Бердоносов В.Д., Редколис Е.В.
Berdonosov V.D., Redkolis E.V.

О КЛАССИФИКАЦИИ СИСТЕМ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ И СОЗДАНИЯ ПРОГРАММ (CASE)

ON CLASSIFICATION OF COMPUTER-AIDED SOFTWARE ENGINEERING SYSTEMS (CASE)

Бердоносов Виктор Дмитриевич – кандидат технических наук, доцент, профессор кафедры «Информационные системы» Комсомольского-на-Амуре государственного технического университета (Россия, Комсомольск-на-Амуре); 681024, г. Комсомольск-на-Амуре, пр. Интернациональный 59 − 5; +7 (962) 2875141. E-mail: ktriz@knastu.ru.
Berdonosov Victor D. – PhD in Engineering, Assistant Professor, Department of Information Systems, Komsomolsk-on-Amur State Technical University. 681024, Komsomolsk-on-Amur. E-mail: ktriz@knastu.ru

Редколис Елена Валерьевна – аспирант кафедры «Информационные системы» Комсомольского-на-Амуре государственного технического университета (Россия, Комсомольск-на-Амуре); 681013, Комсомольск-на-Амуре, пр. Ленина 42/3 – 6; +7 (924) 2257042. E-mail: Lulumzja@mail.ru.
Redkolis Elena V. – PhD Candidate, Department of Information Systems, Komsomolsk-on-Amur State Technical University. 681013, Komsomolsk-on-Amur. E-mail: Lulumzja@mail.ru

Аннотация. Авторы с различной степенью глубины ознакомились с несколькими сотнями CASE-систем. Предлагается рассмотреть эволюцию CASE-систем. При этом движущей силой эволюции является разрешение противоречий, возникающих на предыдущих стадиях, причём разрешение противоречий осуществляется инструментами ТРИЗ. Критериями, в соответствии с которыми происходило и происходит развитие CASE-систем, являются полезность и затратность. В работе выделены линии эволюции CASE-систем, их достоинства и недостатки, а также проанализированы назначение и тенденции развития каждой линии. Использование данного подхода для целей обучения позволяет значительно сократить время на освоение всего разнообразия CASE-систем за счёт систематизации знаний. С другой стороны, такая систематизация позволит, во-первых, выявить приоритетные направления дальнейшего развития CASE-систем, во-вторых, существенно упростит подбор подходящей для целей использования CASE-системы на предприятии и, в-третьих, апробировать применение инструментов ТРИЗ для разрешения противоречий, возникающих в CASE-системах.

Summary: The authors of the present paper have reviewed several hundreds of Computer-Aided Software Engineering (CASE) systems. It is proposed to examine the evolution of CASE-systems. Herewith, the motivating force of this evolution is viewed as the resolution of conflicts that appeared at the previous stages; such contradictions/conflicts are resolved using TRIZ tools. The criteria that are linked to the TRIZ concept of «ideality» and according to which CASE-systems develop are practicality and investment. In the paper, the CASE-systems development lines are singled out, their advantages and disadvantages and the purpose and development trends of each line are analyzed. Application of this approach for training allows reducing significantly the time needed for learning different CASE-systems, by means of knowledge systematization. On the other side, this systematization will allow 1) to identify the priority development areas for CASE-systems, 2) to simplify significantly the choice of CASE-systems being used at enterprises, and 4) to approve the TRIZ tools application for contradiction resolution in CASE-systems.

Материал поступил 15.07.2010

Ключевые слова: CASE, жизненный цикл программного обеспечения, процессно-ориентированный подход, систематизация знаний.

Keywords: CASE, software life cycle, process-oriented approach, systematization of knowledge

УДК 007

Введение

В работах [1; 2] предлагается использовать ТРИЗ для систематизации знаний исследуемой предметной области. Для относительно простой с точки зрения эволюции развитие предметной области (или её фрагмента) можно представить вектором. В этом векторе объекты исследуемой предметной области выстраиваются по степени увеличения идеальности в ТРИЗовском смысле, а переход от объекта к объекту реализуется как последовательное преодоление противоречий инструментами ТРИЗ. Для более сложных предметных областей рассматривается не вектор, а карта (матрица), в которой представлен вектор групп объектов, и каждая группа тоже в свою очередь представлена векторами. Например, для предметной области «Численные методы» [2, 17]: группы объектов – это математические модели реального физического мира, а сами объекты – это методы реализации соответствующей модели. Таким образом, для построения карты эволюции необходимо:

– собрать сведения об объектах исследуемой предметной области;
– выявить группы объектов и оценить идеальность каждой группы;
– расположить группы объектов по степени увеличения идеальности;
– выявить определяющее противоречие для каждой группы;
– определить инструмент ТРИЗ, разрешивший противоречие;
– проделать приведённые выше шаги для объектов каждой группы.

Использование такого подхода к классификации объектов рассматриваемой предметной области позволяет выделить логику систематизации, которая заключается в увеличении идеальности. Движение по линиям увеличения идеальности сопровождается логическими метками – приёмами разрешения противоречий. Это позволяет значительно быстрее освоить классификацию.

Эволюция CASE-систем

Первоначально термином «CASE» определялись инструменты автоматизированного проектирования и создания программных средств. Акроним «CASE» был введен компанией, занимавшейся производством программного обеспечения (ПО), «Nastec Corporation of Southfield» в 1982 г. по отношению к интегрированному графическому и текстовому редактору «GraphiText».

Однако буква «S» в акрониме «CASE» последние 5-6 лет трактуется в более широком смысле: и как первоначальное «software», и как «system». Обусловлено это тем, что ПО – частный случай систем вообще. В настоящее время к функциям CASE-систем относится поддержка практически всего жизненного цикла (ЖЦ) не только ПО, но и организационно-управляющих систем. CASE-системы автоматизируют методы проектирования, документирования и разработки структурированного компьютерного кода на желаемом языке программирования, проводят анализ и частичную оптимизацию систем и др.

В настоящее время, по некоторым публикациям, используются более трех сотен CASE-систем. Естественно, ориентироваться в таком многообразии достаточно тяжело.

Классификация CASE-систем

На сегодняшний день существует ряд классификаций CASE-систем, классификационными признаками в которых выступают: поддерживаемые этапы ЖЦ ПО, используемый тип (или вид) моделирования, степень интегрированности системы с СУБД, степень интегрированности по выполняемым функциям, применяемые методологии и модели, доступные платформы и т.д. Однако ни одна из них не дает исчерпывающей систематизации рассматриваемым системам.

Наиболее проработанной и обоснованной представляется классификация, созданная в 1993 г. А. Фуггетта [3] (профессор информатики в Политехническом университете Милана, старший исследователь в Центре исследования и образования в области информационных технологий (CEFRIEL)), положенная в основу данной работы.

В своей работе А. Фуггетта подробно рассматривает имеющиеся к 1993 г. попытки классифицирования CASE-систем, выявляет их достоинства и недостатки, а также представляет вниманию читателей собственный вариант классификации.

Классификация выполнена по категориям CASE-систем и отражает степень интегрированности по выполняемым функциям. Все рассматриваемые системы разделены им на три класса [3, 28]:

1 «Tools» – инструменты, поддерживающие только определенные задачи в процессах ЖЦ ПО и организационно-управляющих систем.

2 «Workbenches» – рабочие приложения, поддерживающие только определенный вид деятельности. Могут включать в себя системы класса «Tools» или иметь самостоятельную сложную модульную структуру.

3 «Environments» – программные среды, поддерживающие все (или большую часть) процессы ЖЦ ПО и организационно-управляющих систем. Могут включать в себя системы классов «Tools», «Workbenches».

Такое разделение актуально и в настоящее время.

В процессе работы нами были пересмотрены все классы в классификации А. Фуггетта и внесены некоторые изменения (см. рис. 1):

1 «Tools» [3, 29]:

а) *инструменты редактирования* («editing tools» или «editors»);

б) *инструменты программирования* («programming tools»), которые поддерживают различные функции программирования;

в) *инструменты верификации и валидации* («verification and validation tools»). Целью валидации является гарантирование того, что функции реализованного продукта полностью соответствуют желаниям потребителя. Целью верификации – гарантирование того, что структура продукта полностью соответствует установленным требованиям;

г) *инструменты управления конфигурацией* («configuration-management tools»), которые осуществляют управление и контроль функционирования сложных систем, спроектированных из многих частей;

д) *инструменты управления проектами* («project-management tools»).

2 «Workbenches» [3, 32]:

а) *рабочие приложения для бизнес-планирования и моделирования* («business planning and modeling workbenches»), которые поддерживают выявление и формализацию сложных бизнес-процессов;

б) *рабочие приложения для анализа и проектирования* («analysis and design workbenches»), которые автоматизируют большинство методологий анализа и проектирования;

в) *рабочие приложения для разработки пользовательского интерфейса* («user-interface development workbenches»);

г) *рабочие приложения для проектирования и программирования файлов и баз данных* («programming and designing of databases and files workbenches»), которые поддерживают расширенные функции программирования;

д) *рабочие приложения для верификации и валидации* («verification and validation workbenches»), которые содержат модули и встроенные системы тестирования;

е) *рабочие приложения для проведения обратного проектирования и технической поддержки* («maintenance and reverse-engineering workbenches»);

ж) *рабочие приложения для управления конфигурацией* («configuration-management workbenches»), которые поддерживают контроль версий, управление изменениями, учет состояния объектов конфигурационного управления, возможность разработки приложений «клиент-сервер» требуемой конфигурации и т.д.

3 «Environments» [3, 34]:

а) *инструментальные программные среды* («toolkits») – свободно интегрируемые наборы инструментальных средств, легко расширяемые посредством соединения различных CASE-систем классов «Tools» и «Workbenches». Поддержка ограничивается функциями программирования, управления конфигурацией и управления проектами;

б) *языко-ориентированные программные среды* («language-centered environments»), написанные под определенный язык, позволяют пользователям объединять несколько сред или расширять их функциональность посредством написания дополнительных модулей;

в) *интегрированные программные среды* («integrated environments»), которые имеют однородный совместимый интерфейс и общую базу данных, обеспечивающую централизованное управление информацией;

г) *программные среды четвертого поколения* («fourth generation environments»), которые поддерживают обработку сложно структурированных данных и разработку определенного класса программ: программы обработки электронных данных, бизнес-ориентированные приложения;

д) *процессо-ориентированные программные среды* («process-centered environments»), которые базируются на формализации бизнес-процессов. Разработка в таких системах осуществляется посредством автоматизации фрагментов процессов.

Рассмотренные примеры CASE-систем были распределены по группам классификации (см. рис. 1, приложение 1, табл. П1.1). Следует отметить, что примеры CASE-систем были отобраны на основе обзоров фирм-разработчиков систем, упоминаний в периодических изданиях, а также отзывов компаний, использующих CASE-системы.

Отметим, что данная классификация не является полностью законченной. Перспективным является выделение в классах «Tools» и «Workbenches» групп, в зависимости не от выполняемых функций, а от технологии и специфики их реализации (подобное выделение групп сейчас имеет место в классе «Environments»). Однако выполнение такой работы требует наличия большей информации о специфических особенностях построения и функционирования CASE-систем. Такого рода информация может быть получена от производителей и поставщиков CASE-систем, а также путем непосредственного тестирования доступных на рынке систем.

Направления эволюции CASE-систем

Рассмотрим эволюцию CASE-систем в виде матрицы [1]. Отметим, что при этом движущей силой эволюции является разрешение противоречий, возникающих на предыдущих стадиях, разрешение которых осуществляется инструментами ТРИЗ [2, 18], а сама эволюция характеризуется повышением идеальности.

Перечислим критерии, связанные с ТРИЗовским понятием «идеальности», в соответствии с которыми происходило и происходит развитие CASE-систем:

1 Критерии, отражающие полезность использования CASE-систем (распределены по фазам ЖЦ ПО).

1.1 Фаза формирования требований. Критерии:

1.1.1 Ввод и редактирование спецификаций требований и проектных спецификаций.

1.2 Фаза проектирования. Критерии:

1.2.1 Построение диаграмм:

1.2.1.1 Бизнес-процессов;

1.2.1.2 Данных;

1.2.1.3 Пользовательских диаграмм;

1.2.1.4 Прочих типов диаграмм.

1.2.2 Анализ моделей.

1.2.3 Контроль построения диаграмм:

1.2.3.1 Контроль соответствия декомпозиций диаграмм;

1.2.3.2 Контроль соответствия диаграмм разных типов.

1.2.4 Оптимизация:

1.2.4.1 Бизнес-процессов;

1.2.4.2 Данных.

1.2.5 Имитационное моделирование.

1.2.6 Синтаксический и семантический контроль проектных спецификаций.

1.2.7 Проектирование интерфейса:

1.2.7.1 Проектирование архитектуры ПО;

1.2.7.2 Прототипирование;

1.2.7.3 Генерация экранных форм.

1.2.8 Возможность трассировки.

1.2.9 Возможность создания библиотек моделей, словарей.

1.2.10 Автоматизированное проектирование отчетов.

1.3 Фаза реализации. Критерии:

1.3.1 Работа с кодом:

1.3.1.1 Синтаксически управляемое редактирование;

1.3.1.2 Генерация кода;

1.3.1.3 Компиляция кода;

1.3.1.4 Генерация структуры базы данных, а также фрагментов кода для СУБД;

1.3.1.5 Анализ правильности модели данных;

1.3.1.6 Обратный инжиниринг;

1.3.1.7 Анализ исходного кода;

1.3.1.8 Реструктуризация исходного кода.

1.3.2 Документирование:

1.3.2.1 Проверка полноты и непротиворечивости документации в соответствии со стандартами документирования;

1.3.2.2 Редактирование с помощью форм;

1.3.2.3 Возможности издательских систем;

1.3.2.4 Поддержка функций и форматов гипертекста;

1.3.2.5 Извлечение данных из хранилища;

1.3.2.6 Генерация документации по стандартам;

1.3.2.7 Генерация документации по спецификациям пользователя.

1.4 Фаза тестирования и отладки. Критерии:

1.4.1 Описание тестов;

1.4.2 Фиксация и повторение действий оператора;

1.4.3 Автоматический запуск тестовых примеров;

1.4.4 Регрессионное тестирование;
1.4.5 Автоматизированный анализ результатов тестирования;
1.4.6 Анализ тестового покрытия;
1.4.7 Анализ производительности;
1.4.8 Анализ исключительных ситуаций при тестировании;
1.4.9 Динамическое моделирование среды;
1.4.10 Отладка.

1.5 Фаза внедрения. Критерии:
1.5.1 Управление конфигурацией:
1.5.1.1 Контроль доступа и изменений;
1.5.1.2 Управление версиями;
1.5.1.3 Учет состояния объектов конфигурационного управления;
1.5.1.4 Архивирование.
1.5.2 Возможность разработки приложений «клиент-сервер» требуемой конфигурации.
1.5.3 Возможность адаптации проекта к различным системно-техническим платформам.

1.6 Прочие фазы (сопровождение, эксплуатация). Критерии:
1.6.1 Возможность экспорта данных (открытость архитектуры, количество совместимых приложений и форматов);
1.6.2 Интеграционные возможности, импорт данных;
1.6.3 Возможность поддержки нескольких языков программирования, сред разработки;
1.6.4 Управление проектами;
1.6.5 Поддержка коллективной работы.

2 Критерии, отражающие:
2.1 Стоимость приобретения;
2.2 Стоимость инсталляции;
2.3 Стоимость первоначальной адаптации к условиям конкретной фирмы;
2.4 Стоимость обучения;
2.5 Стоимость сопровождения;
2.6 Стоимость технической поддержки.

Для построения линий эволюции все рассматриваемые CASE-системы были оценены по выделенным критериям. Оценка производилась на основании материалов обзоров производителей и поставщиков систем, а также на основании отзывов покупателей и на основании опыта работы авторов данной статьи с некоторыми системами.

Кроме того, при построении линий развития была выделена группа свободно распространяемых CASE-систем (псевдоидеальных систем). К системам этой группы относятся, как правило, малофункциональные продукты класса Tools, свободно предоставляемые на рынке CASE-систем: ArchE, Topcased, AmaterasUML, Umbrello UML Modeller, Open ModelSphere, StarUML, Jink-uml, Fujaba, Poseidon, MOSKitt, BOUML, Acceleo, Astade, Gmodeler, Case/4/0, Violet, ArgoUML, TAU, SQLyog, Software Ideas Modeler, UMLet, ADONIS, Dia, dbForge Studio for MySQL.

С точки зрения здравого смысла системы этой группы, ввиду ограниченности своей функциональности, находятся в начале вектора эволюции (как и все системы класса «Tools» по отношению к классам «Workbenches» и «Environments»). Однако, возможно, получение дополнительной информации о значениях критериев 2.2-2.5 для таких систем поможет более точно определить место систем в общей линии эволюции.

Графики идеальности, полезности и затратности CASE-систем классов «Tools» и «Workbenches» представлены на рис. П1.1, *а* и рис. П1.2, *а* приложения 1. Также на графике присутствует линия скорректированной идеальности.

Коррекция идеальности была выполнена по причине того, что многие системы имеют расширенную функциональность в рамках только одной фазы ЖЦ и не поддерживают другие фазы (или не имеют возможности интеграции с системами, поддерживающими другие фазы). Коррекция была произведена путем умножения показателя идеальности на отношение количества фаз ЖЦ, поддерживаемых определенной системой, к количеству фаз ЖЦ в целом, охватываемых CASE-системами.

Формулирование противоречий

После того как для каждой рассматриваемой CASE-системы были получены значения идеальности, примеры конкретных систем были соотнесены с классификацией (см. рис. 1). Это позволило сформировать карту эволюции (см. рис. 2).

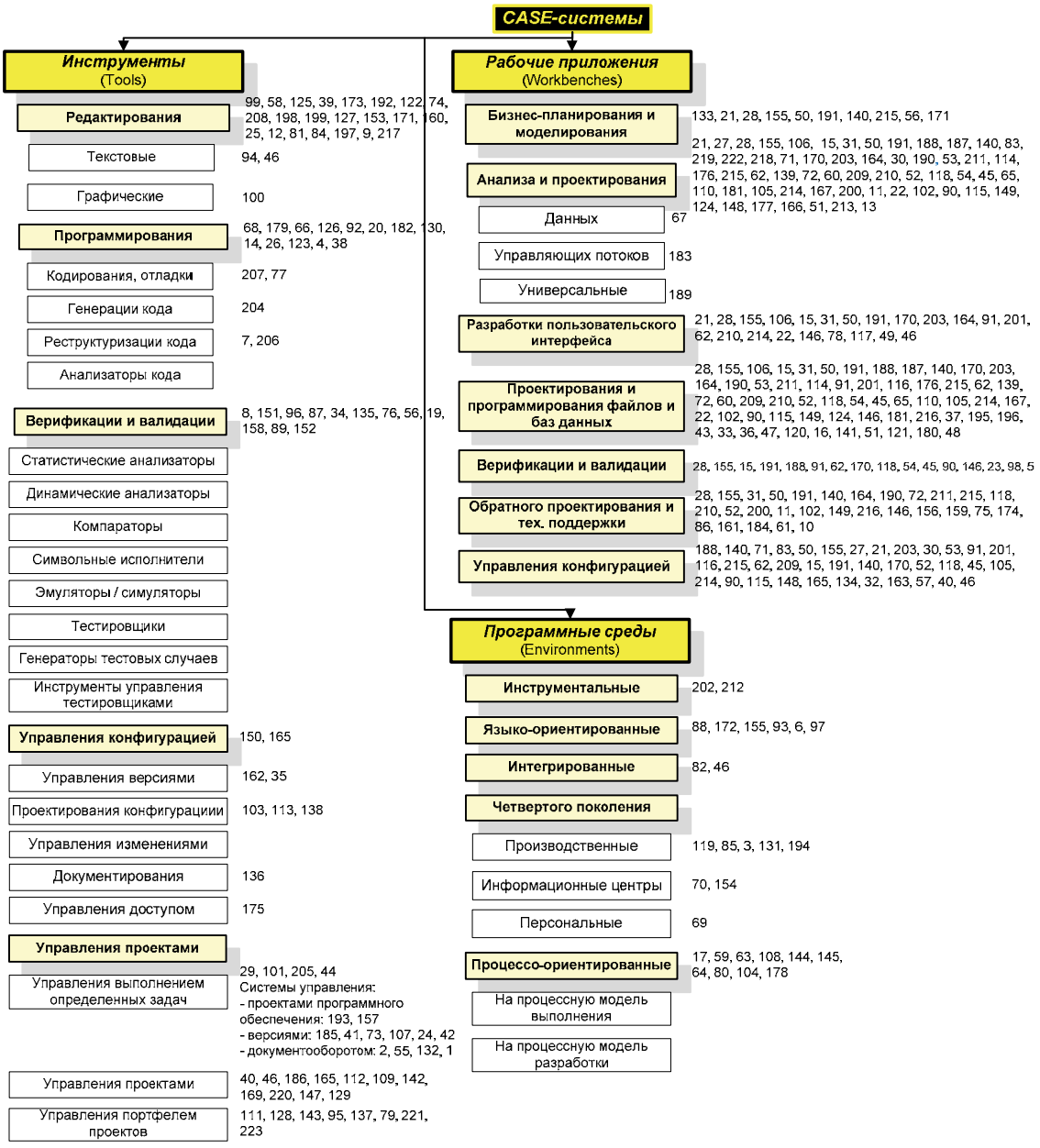

Рис. 1. Классификация CASE-систем

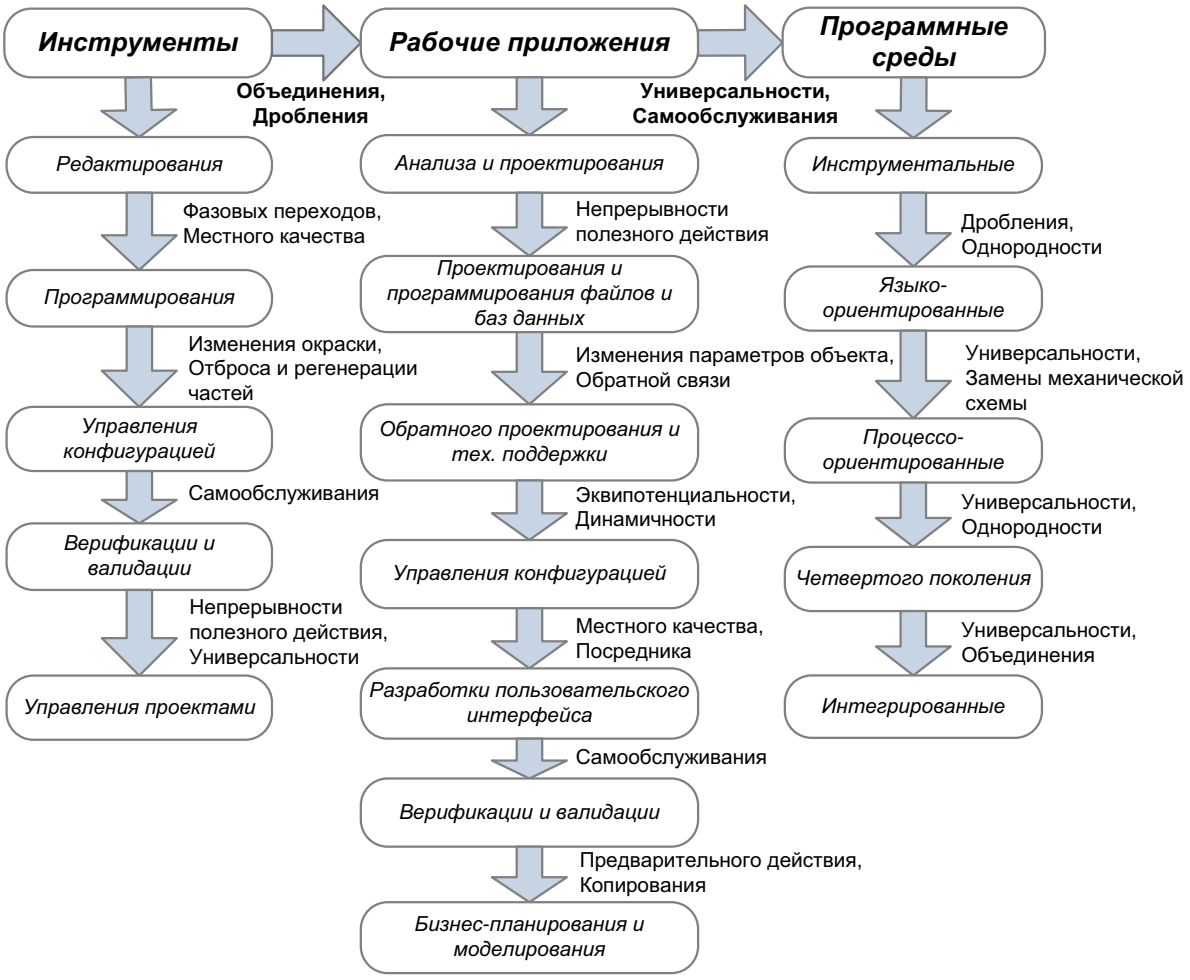

Рис. 2. Карта эволюции CASE-систем

В основной линии развития CASE-систем были выделены следующие *противоречия* (см. рис. 3):

1 противоречие на этапе появления систем класса «Workbenches»: при увеличении количества выполняемых функций недопустимо растет требуемый уровень интеграции системы с другими платформами и программными продуктами;

2 противоречие на этапе появления систем класса «Environments»: при увеличении количества охватываемых этапов ЖЦ проектируемых систем недопустимо увеличиваются затраты на получение результатов.

Аналогично эволюции классов «Tools» и «Workbenches» представлялась эволюция для каждого класса программных продуктов.

Противоречия, возникающие в процессе эволюции CASE-систем, были разрешены с использованием специальных приемов разрешения противоречий (см. рис. 4).

Приведем пример выявления противоречий в классе систем «Environments».

Системное свойство программных продуктов данной группы – поддержка различных фаз ЖЦ ПО и организационно-управляющих систем. *Главная полезная функция* – поддержка большинства фаз ЖЦ ПО и организационно-управляющих систем. *Объект воздействия* – различные задачи в рамках фаз ЖЦ. *Источник энергии* – данные различного вида (графические, текстовые, табличные). *Двигатель* – математическая модель преобразования данных. *Трансмиссия* – механизмы анализа преобразованных данных. *Рабочий орган* – интерфейс CASE-системы. *Орган управления* – интерфейс управления. *Параметры системы*:

П1: фазы ЖЦ, требующие «поддержки» («1» – формирование требований, «2» – проектирование, «3» – реализация, «4» – тестирование и отладка, «5» – внедрение, «6» – сопровождение и эксплуатация);

П2: тип системы («1» – ПО, «2» – организационно-управляющая).

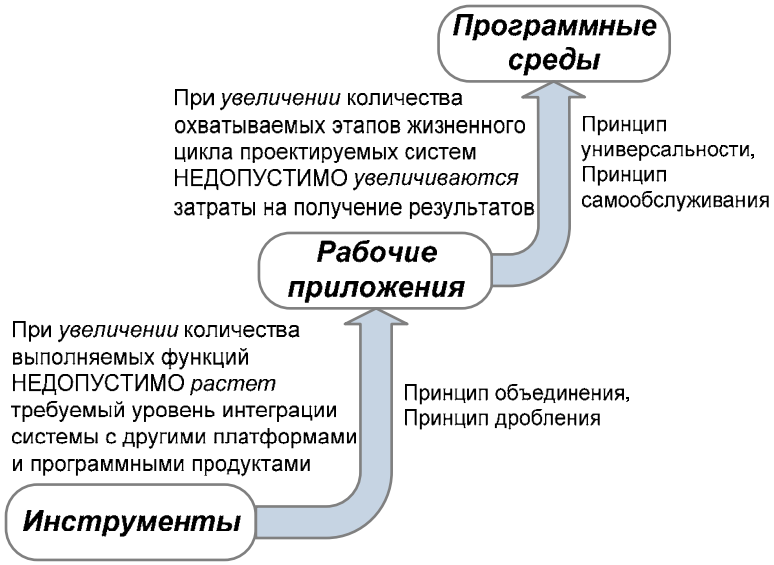

Рис. 3. Разрешение противоречий

«Инструментальные программные среды» [П1: 3, 5. П2: 1, 2]. Поддержка в системах ограничена функциями программирования, управления конфигурацией и управления проектами. Возникает *противоречие*: при *увеличении* количества выполняемых функций НЕДОПУСТИМО *растет* требуемый уровень интеграции системы с другими платформами и программными продуктами.

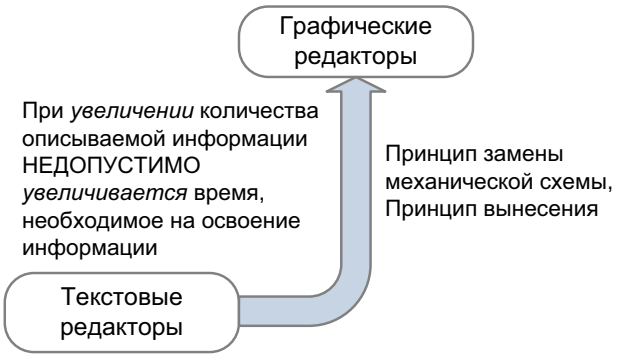

Рис. 4. Эволюция инструментов редактирования («editing tools»)

Противоречие разрешается при помощи использования принципа «дробления» и принципа «однородности»: осуществляется переход к «Языко-ориентированным программным средам» − системам, которые обеспечивают легкое расширение среды посредством написания дополнительных модулей на «заданном» языке.

Переход к «Языко-ориентированным программным средам» [П1: 3, 4, 5. П2: 1, 2] порождает новые проблемы. Код, написанный на другом языке программирования в такой сис-

теме, не может быть выполнен. Возникает *противоречие*: при *увеличении* уровня интеграции НЕДОПУСТИМО *сокращается* количество доступных языков программирования.

Противоречие разрешается при помощи использования принципа «универсальности» и принципа «замены механической схемы»: осуществляется переход к «Процессо-ориентированным программным средам», которые поддерживают крупномасштабную разработку без привязки к конкретному языку.

В свою очередь, использование «Процессо-ориентированных программных сред» [П1: 1, 2, 3, 4, 5, 6. П2: 1, 2] сталкивается с проблемой обязательной графической формализации бизнес-процессов и т.д.

Заключение

Двумерная классификация (карта эволюции) CASE-систем, представленная в данной статье, является наиболее эффективной для целей обучения, а также для систематизации информации о большинстве CASE-систем, имеющихся в настоящее время на рынке программного обеспечения. Такая систематизация существенно упростит подбор подходящей для определенных целей использования на предприятии CASE-системы.

В представленной классификации по одной координате располагаются классы CASE-систем («Tools» – инструменты, «Workbenches» – рабочие приложения, «Environments» – программные среды), а по другой – линии эволюции систем каждого класса. CASE-системы в рамках одной линии выстроены по степени увеличения их идеальности.

Отметим также, что дальнейшая разработка карты эволюции CASE-систем путём достройки позволит предсказать направления развития как классов систем, так и линий эволюции в рамках каждого класса.

Наиболее эволюционированными системами в классе «Environments» являются интегрированные CASE-системы: DEC Cohesion, IBM AD/Cycle; в классе «Workbenches» – системы бизнес-планирования и моделирования: ARIS (Business Performance Edition), CA ERwin Modeling Suite, Rational Rose, Oracle Designer, Borland Together и др.

ЛИТЕРАТУРА

1. Berdonosov, V. «Fractality of knowledge and TRIZ», Proceedings of the ETRIA TRIZ Future Conference, Kortrijk, 9-11 October 2006, published by CREAX Press, ISBN 90-77071-05-9. – pp. 31-36.
2. Berdonosov, V., Redkolis, E. «TRIZ-fractality of mathematics» Proceedings of the ETRIA TRIZ Future Conference, Frankfurt on Mian, 6-8 November 2007, published by Kas-sel University Press GmbH, ISBN 978-3-89958-340-3. – pp. 15-22.
3. Fuggetta, A. A Classification of CASE Technology. Dipartimento di Elettronica ed Inf., Politecnico di Milano: Computer, vol. 26, no. 12, Dec. 1993, doi:10.1109/2.247645. – pp. 25-38.

ПРИЛОЖЕНИЕ 1

Таблица П1.1

Спецификация CASE-систем

	Название системы		Название системы
1.	АРХИВНОЕ ДЕЛО	112.	Microsoft Project
2.	ЕВФРАТ-Документооборот	113.	MMS
3.	4 GL/Online	114.	Model Maker
4.	Acceleo	115.	Modelio
5.	Act	116.	MOSKitt
6.	Ada	117.	Multi/CAM
7.	AdaReformat	118.	MySQL Workbench
8.	AdaXRef	119.	Natural 2
9.	ADONIS	120.	NETRON/CAP
10.	Adpac CASE Tools	121.	NewEra
11.	Agilej	122.	ObjectDomain
12.	AmaterasUML	123.	Objecteering MDA Modeler
13.	Analyst/designer	124.	ObjectiF
14.	AndroMDA	125.	OmniGraffle
15.	AnyLogic	126.	Omondo
16.	APS	127.	Open ModelSphere
17.	Arcadia	128.	OpenAir
18.	ArchE	129.	OpenProj
19.	Arena	130.	OptimalJ
20.	ArgoUML	131.	Pacbase
21.	ARIS	132.	PayDox
22.	Astade	133.	PC Prim
23.	Battlemap	134.	PCMS
24.	Bazaar	135.	Playback
25.	Bonapart	136.	Plib86
26.	BOUML	137.	PM.Contract
27.	Business Studio	138.	Pmaker
28.	CA ERwin Modeling Suite	139.	Poseidon
29.	CA Estimacs	140.	Power Designer Process Modeler
30.	CASE.Аналитик	141.	PowerBuilder
31.	Case/4/0	142.	Primavera
32.	CCC	143.	Primavera ProSight
33.	Chen Toolkit	144.	Process Weaver
34.	CICS Simulcast	145.	Process Wise
35.	CMS	146.	PRO-IV Workbench
36.	COBOL2/Workbench	147.	Projectmate
37.	Code Center	148.	ProKit*Workbench
38.	CodeLogic	149.	Prosa UMLmodeler
39.	Concept Draw	150.	PVCS (Intersolv)
40.	Coordinator	151.	Q/Auditor
41.	CVS	152.	Quality Works
42.	Darcs	153.	QuickCRC
43.	Data Base Design	154.	Ramis
44.	DateBook	155.	Rational Rose
45.	dbForge Studio for MySQL	156.	Recorder
46.	DEC Cohesion	157.	Redmine
47.	DECASE	158.	Re-Think
48.	Delphi	159.	Rigi
49.	Design/OA	160.	SAP Strategic Management

Продолжение табл. П1.1

	Название системы		Название системы
50.	Designer Oracle	161.	Scan/COBOL
51.	Developer	162.	SCCS
52.	Devgems Data Modeler	163.	SCLM
53.	DeZign forData-bases	164.	S-Designor
54.	Dia	165.	SE Companion
55.	DocsVision	166.	SELECT
56.	DOORS	167.	Select Architect
57.	DSEE	168.	Semtalk
58.	DVDraw	169.	SensoryProTracker
59.	East	170.	Silverrun
60.	EasyCASE	171.	Simprocess
61.	Ensemble	172.	Smalltalk
62.	Enterprise Architect	173.	SmartDraw
63.	Enterprise II	174.	SmartSystem
64.	EPOS	175.	SoDA
65.	ER / Studio	176.	Software Ideas Modeler
66.	ESS-Model	177.	Software Through Pictures
67.	Excelerator	178.	SPADE/SLang
68.	Extend	179.	SQL Maestro
69.	Filemaker Pro	180.	SQLWindows
70.	Focus	181.	SQLyog
71.	Fox Manager	182.	StarUML
72.	Fujaba	183.	Statemate
73.	Git	184.	SuperStructure
74.	Gmodeler	185.	SVN
75.	Hindsight	186.	Synchronize
76.	HP Basic Branch Analyzer	187.	System Architect
77.	HP Cross Compilers	188.	TAU
78.	HP Interface Architect	189.	TeamWork
79.	HP Project and Portfolio Management	190.	Toad Data Modeler
80.	HPSF	191.	Together Borland
81.	HyperionPerformance Scorecard	192.	Topcased
82.	IBM AD/Cycle	193.	Trac
83.	IBM WebSphere Business Modeler	194.	Transform
84.	IDEF0.EMTools	195.	Turbo C++
85.	Informix	196.	Turbo Pascal
86.	Inspector/Recorder	197.	UFO-toolkit
87.	Instrumentation Tool	198.	Umbrello UML Modeller
88.	Interlisp	199.	UMLet
89.	Ithink Analyst	200.	UMLStudio
90.	iUML	201.	Uniface
91.	JAM	202.	UNIX Programmer’s Workbench
92.	Jink-uml	203.	Vantage Team Builder
93.	KEE	204.	VAX Cobol Generator
94.	KeyOne	205.	VAX Notes
95.	Kildrummy	206.	Via/Renaisance
96.	lint-Plus	207.	Via/Smarttest
97.	Lisp	208.	Violet
98.	Logiscope	209.	Visible Analyst Workbench
99.	MacBubbles	210.	Visio
100.	MacDraw	211.	Visual Paradigm Suite
101.	MacProject	212.	VMS VAX Set

Окончание табл. П1.1

	Название системы		Название системы
102.	Magic Draw	213.	vsDesigner
103.	Make	214.	Webratio
104.	Marvel	215.	Win(MAC)A&D
105.	Mavim Rules	216.	Win(Mac)Translator
106.	MEGA	217.	Workflow Modeler
107.	Mercurial	218.	Бизнес-Инженер
108.	Merlin	219.	ИНТАЛЕВ
109.	MetaFour	220.	Команд
110.	Metastorm Pro Vision	221.	ЛИДЕР
111.	MS Office Project Portfolio Server	222.	ОРГ-Мастер Про

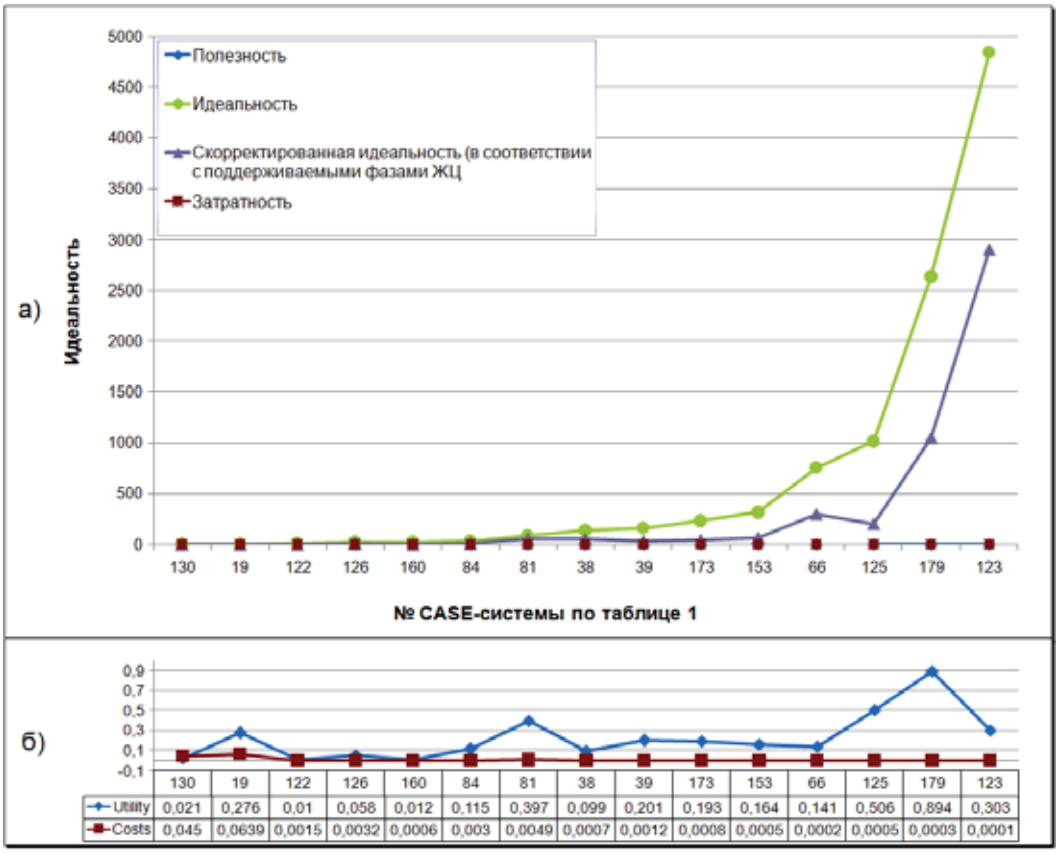

Рис. П1.1. Эволюция систем группы «Tools»:
а – графики полезности, затратности, идеальности, скорректированной идеальности;
б – графики полезности и затратности в более крупном масштабе

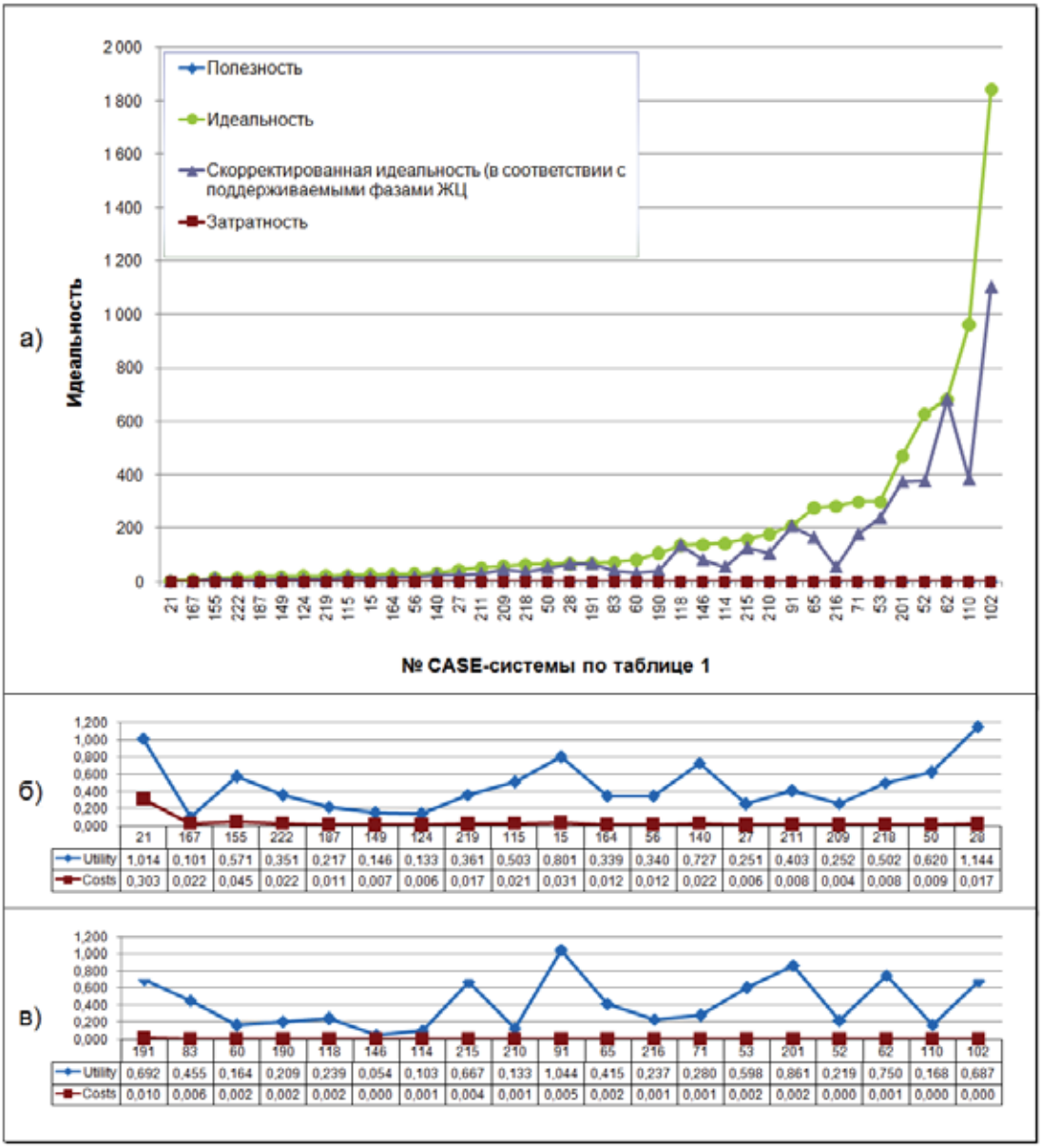

Рис. П1.2. Эволюция систем группы «Workbenches»:
а – графики полезности, затратности, идеальности, скорректированной идеальности;
б, в – графики полезности и затратности в более крупном масштабе