеющие развитые программные интерфейсы, встроенные скриптовые языки, модульную архитектуру и т. д. (например, Borland Tother Control Center, IBM Rational Rose и др.).

Основная идея определения DSM-платформы — подчеркнуть, что средства визуального моделирования сегодня не создаются “с нуля”, а используют существующие на рынке библиотеки, пакеты и технологии (технология “IP блоков”). DSM-платформу для конкретного проекта составляет то, что используется для создания биб-

Таблица 9

Базовые компоненты DSM-подхода

Наименование		Определение
Предметная область	Problem domain, Domain	Часть реального мира (люди, знания, процессы и др.), объединенная в одно целое для решения конкретных задач
Предметно-ориентированный язык	Domain Specific Language — DSL	Это язык (визуальный), который создается для использования в рамках некоторой предметной области
DSM-решение	DSM solution	Предложенный для предметной области DSL, метод его применения, средства инструментальной поддержки и внедрения
DSM-пакет	DSM package	Часть DSM-решения, соответствующая программным средствам (IP Cores)
DSM-продукт	DSM product	
DSM-инструменты	DSM tools	
DSM-платформа	DSM platform	Инструментальные средства разработки DSM-решений

лиотечных визуальных элементов. DSM-платформы включают общие "тяжелые" компоненты, такие, как графические средства, репозиторий, среду. Локальный DSM-пакет имеет четко определенную функциональность в рамках конкретного проекта. Он может быть графическим редактором для диаграмм какого-либо одного вида, а может обеспечивать кодогенерацию по моделям, отладку спецификаций в терминах модели, реализовывать проектные процедуры [5—7, 12—17].

Применение DSM-подхода позволяет синтезировать локальные визуальные методы проектирования, ориентированные на учет уникальных бизнес процессов, обеспечивая принципы минимальной достаточности (без избыточного формализма), "функционального комфорта" (ориентации на требования заказчика) и гибкого регламента.

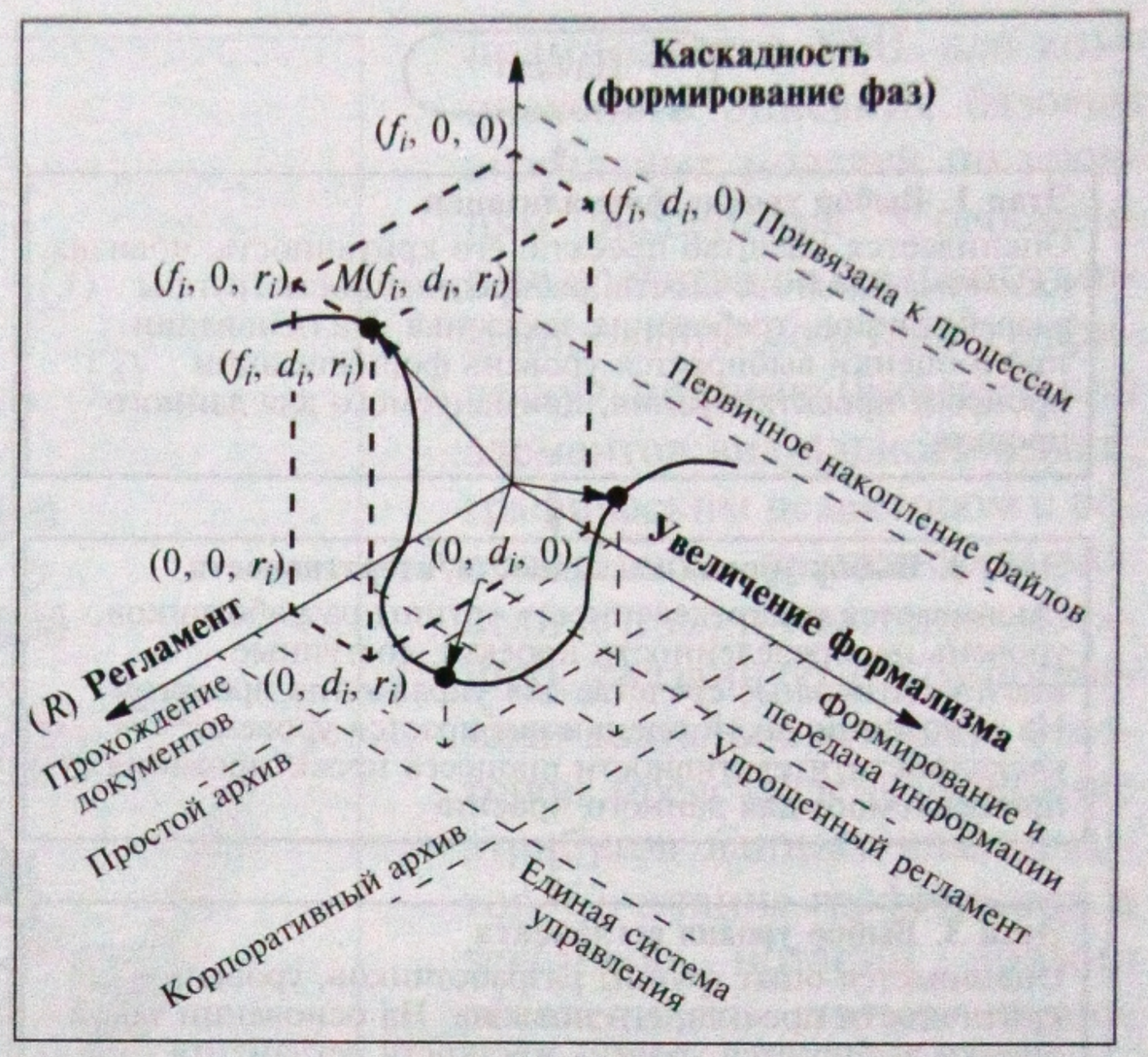

Рис. 8. Эволюция методов проектирования в трехмерном пространстве свойств

ЭВОЛЮЦИЯ МЕТОДОВ ВИЗУАЛЬНОГО МОДЕЛИРОВАНИЯ В ТРЕХМЕРНОМ ПРОСТРАНСТВЕ СВОЙСТВ

Проводя оценку применимости и развития визуальных методов проектирования по трехмерному пространству свойств (см. рис. 2) можно четко представить, в каких направлениях нужно двигаться дальше, чего недостает в текущий момент и каким образом органично использовать уже существующие компоненты. В общем случае, процесс визуального проектирования можно представить в виде кривой в трехмерном пространстве свойств (рис. 8). Причем, чем круче эта кривая, тем быстрее идет процесс эволюции, а чем больше значения всех трех координат, тем выше уровень каскадности реализуемых методов проектирования. Предложенная модель не является застывшим образованием. Она претерпевает три основные фазы своей эволюции (каскадность, формализм, регламент).

Если обратиться к руководствам по системному анализу и проектированию, то можно обнаружить насколько важным считается возможность заменить толстые тома компактными диаграммами [5—8, 12—17]. Действительно, документация с большим количеством диаграмм становится доступней и компактней. Но и наличие самой подробной документации не всегда спасает, не существует единого наилучшего уровня формализации процесса, он должен выбираться каждый раз для каждого конкретного проекта. Разработка идет значительно быстрее, а ошибок совершается значительно меньше, если помимо документации есть эксперт, который знает, что в ней написано, и может объяснить это всем заинтересованным лицам. Точно так же, как постоянный контакт с заказчиком и возможность быстро получить ответ на любой вопрос ускоряют работу, также возможность обратиться непосредственно к эксперту предметной области и уточнить детали рассматриваемого решения ускоряет работу. Кроме того, такие контакты позволяют участникам разработки получить объективную оценку качества предложенных ими решений и накопить опыт. К сожалению, оси качества и надежности в пространстве свойств в данном контексте не рассматриваются явно, хотя их оценка также крайне важна при выборе метода проектирования.

На основании приведенного анализа можно описать обобщенный алгоритм выбора метода для конкретного проекта, основанный на оценке параметров проекта и получении уровней ключевых характеристик (компетенций) (рис. 9).

На начальных этапах данного алгоритма происходит оценка параметров проекта с целью определения уровней ключевых характеристик визуального проектирования. На основе полученных уровней определяется положение реализуемого процесса в трехмерном пространстве свойств, что позволяет выбрать наилучшую для него методологию.

Исходя из расположения "эффективной площади методики" в трехмерном пространстве свойств можно сделать выводы о применимости той или иной методики на различных стадиях моделирования (рис. 10), при этом необходимо обеспечение базового требования "однократнос-

Рис. 9. Алгоритм выбора наилучшего метода с учетом решаемой задачи

ти ввода данных", что обуславливает необходимость применения единого метаописания для набора эффективных визуальных методик. Следовательно, используемые инструменты визуального моделирования должны быть способны, во-первых, предоставлять информацию о предметной области, в нашем случае о конструкторско-технологических решениях встраиваемых систем, под которыми понимается совокупность элементов и деталей с различными физическими свойствами и формами, находящимися в определенной пространственной, механической, тепловой, электромагнитной и энергетической взаимосвязи, и процессов их преобразования. Эта взаимосвязь определяется системотехнической, схемотехнической, конструкторской и технологической документацией и обеспечивает выполнение встраиваемыми системами заданных функций с необходимой точностью и надежностью в условиях воздействия на нее различных факторов: эксплуатационных, производственных, социальных.

Во-вторых, необходимо иметь возможность описания процессных моделей технологических процессов — определенных действий, направленных на изменение исходных свойств объекта производства (в нашем случае ЭА) и достижения им определенного состояния, соответствующего технической документации. Конструирование и технология производства являются, с одной стороны, отдельными частями сложного процесса разработки ЭА, а с другой, не могут выполняться в отдельности, без учета взаимосвязей между собой и с другими этапами разработки. Являясь этапами более общего процесса "разработка — производства — эксплуатация — утилизация" (жизненного цикла изделия), как конструирование, так и технология определяют в конечном итоге общие потребительские свойства ЭА.

При таком подходе используемые визуальные среды моделирования должны предоставлять синхронный инструментарий описания функционально законченных компонентов, материалов, элементной базы, синтеза технологических процессов их изготовления, исследования физических и физико-химических явлений в процессах их получения, проектирование и конструирование, методы разработки и применения диагностического и технологического оборудование в рамках комплексной сквозной информационной поддержки жизненного цикла.

Современный характер рассмотрения сложных систем определяет проблемы анализа протекающих процессов, трудоемкости формального описания, хранения и обработки знаний об их объектах и процессах. Поиск решения данных проблем в настоящий момент построен вокруг использования визуального моделирования и проектирования в качестве основного инструмента генерации, хранения и обработки знаний в конкретной предметной области. Как показано на рис. 10, выделяются концептуальный, структурно-функциональный, логический и физический уровни анализа предметной области. При этом на каждом из уровней имеются объекты одной природы, но представленные описания модели разного уровня не синхронизированы (проблема синхронизации). Синхронизировать параметрический и структурный составы взаимосвязанных объектов различных уровней — одна из основных задач развития комплексных систем визуального моделирования.

Проблема когнитивности обусловлена разнородностью используемых методов в современных визуальных моделях для получения и хранения знаний о предметной области.

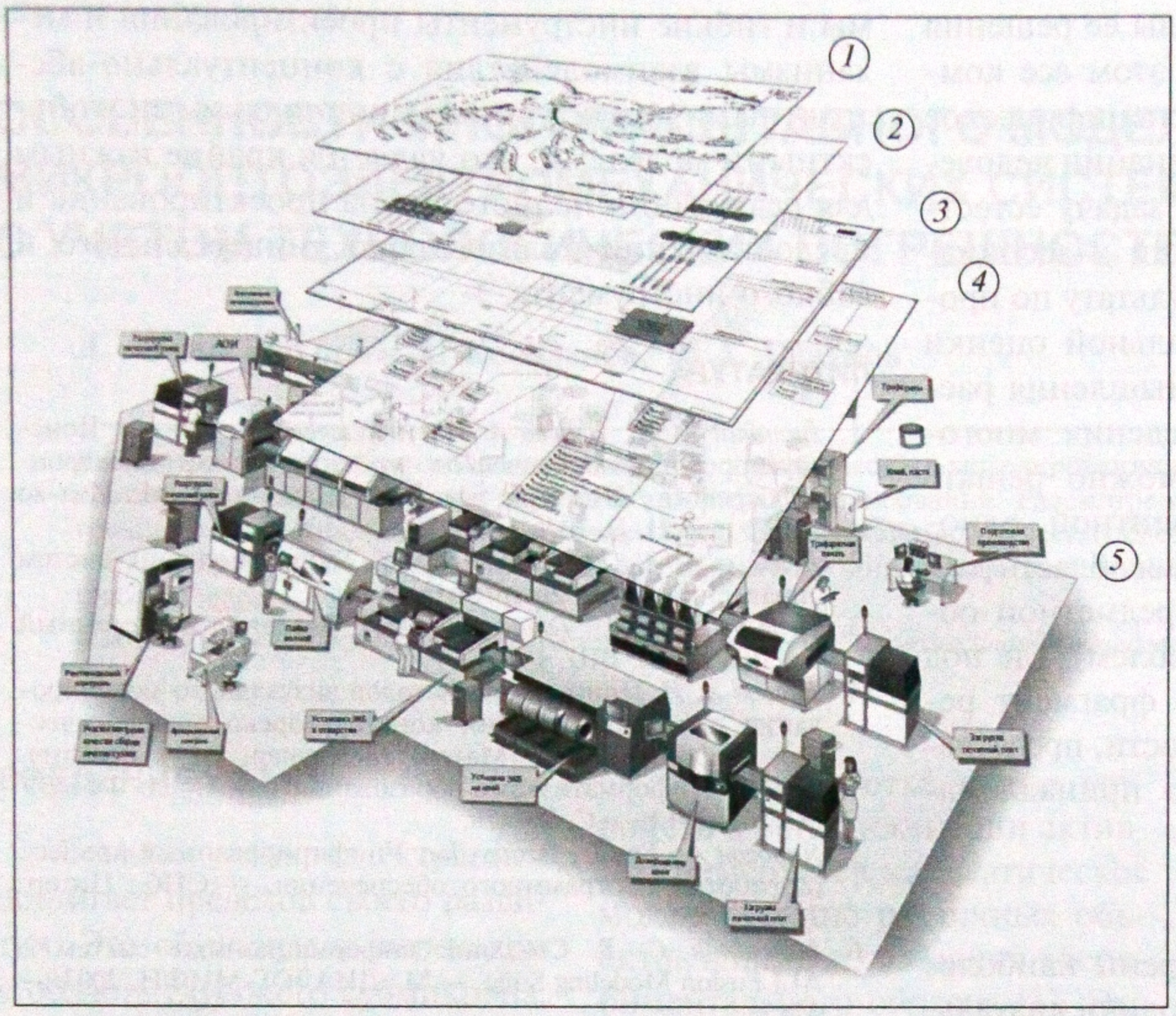

Рис. 10. Пирамида инкапсуляции уровней визуального моделирования:
1 — концептуальный уровень; *2* — логический уровень; *3* — структурно-функциональный уровень; *4* — объектный уровень; *5* — физический уровень

Проблема конвергенции (от английского convergence — схождение в одной точке) обусловлена сложностью взаимопроникновения технологий (концептуальный, структурно-функциональный, логический и физический уровни модели), когда границы между отдельными уровнями стираются, а многие интересные решения находятся именно на стыке уровней визуальных моделей.

Проблема инкапсуляции обусловлена на данном этапе развития визуальных методов фрагментарностью визуального анализа и изолированностью его от применения на разных этапах жизненного цикла.

Совершенно очевидно, найдя способы решения указанных выше проблем на основе визуальных методов, возможно предложить адекватную методику для формализованного комплексного представления и управления техническими, производственными, экономическими, социально-психологическими знаниями в сложных социально-производственных системах, обеспечив высокий уровень интерпретации знаний на каждом из уровней экспертизы.

Создание правил описания понятий и суждений позволит создать язык для описания конструкторско-технологических знаний. В [4, 9, 11] предложен язык I-XML (Intelligence XML) как подмножество XML для комплексного описания базовых визуальных моделей по уровням экспертизы. Современные средства визуального моделирования сочетают графическую нотацию (библиотекой элементов визуального языка, графическим редактором с репозиторием, навигатором модели и т. п.) и текстовую нотацию, связанную с компонентами визуального языка, в которой представлено описание атрибутов данных компонентов. Сочетание графической и текстовой нотации должно обеспечивать синтез полнотекстовых документов по разработанным моделям.

Универсальная среда моделирования должна обеспечивать в общем случае инструменты по инкапсуляции понятий предметной области в моделях различного уровня экспертизы, реализацию общих сервисных функций по динамическому формированию меню и методо-ориентированных панелей инструментов с функциями создание/удаление/открытие/закрытие компонентов, диаграмм и всего проекта в целом, а также набора сервисных функций (печати диаграмм, спецификаций, различных форм отчетов, настройки цветов, шрифтов, толщины линий, геометрии и т. п.). Среда обычно включает в себя несколько рабочих областей — “рабочее поле диаграммы”, навигатор модели, сервисные окна, библиотеку и панель инструментов с элементами графической нотации. Среды могут существенно различаться по предоставляемому набору функциональности: от поддержки минимального набора простейших функций до сложных сред с реализацией многопользовательской работы с моделями, интеграцией со средствами контроля версий и т. д.

В настоящий момент крайне актуальной стало решение проблемы по преодолению семантического разрыва между уровнями описания моделей жизненного цикла (производственными, информационными и т. п.), т. е. для каждого класса задач и процессов в их решения вводятся компонентные обобщения, связанные между собой посредством стандартизованных интерфейсов. При этом каждый из компонентов может иметь воз-

можность описывать задачу и методы ее решения на своем собственном языке. При этом все компоненты в комплексе образуют специфическое отражение окружающего мира в сознании человека, благодаря которому он решает задачу естественным путем на основе обобщений и ассоциаций, двигаясь к эффективному результату по прогнозируемому градиенту эмоциональной оценки ситуации. При этом сам процесс мышления рассматривается как способность сведения многоэкстремальной задачи, которую можно решать только методом перебора, к градиентной, одноэкстремальной. При этом проблемная область рассматривается, как совокупность предметной области и решаемых в ней задач (проблем), где под предметной областью понимается фрагмент реальной (виртуальной) действительности, представляемый некоторой совокупностью принадлежащих ему сущностей.

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены наиболее удобные и часто используемые методики визуального проектирования сложных технических систем. В результате анализа данных методик выявлены их достоинства и недостатки, описаны характерные черты основных методик. Предложен алгоритм, позволяющий оценивать характеристики проекта, сопоставлять их с характеристиками методик и выбирать наиболее подходящую для выполнения рассматриваемого проекта методику проектирования для каждого уровня экспертизы проекта.

Принимая во внимание характеристики объекта проектирования (сложные встраиваемые электронные системы), при рассмотрении концепции визуального проектирования наряду с удобством и гибкостью выбранного метода очень важным параметром является его взаимодействие с другими методами. Наиболее эффективно взаимодействие между различными уровнями проектирования может быть достигнуто при помощи единого пространства моделирования. При этом взаимодействие между моделями различных этапов должно быть описано на метауровне и реализовываться с помощью инструментальных средств автоматизированной системы визуального проектирования.

В связи с этим актуальным является создание универсального подхода проектирования, включающего в себя инструменты для проектирования на всех необходимых уровнях детализации. Вслед за этим возникает задача разработки автоматизированной системы визуального проектирования, использующей методики, включающие механизмы и гибкие инструменты проектирования и механизмы взаимодействия с концептуально-абстрактными, структурно-функциональными и объектными моделями, что является крайне важным для реализации полного цикла проектирования и предоставления пользователю универсального и гибкого инструмента.

ЛИТЕРАТУРА

1. *Билибин К. И., Власов А. И., Журавлева Л. В.* и др. Конструкторско-технологическое проектирование электронных средств / Под общ. ред. В. А. Шахнова. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2002. — 500 с.
2. *Денисов А. А., Колесников Д. М.* Теория больших систем управления. — Л.: Энергоиздат, 1982.
3. *Перегудов Ф. И., Тарасенко Ф. Л.* Введение в системный анализ. — М.: ВШ, 1989.
4. *Власов А. И.* Применение методов визуального моделирования для формализации конструкторско-технологической информации // Матер. Междунар. науч.-практич. конф. “Информатизация образования”. — Орел: ФГБОУ “ОГУ”, 2012. — С. 70—78.
5. *Якобсон А., Буч Г., Рамбо Дж.* Унифицированный процесс разработки программного обеспечения. — СПб.: Питер, 2002. — 496 с.
6. *Маклаков С. В.* Создание информационных систем с ALLFusion Modeling Suite. — М.: ДИАЛОГ-МИФИ, 2003. — 428 с.
7. *Иванова Г. С.* Технология программирования: учебник для вузов. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2002. — 320 с.
8. *Резчикова Е. В., Ревенков А. В.* Теория и практика решения технических задач: учеб. пособие. — М.: Форум, 2009. — 284 с.
9. *Власов А. И., Журавлева Л. В., Тимофеев Г. Г.* Методы генерационного визуального синтеза технических решений в области микро- наносистем // Научное обозрение. — 2013. — № 1. — С. 107—111.
10. *Власов А. И., Лыткин С. Г., Яковлев В. Л.* Краткое практическое руководство разработчика по языку PL/SQL. — М.: Машиностроение, 2000. — 64 с.
11. *Власов А. И.* Гексагональная понятийная модель визуального представления сложных производственных систем // Вестник МГТУ им. Н. Э. Баумана, сер. Приборостроение. — 2012. — Спец. выпуск № 5. — С. 157—169.
12. *Калянов Г. Н.* CASE технологии. Консалтинг при автоматизации бизнес-процессов. — М.: Горячая линия—Телеком, 2000. — 320 с.
13. *Репин В. В., Елиферов В. Г.* Процессный подход к управлению. Моделирование бизнес-процессов. — М.: РИА “Стандарты и качество”, 2004. — 404 с.
14. *Закис А.* RUP и другие методологии разработки ПО: принципы сравнения методологий разработки ПО // КомпьютерПресс. — 2006. — № 8. — С. 158—159.
15. *Питц-Моултис Н., Кирк Ч.* XML. Пер. с англ. — СПб.: БХВ-Петербург, 2001. — 736 с.
16. *Лихолетов В. В., Плужников В. Г., Комарова Е. В.* Инновационный менеджмент: учеб. пособие. — Челябинск: ЮУрГУ, 2005.
17. *Кознов Д. В.* Разработка и сопровождение DSM-решений на основе MSF // Системное программирование. — Вып. 3 / Под ред. А. Н. Терехова, Д. Ю. Булычева. — СПб.: СПбГУ, 2008. — С. 80—96.

Андрей Игоревич Власов — канд. техн. наук, зам. заведующего кафедрой “Проектирование и технология производства электронной аппаратуры” МГТУ им. Н. Э. Баумана.

☎ *(499) 263-65-53*

E-mail: vlasov@iu4.ru