Методы и технологии реинжиниринга ИС

К.В. Ахтырченко, Т.П. Сорокваша

Аннотация. Сегодня можно говорить, что эра, когда разработчики информационных систем приходили в организацию и начинали проекты информатизации «с нуля», прошла. Большинство организаций уже имеет некоторые информационные системы (ИС), которые со временем начинают требовать целенаправленной деятельности по их реинжинирингу. В настоящей статье исследуются существующие подходы, методы и технологии реинжиниринга ИС, предлагается подход к их классификации. На основании результатов исследований и вводимой классификации дается оценка текущего состояния в данной области.

Ключевые слова: реинжиниринг ИС, методы и технологии реинжиниринга ИС, подход к классификации методов и технологий реинжиниринга, модернизация ИС, миграция ИС, эволюция ИС, реструктуризация ИС, унаследованные ИС.

1. Введение

Результаты исследований состояния информатизации в различных организациях позволяют сделать вывод, что в настоящий момент большинство из них уже имеет некоторые информационные системы (ИС). Эти ИС в различной степени автоматизируют процессы, протекающие в организациях.

Исследования проектов информатизации, и, в первую очередь, проектов разработки ИС так же показывают, что создание новой информационной системы в большинстве случаев предусматривает изменение состояния существующих ИС. Типичными стали проекты:

- по разработке новых ИС и их интеграции с существующими ИС;
- по разработке новых ИС с целью замены существующих ИС;
- по модернизации (наращиванию функциональности, развитию) существующих ИС.

По сути, сегодня можно говорить, что эра, когда разработчики ИС приходили в организацию и начинали проекты информатизации «с нуля», прошла. Наступает время проектов по систематической трансформации существующих ИС или эра реинжиниринга ИС.

Следствием сложившейся ситуации становится объективная потребность в исследовании, пересмотре и переосмыслении существующих подходов, методологий и технологий разработки ИС, что, в свою очередь, может потребовать их модернизации, а возможно, и разработки новых решений.

Ситуация осложняется тем, что в настоящий момент различными исследователями и практиками понятие реинжиниринга ИС трактуется по разному. Во многом это обусловлено необычайно широким спектром задач по реинжинирингу, с которыми приходится сталкиваться в реальных проектах.

Сегодня в мире существует большое количество подходов, методов и технологических решений, напрямую или косвенно соотносимых с деятельностью по реинжинирингу ИС. Однако они не интегрированы на уровне методологий (процессов разработки). Как результат, можно наблюдать наличие огромного количества методологий, где основной акцент сделан на разработку ИС «с нуля», и практическое отсутствие «стройных» методологий, целью создания которых являлось бы комплексное, целостное решение задач реинжиниринга ИС.

В настоящей статье исследуются существующие подходы, методы и технологии реинжиниринга ИС, предлагается подход к их классификации. На основании результатов исследований и вводимой классификации дается оценка текущего состояния в данной области.

Заметим, что в процессе написания статьи не ставилась цель исследовать все решения в области реинжиниринга ИС, представить детальную информацию по каждому из них. В то же время авторами была предпринята попытка рассмотреть основные подходы, методы, технологии и инструментальные средства, которые характеризуют состояние дел в данной области. Детальную информацию об интересующих решениях можно найти в литературе, ссылки на которую представлены в конце статьи, а так же на следующих Web – сайтах:

- http://www.informatik.uni-stuttgart.de/ifi/ps/reengineering/
- http://www.iam.unibe.ch/
- http://www.stsc.hill.af.mil/reng/
- http://www.sei.cmu.edu/reengineering/index.html
- http://www.tcse.org/revengr/

2. Понятие «реинжиниринга ИС», его содержание и место в ЖЦ ИС

Существует ряд работ [1, 2, 5, 9, 10, 13, 15, 16, 20, 34-36], в которых в различной степени определяется соотносимая с реинжинирингом ИС деятельность, выявляется и исследуется место реинжиниринга в ЖЦ ИС.

2.1. Понятие «реинжиниринга ИС»

Сразу следует признать, что в настоящий момент понятие «реинжиниринг ИС» не является повсеместно устоявшимся. Как следствие довольно часто возникает определенная терминологическая путаница. Авторами исследуются одни и те же проблемы, подходы, методы и технологии их решения, однако в качестве базовых понятий, наряду с «реинжинирингом ИС» [1, 9, 16, 20] употребляются

«эволюция ИС» [10, 13], «миграция ИС» [15], «модернизация ИС» [2], «реструктуризация ИС» [5].

Нельзя отрицать, что деятельность по миграции ИС имеет определенную специфику (окраску) по отношению к деятельности по модернизации ИС. Однако, принимая во внимание определение реинжиниринга ИС, приводимое в [1]:

«Реинжиниринг представляет собой систематическую трансформацию существующей системы с целью улучшения ее характеристик качества, поддерживаемой ею функциональности, понижения стоимости ее сопровождения, вероятности возникновения значимых для заказчика рисков, уменьшения сроков работ по сопровождению системы», становится очевидным, что и миграция, и модернизация ИС являются частью деятельности по реинжинирингу ИС. Как результат, подходы, методы и технологии миграции, модернизации, эволюции ИС, следует считать частью методологического и инструментально-технологического обеспечения процесса реинжиниринга ИС.

Такой взгляд на реинжиниринг ИС согласуется с таксономией, вводимой в ряде работ [34, 36, 38-40]. В этих работах авторами делается попытка выстроить систему понятий, соотносимых с данным видом деятельности. Так, в [38] реинжиниринг ИС определяется как «исследование (изучение, обследование) и перестройка исходной системы с целью ее воссоздания в новой форме с последующей реализацией этой новой формы»[1]. Далее, в контексте деятельности по реинжинирингу вводятся и определяются такие важные понятия, как:

- прямой инжиниринг (Forward engineering);
- редокументирование (Redocumentation);
- рефакторинг (Refactoring);
- реструктуризация (Restructuring);
- переориентация (Retargeting);
- обратный инжиниринг (обратное проектирование) (Reverse engineering);
- сопровождение программных продуктов (Software maintenance);
- трансляция исходного кода (Source Code Translation);
- и т.д.

Перечисленные понятия раскрывают понятие «реинжиниринг ИС», а соотносимая с ними деятельность рассматривается либо как одна из форм реинжиниринга ИС, либо как подпроцесс процесса реинжиниринга. Определения этих и других понятий приводятся в приложении к данной статье в глоссарии понятий и терминов. При этом в тех случаях, когда проявлялись отличия в определении того или иного понятия, в глоссарий включались альтернативные варианты определения, а при необходимости, и поясняющие определение комментарии.

[1] Далее в настоящей работе при исследовании существующих решений употребляется оригинальная терминология, предлагаемая авторами работ. При этом в качестве рабочего варианта используется определение реинжиниринга ИС, приводимое в [1].

2.2. Основное содержание реинжиниринга ИС и его место в ЖЦ ИС

Следующим шагом на пути исследования подходов, методов и технологий реинжиниринга ИС следует считать определение основного содержания деятельности по реинжинирингу ИС, места реинжиниринга в ЖЦ ИС.

Так, в [1] авторы придерживаются следующей позиции при определении границ деятельности по реинжинирингу, и, как следствие, места реинжиниринга в ЖЦ ИС.

Утверждается, что реинжиниринг ИС занимает промежуточное местоположение по отношению к разработке и сопровождению ИС. При этом сопровождение ИС рассматривается как деятельность, предусматривающая выполнение изменений, направленных на коррекцию, усовершенствование и адаптацию ИС, а разработка ИС как деятельность, включающая реализацию новых возможностей, добавление новой функциональности, осуществление таких существенных улучшений, как переход на использование новых компьютеров, внедрение новых информационных технологий. Авторами правомерно утверждается, что реинжиниринг характеризуется деятельностью, как по сопровождению, так и по разработке ИС. При этом эти два вида деятельности в контексте реинжиниринга ИС могут существенно перекрываться.

В отличие от [1], работа [2] посвящена вопросам модернизации унаследованных информационных систем. В ней делается обзор различных методик модернизации унаследованных систем, определяется роль модернизации систем в процессе управлениях их эволюцией. Утверждается, что коммерческий рынок предлагает разнообразные решения проблемы модернизации унаследованных систем, которая становится все более актуальной. В сложившейся ситуации понимание преимуществ и слабых мест каждой такой методики является чрезвычайно важным, поскольку обуславливает выбор правильного решения, и как следствие успех деятельности по модернизации системы.

В контексте исследований, связанных с эволюцией ИС, авторами выделяются деятельности по сопровождению, модернизации и замещению ИС.

Определяется, что сопровождение представляет собой инкрементальный итеративный процесс, в рамках которого выполняются малые изменения в системе, не затрагивающие структурной организации системы (архитектуры системы).

В отличие от сопровождения, модернизация характеризуется как деятельность, которая предусматривает значительные изменения существующей системы (в том числе в ее структуре), но не ее утилизацию или замещение новой системой.

И, наконец, замещение рассматривается как процесс, который заключается во внедрении новой системы, способной полностью заменить существующую ИС. Замещение обычно применяется к системам, которые недокументированны,

устарели или не расширяемы, расходятся с требованиями бизнеса и для которых модернизация невозможна или экономически не выгодна.

Определяя место этих видов деятельности в контексте ЖЦ ИС, авторы рассматривают следующую последовательность их выполнения (см. Рис. 1).

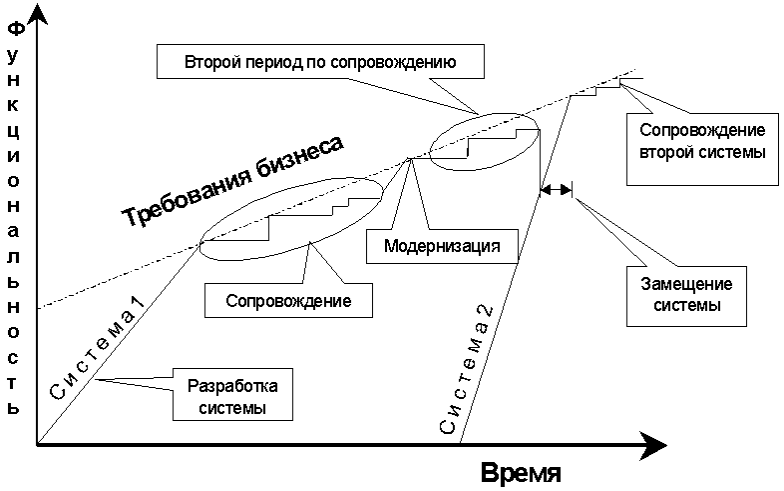

Рис. 1. *Жизненный цикл ИС.*

Первоначально, осуществляется разработка (построение) ИС. Далее выполняется деятельность по ее сопровождению. В процессе сопровождения возникает необходимость в реструктуризации системы и как следствие выполняется деятельность по ее модернизации. После этого снова осуществляется деятельность по сопровождению системы. В тот момент, когда ИС перестает удовлетворять заказчика, осуществляется ее замещение на новую систему и последовательность выполняемых видов деятельности повторяется. Безусловно, в реальной жизни может возникать и другая последовательность, когда, сразу, не прибегая к модернизации, осуществляется замещение одной ИС другой. Однако общий подход к последовательности выполнения сопровождения, модернизации и замещения останется неизменным.

Приводимые в [2] виды деятельности основаны на таксономии, вводимой в [16]. В отличие от предыдущей работы, в [16] в качестве базового понятия используется именно «реинжиниринг», а вводимая таксономия рассматривается как множество видов деятельности, соотносимых с реинжинирингом ИС. Применительно к унаследованным ИС выделяются виды деятельности по оценке, сопровождению, трансформации, замещению ИС, а также используются смешанные стратегии, предусматривающие совместное выполнение перечисленных видов деятельности.

Обеспечивая концептуальное понимание процесса реинжиниринга ИС, в ряде работ [1, 9] определяются основные виды деятельности (фазы), соотносимые с этим процессом.

Так, в [1] рассматриваются следующие основные фазы:

- оценки показателей проекта по реинжинирингу, в том числе характеристик унаследованной информационной системы (фаза оценки);
- анализа решений по реинжинирингу, в том числе принятие решения о необходимости проведения работ по реинжинирингу или сопровождению/разработке ИС;
- осуществления реинжиниринга (выполнения работ по реинжинирингу);
- внедрения системы, трансформированной в результате проведения реинжиниринга.

Другой подход к определению деятельности по реинжинирингу базируется на так называемой модели «подковы» [9, 21].

В основу данной модели положены следующие процессы (виды деятельности), соотносимые с реинжинирингом ИС:

- анализ существующей системы, основанный на одном или более ее логических описаний;
- трансформация этих логических описаний в новое, улучшенное логическое описание системы.
- разработка новой системы, основанной на новых логических описаниях системы.

Эти три основных процесса соединяются в модели в виде «подковы» (см. Рис. 2).

Рис.2. *Модель «подковы».*

Первый процесс (Architecture Recovery) предусматривает восстановление архитектуры существующей системы посредством извлечения на основании

исходного кода характеризующих ее артефактов. Полученная архитектура анализируется на предмет соответствия требованиям к изменяемости, надежности, защищенности и так далее.

Второй процесс (Architecture Transformation) заключается в трансформации (реинжиниринге) восстановленной архитектуры к желаемой новой архитектуре. Полученная в результате трансформации новая архитектура оценивается с позиции ее качества с учетом накладываемых на нее организационных и экономических ограничений.

И, наконец, третий процесс (Architecture-based Development) включает деятельность по разработке системы, соответствующей новой архитектуре. Здесь решаются вопросы декомпозиции элементов системы по пакетам, осуществляется выбор стратегий взаимодействия элементов/компонентов системы. В рамках данного процесса так же обеспечивается интеграция в новую систему артефактов унаследованной системы, например, посредством переписывания части унаследованного кода и/или применения технологии построения оболочек для компонентов унаследованной системы.

С моделью «подкова» соотносится три уровня, на каждом из которых может осуществляться трансформация существующей системы.

Представление на уровне структуры кода (Code-Structure Representation) включает программный код, а так же такие артефакты, как абстрактные синтаксические деревья и графы потоков (flow graphs), получаемые на основании выполнения различного рода аналитических операций (например, разбора кода). Текстуальный перевод, а так же перевод на основе синтаксических деревьев являются примерами трансформации системы на этом уровне.

В отличие от предыдущего уровня, представление на уровне функциональности системы (Function-Level Representation) предусматривает определение связей между функциями программ (например, последовательности вызовов), данными (как частный случай, связей между сущностями данных, соотношений данных и функций) и файлами (к примеру, распределение функций и данных по файлам). Трансформация системы на данном уровне осуществляется на основании пересмотра функциональности системы и заключается, например, в переходе от функционального подхода к анализу и проектированию к объектному подходу, в переходе от реляционной БД к объектной БД.

На архитектурном уровне артефакты предыдущих уровней, объединяются в подсистемы в терминах архитектурных компонентов архитектуры ИС. Трансформация системы на этом уровне предусматривает коренные изменения в структуре программы, в том числе в части применения основных образцов взаимодействия компонентов: типов программных компонентов, используемых соединителей (connectors), вариантов декомпозиции функциональности, образцов управления и обмена данными времени выполнения.

Согласно модели «подкова» трансформация системы, в зависимости от ситуации, может осуществляться на каждом из представленных уровней, при этом более низкий уровень поддерживает трансформацию на более высоком уровне.

Следует признать, что модель «подкова» находит широкое применение в рамках деятельности, связанной с реинжинирингом ИС. Так, в [20] на ее основе определяются требования и основной каркас для интеграции инструментальных средств реинжиниринга на архитектурном уровне и уровне программного кода. С учетом соотносимых с ней процессов и уровней представления, осуществляется расширение модели CORUM (Common Object-based Reengineering Unified Model). Эта модель, в свою очередь, разрабатывалась:

- для представления требуемой для систем управления на основе программного кода (Code-based Management System) информации о программных средствах;
- для обеспечения интероперабельности между системами данного класса (в том числе между средствами реинжиниринга программ на уровне программного кода).

В контексте вводимых расширений в этой же работе для модели «подкова» определяется семантическая модель, обеспечивающая на основании формулируемых унифицирующих свойств семантическую связь между архитектурным уровнем и уровнем программного кода.

Подход, предложенный в [34-36], очень близок к подходу, основанному на модели «подкова». Характеризуя жизненный цикл реинжиниринга ИС, авторы определяют следующие шаги процесса реинжиниринга:

- анализ требований для выявления конкретных целей реинжиниринга унаследованной системы;
- восстановление модели, в том числе документирование и понимание структуры унаследованной системы;
- выявление проблем, связанных с унаследованной системой;
- анализ проблем, включающий выбор архитектуры, позволяющий устранить обнаруженные в унаследованной системе дефекты;
- реорганизация, включающая выбор и применение оптимального подхода трансформации унаследованной системы;
- распространение изменений.

В отличие от предыдущих работ, в [10, 13] деятельность по реинжинирингу рассматривается в качестве одной из форм деятельности по эволюции унаследованных ИС, когда, требуется более сильнодействующее «лекарство» для «лечения» ИС, нежели серия локальных инкрементальных изменений [13].

Так, в [13] описывается каркас (enterprise framework), характеризующий:

- глобальную среду, в которой осуществляется эволюция системы;
- действия, процессы и рабочие продукты (артефакты), которые сопровождают деятельность по эволюции системы.

Авторами подчеркивается, что помимо технических вопросов, связанных с эволюцией унаследованных ИС, существует так же множество организационных вопросов. Например, «Как планировать эволюцию большой сложной системы, включая ее реинжиниринг?», «Какие существуют критичные факторы успеха эволюции системы?», «Как оценить, что люди, осуществляющие эволюцию системы, на правильном пути?». Кроме этого, важным является необходимость учета стратегических, организационных, и других аспектов бизнеса, влияющих на эволюцию унаследованной ИС.

Поэтому основной целью создания каркаса (enterprise framework) следует считать необходимость оценки среды, в рамках которой осуществляется эволюция унаследованной ИС, необходимость обеспечения понимания широкого спектра вопросов, как технического, так и управленческого характера, которые соотносятся с эволюцией унаследованных ИС. Важнейшим здесь становится выявление факторов, определяющих успех деятельности по эволюции ИС, а так же разработка согласованного множества технических и управленческих инструкций (действий), требуемых для эффективного планирования, оценки и управления инициативами по эволюции систем.

Отражая, применительно к эволюции системы, пространство проблем и пространство решений, предлагаемый в [13] каркас включает такие элементы, как организация, проект, унаследованная система, инженерия систем и программных средств, технологии, целевая система. Между этими элементами определяется связь. При этом для каждого из них приводится список вопросов (checklists), позволяющий охарактеризовать (исследовать и оценить) интересующий элемент.

Следует признать, что положенная в основу каркаса концепция является расширением концепции разработки ИС с учетом деятельности по внедрению, повседневных операций и деятельности по сопровождению. Каркас может быть использован для выполнения следующих видов деятельности:

- исследование и анализ пространства проблем и пространства решений в контексте инициатив по эволюции системы;
- разработка руководств по составлению стратегических и тактических планов по реинжинирингу унаследованных систем;
- выявление технологических вопросов и потенциальных проблем на протяжении всего пути по эволюции систем;
- рецензирование планов, ранжирование (приоритезация) технических вопросов, разработка рекомендаций по улучшениям процессов эволюции систем и результатов их выполнения (рабочих продуктов).

Общий подход к использованию каркаса представлен на Рис. 3.

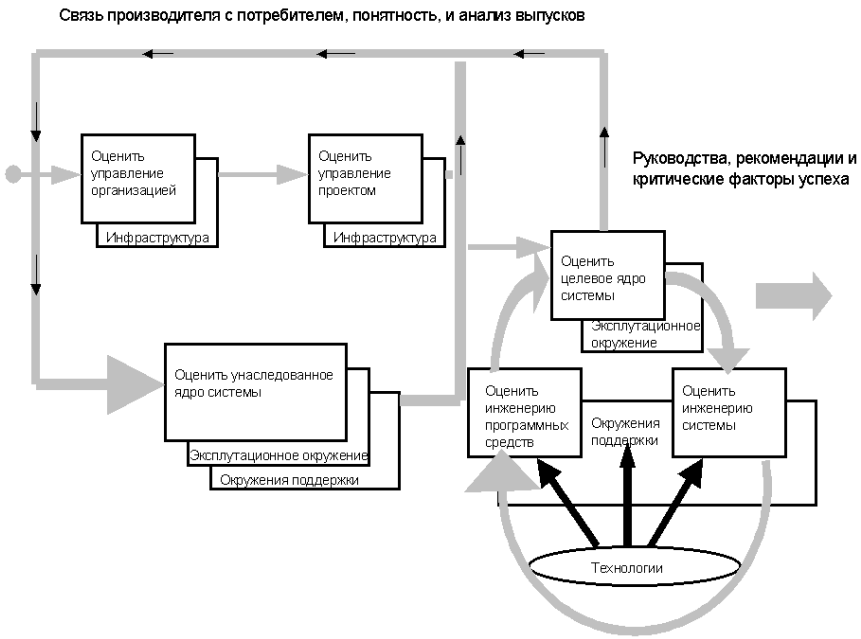

Рис. 3. *Общий подход к использованию каркаса.*

В рамках подхода выделяются две значительные составляющие: основная фаза и фаза эволюции. Основная фаза сфокусирована на таких элементах, как организация, проект и унаследованная система. Фаза эволюции концентрируется на целевой системе, инженерии систем и программных средств, технологиях, используемых для построения целевой системы. Другими словами, первая фаза сфокусирована на пространстве проблем, а вторая на пространстве решений.

В [10] авторами предлагается комплексный, основанный на рассмотренном ранее каркасе, подход к эволюции систем, который определяется в контексте унаследованных систем и современных программных технологий.

В основу подхода положены следующие положения (принципы):

- различие между эволюцией систем и сопровождением программных средств;
- использование описанного ранее каркаса (enterprise framework) при поддержке принятия решений в процессе эволюции системы;
- достижение технического понимания систем на высоком уровне абстракции;
- применение технологий распределенных объектов, технологии «wrapping» [10] для эволюции системы;
- применение «net-centric» [10] вычислений для эволюции системы.

Определяя различие между сопровождением программных средств и эволюцией систем в части реинжиниринга ИС, авторы рассматривают сопровождение как «мелкозернистую», «краткосрочную» деятельность, направленную на планирование и внесение локализованных изменений. При сопровождении структура (архитектура) системы остается относительно неизменной, а требуемая «порция» вносимых за определенный промежуток изменений, как правило, связана с изменением какого-либо одного требования к системе. Такие изменения, обычно, не сопровождаются существенным изменением значений характеристик и атрибутов качества программных средств.

В отличие от сопровождения, эволюция систем рассматривается как «крупнозернистая», «высокоуровневая», форма изменений на уровне структуры (архитектуры) системы. Вносимые в процессе эволюции изменения, приводят к изменениям значений атрибутов качества, что, как правило, существенно упрощает сопровождение систем. Для этого при внесении изменений в структуру системы могут использоваться стратегии «снизу вверх» и «сверху вниз», применение которых основано на ревизии программного кода, восстановлении описания архитектуры унаследованной системы, проектировании новой структуры, новой формы документации и т.д.

Авторами подчеркивается, что эволюция позволяет адаптировать систему сразу с учетом большого количества выдвигаемых к ней требований (включая приобретение новых возможностей), что в конечном итоге увеличивает стратегическую и экономическую значимость программных средств. При этом акцент смещается от понимания программ к пониманию систем, от сопровождения к эволюции и миграции, от стратегий «снизу вверх» к стратегиям «сверху вниз».

В отличие от рассмотренных ранее работ, в [15] основное внимание уделяется исследованию и решению технических проблем, связанных с миграцией унаследованных систем. Характеризуя понятие «миграция ИС» в данной работе выделяются отдельно эволюция, сопровождение и миграция ИС. Основываясь на том факте, что объектом эволюции и сопровождения является унаследованная ИС, а объектом миграции как унаследованная, так и новая (целевая) системы, авторы разделяют миграцию и другие процессы ЖЦ ИС. При этом миграция рассматривается как деятельность, которая начинается с унаследованной ИС и заканчивается сопоставимой целевой ИС.

В основу предлагаемого авторами подхода положена декомпозиция структуры системы на компоненты пользовательского интерфейса, компоненты – приложения и компоненты управления базами данных. При этом в качестве основных инкрементально выполняемых шагов миграции выступают:

- анализ унаследованной ИС;
- декомпозиция структуры унаследованной ИС;
- проектирование интерфейсов (пользовательских и системных) целевой системы;

- проектирование приложений (функциональной логики) целевой системы;
- проектирование базы данных целевой системы;
- развертывание среды, требуемой для эксплуатации и сопровождения целевой системы;
- создание и развертывание необходимых шлюзов;
- миграция унаследованных данных;
- миграция унаследованных приложений (компонентов, реализующих функциональную логику);
- миграция унаследованных интерфейсов (пользовательских и системных);
- переход к использованию целевой системы.

3. Классификация подходов, методов и технологий.

В настоящий момент существует значительное количество работ, посвященных проблемам, методам и технологиям реинжиниринга ИС. Эти работы охватывают данную проблематику с различных точек зрения, рассматривая и исследуя в различной степени как проблемы концептуального уровня (иногда даже философского характера), так и конкретные методы, и инструментальные средства, предназначенные для реинжиниринга ИС.

Несмотря на наличие множества различных решений, их исследование и комплексное применение на практике бывает затруднено. Причинами возникающих трудностей следует считать:

- понятие «реинжиниринг ИС» различными исследователями до сих пор трактуется по разному, существует множество близких понятий, наличие которых приводит к появлению внешне отличающихся, но по сути схожих подходов, методов и технологий;
- предлагаемые решения не позиционируются в контексте других существующих решений;
- решения не интегрированы на уровне методологий и технологий, большое количество методов и инструментальных средств направлено на решение отдельных локальных задач, связанных с реинжинирингом ИС;
- наблюдается разрыв между решениями концептуального характера и решениями, направленными на решение конкретных прикладных задач.

Следует признать, что на множестве существующих решений (подходов, методов и технологий) отсутствует их систематизация, обеспечивающая позиционирование и определяющая взаимосвязь решений на различных уровнях рассмотрения задач реинжиниринга, в том числе, уровнях методологии и инструментальных средств.

В сложившейся ситуации целесообразным видится определение классификации, определяющей структуризацию на множестве существующих подходов, методов и технологий реинжиниринга ИС.

Так, классифицируя существующие подходы, методы и технологии, можно выделить следующие уровни рассмотрения и исследования аспектов, соотносимых с деятельностью по реинжинирингу ИС (см. Рис. 4).

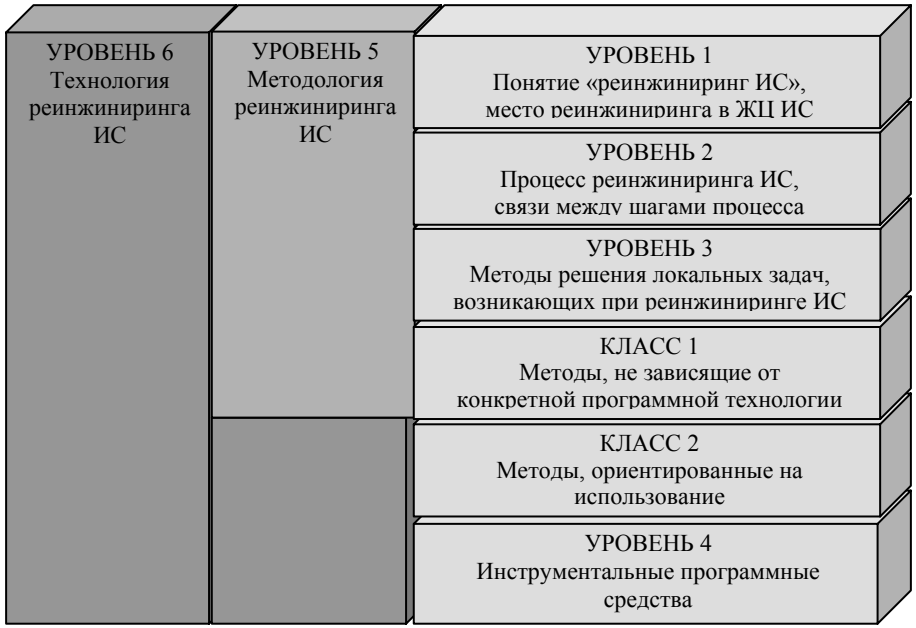

Рис. 4. *Уровни рассмотрения и исследования аспектов, соотносимых с реинжинирингом ИС.*

Первый уровень включает исследования, направленные на достижение концептуального понимания деятельности по реинжинирингу ИС. Именно на этом уровне исследуются вопросы адекватного определения понятия «реинжиниринг ИС», определения места реинжиниринга в жизненном цикле (ЖЦ) ИС, в том числе выявление связей процесса реинжиниринга ИС в целом с другими процессами ЖЦ ИС. Следует считать, что большая часть рассмотренных ранее аспектов, относятся к этому уровню.

В отличие от первого, *второй уровень* содержит исследования, основная цель которых заключается в выявлении основных шагов (действий), реализуемых в процессе реинжиниринга и в определении связей между основными шагами процесса. Здесь в сферу рассмотрения попадают потоки управления и потоки данных между основными шагами процесса, основные роли, соотносимые с исполнителями процесса, а так же правила распределения ролей среди команды

исполнителей. Исследования и разработки на этом уровне проводятся как без учета, так и с учетом вводимых ограничений (например, архитектурных решений, которым должны соответствовать подлежащие реинжинирингу ИС). Следует признать, что наиболее детально вопросы, связанные с данной проблематикой, исследуются в [1, 15, 33], в меньшей степени в [8, 13, 16, 20].

Так, в [1] выделяются основные фазы процесса реинжиниринга ИС, а для каждой фазы определяются действия (деятельности) и соотносимые с ними потоки управления. Процесс дается в самом общем виде и не зависит от каких-либо ограничений, например используемых программных технологий. В [15] авторами так же определяется выполняемый в рамках процесса реинжиниринга поток работ. Однако здесь основное внимание уделяется вопросам технического характера, а выполняемые при реинжиниринге шаги предусматривают декомпозицию структур, соответственно, унаследованной и целевой системы на компоненты пользовательского интерфейса, компоненты – приложения и компоненты управления базами данных. В отличие от предыдущих работ в [33] авторами основной акцент сделан на решение задач оценки унаследованных систем и поддержки принятия решений при реинжиниринге ИС. Авторами определяется процесс оценки, состоящий из следующих входящих в его состав процессов (подпроцессов):

- технической оценки (Reengineering Technical Assessment),
- экономической оценки (Reengineering Economic Assessment),
- принятия управленческих решений (Reengineering Management Decision).

Для каждого из процессов описывается его область применения, поток входящих в него работ (шагов процесса), определяются связи с другими процессами оценки. Следует признать, что потенциально эти процессы могут быть интегрированы в любой процесс разработки. Однако стоит заметить, что оценка унаследованных систем является лишь одной из задач по реинжинирингу ИС.

Еще одним примером процесса, направленного на решение локальной задачи, является итеративный процесс, определяющий следующую последовательность шагов, которые должны быть выполнены при планировании проектов реинжиниринга ИС [37]:

- Определение целей, направлений деятельности организации и информационной системы.
- Формирование объединенной команды, которая будет осуществлять реинжиниринг унаследованной системы.
- Определение среды разработки и сопровождения, базирующейся на применении CMM (Capability Maturity Model).
- Выбор стандартного множества метрик для оценки программных средств.

- Анализ унаследованной системы.
- Определение процесса реализации деятельности по реинжинирингу.
- Разработка/Обновление стандартных средств тестирования и валидации.
- Анализ средств реинжиниринга.
- Обучение.

На третьем уровне рассматриваются (исследуются и разрабатываются) методы, каждый из которых направлен на решение некоторой локальной задачи, возникающей в процессе реинжиниринга ИС, например, выполнения определенного шага процесса. По сути, эти методы воплощают собой некоторые вполне конкретные решения, с которыми соотносится определенная область применения. Как правило, в проектах по реинжинирингу применятся некоторая комбинация таких методов, при этом каждый из них может стать частью методологии реинжиниринга ИС, но в отдельности таковой не является. Более того, объединение этих методов так же нельзя рассматривать в качестве методологии, поскольку между ними не определены связи, обеспечивающие их интегральную целостность. Другими словами отсутствуют системообразующие факторы, делающие набор методов целостным образованием – системой.

С некоторой условностью все методы реинжиниринга ИС можно разделить на два класса.

Методы, относящиеся к первому классу, определены на концептуальном уровне и, в целом, не зависят от какой-то одной программной технологии. Так, в [2] дается обзор методов модернизации и миграции унаследованных систем. Среди всего множества рассматриваемых методов к первому классу относятся метод репликации баз данных и основанный на объектно-ориентированном подходе метод построения оболочек для компонентов унаследованной системы (object – oriented wrapping), методы «белого» и «черного» ящика модернизации системы. Другими представителями данного класса являются: методы оценки вариантов реинжиниринга ИС [9, 11, 33], метод планирования миграции программных средств [8], методы извлечения знаний о существующей системе [3, 4, 17], методы трансформации (реконструкции) архитектуры ИС [3, 7, 19, 21], методы автоматизации реинжиниринга программ [6, 23] и т.д. В целом следует считать, что область применения таких методов, как правило, характеризуется некоторым классом программных технологий. А для применения в реальных проектах каждый из них адаптируется с учетом используемых в проекте технологий и инструментальных средств.

Отдельным направлением исследований, относящимся к данному классу и получившим развитие в последние годы, является исследование и разработка образцов реинжиниринга ИС [26-30, 36]. Каждый из образцов реинжиниринга ИС нацелен на решение некоторой типовой задачи (проблемы), которая

сопровождает деятельность по реинжинирингу ИС. Не смотря на некоторые отличия, авторы в целом придерживаются единого подхода к описанию таких образцов. Так, в [26] определяется следующий шаблон описания.

- Имя образца.
- Цели применения.
- Область приложения.
- Мотивация к применению.
- Структура системы до и после применения образца.
- Процесс применения образца.
- Обсуждение образца.
- Особенности, зависящие от языка программирования.

Стоит заметить, что хотя последний из разделов шаблона «привязывает» шаблон к определенной среде реализации, языкам программирования, все же каждый из образцов представляет некоторое концептуальное решение проблемы. Подробную информацию об образцах реинжиниринга можно найти в книге «Object-Oriented Reengineering Patterns» [27].

В отличие от первого класса, методы второго изначально ориентированы на использование определенных программных технологий. Ко второму классу относятся так же адаптации методов из первого класса. Здесь методы в наибольшей степени приспособлены к их непосредственному (прямому) применению в конкретных проектах. Примерами методов данного класса следует считать [2]: методы интеграции с использованием CGI, методы интеграции на основе технологии XML, метод построения оболочек для компонентов унаследованной системы с использованием технологии CORBA.

И наконец, *четвертый уровень* включает исследование и разработку инструментальных программных средств, автоматизирующих применение подходов, методов и технологий, рассматриваемых на предыдущих уровнях. Следует признать, что в настоящий момент таких средств существует большое количество, среди которого выделяются следующие [3, 4, 6, 23, 31, 32, 35]:

- средства переноса приложений, написанных на устаревших языках, на современные языки и платформы (например, с языков PL/1, Кобол на языки C++, Java, Visual Basic);
- средства интеграции унаследованных приложений, к примеру, на основе метода построения оболочек для компонентов унаследованной системы;
- средства автоматизированного извлечения данных из унаследованных систем;
- средства автоматизированного извлечения знаний об унаследованных системах;
- средства оптимизации (реструктуризации) унаследованных систем при их переносе на современные языки и платформы (как на уровне программного кода, так и на уровне архитектуры ИС);

- различные средства анализа программного кода.

Полезная классификация инструментальных средств дается в [31]. В рамках этой классификации выделяются такие типы средств, как:

- средства реинжиниринга бизнес процессов;
- средства преобразования имен данных;
- средства реинжиниринга данных (БД);
- средства прямого инжиниринга;
- средства преобразования форматов;
- средства реструктуризации;
- средства обратного проектирования;
- средства трансляции исходного кода;
- и др.

В этой же работе авторами приводятся характеристики инструментальных средств реинжиниринга ИС. Для каждого из них специфицируется имя программного продукта; требования к аппаратной и программной платформе; координаты компаний-поставщиков; поддерживаемые языки программирования, СУБД, платформы; принадлежность к тому или иному типу средств.

Осуществляя классификацию и исследование существующих подходов, методов и технологий реинжиниринга ИС, дополнительно к уже выделенным уровням, следует добавить еще два, являющихся по своей природе интегральными. Это уровень методологии и уровень технологии реинжиниринга ИС. Первый из них обеспечивает целостное рассмотрение и применение подходов и методов без учета среды их реализации, специфики конкретных проектов. Это соответствует интеграции первых трех уровней, причем на третьем уровне в сферу рассмотрения, в первую очередь, попадают методы первого класса. Уровень технологии обеспечивает адаптацию (конкретизацию) методологии реинжиниринга ИС с учетом среды реализации, специфики конкретных проектов посредством применения методов и инструментальных средств, соответствующих третьему и четвертому уровням. При этом в отличие от методологии, на третьем уровне, прежде всего, рассматриваются методы второго класса.

Следует признать, в процессе проводимых авторами настоящей статьи исследований не было обнаружено целостных методологий и технологий реинжиниринга ИС, соответствующих уровню проработки и области охвата RUP [24, 25]. В наибольшей степени эту проблема исследуется в [1, 8, 15, 33]. В тоже время, в [1] предлагается лишь каркас, определяющий основной ход работ, в [8] даются рекомендации по выполнению основных видов деятельности по реинжинирингу. В [33] определяются лишь процессы оценки унаследованных систем, а в [15] основное внимание уделяется вопросам технического характера, при этом основной акцент сделан на решение проблем интеграции систем, в то

время как методы анализа унаследованной системы практически не рассматриваются.

Представленный подход к классификации обеспечивает систематизацию на множестве существующих подходов, методов и технологий реинжиниринга ИС. В качестве его основных областей применения следует рассматривать:

- оценку состояния в области методологического и технологического обеспечения реинжиниринга ИС;
- адаптацию и разработку методологий и технологий реинжиниринга ИС.

В завершении данного раздела стоит отметить, что возможны и другие полезные подходы к систематизации методов и технологий реинжиниринга ИС. Так, предлагаемые и исследуемые различными авторами процессы реинжиниринга, как правило, охватывают систему в целом, в то время как методы могут соотноситься с некоторыми ее составляющими, с отдельными шагами процесса. Последний факт обуславливает возможность классификация многих методов:

- по объектам применения (например, в зависимости от их типа (компонент пользовательского интерфейса, компонент данных, вычисляющий компонент (компонент бизнес - логики)));
- по видам деятельности процесса реинжиниринга (методы извлечения знаний о существующих ИС (методы обратного проектирования), методы оценки (анализа) существующих ИС и т.д.).

4. Заключение.

На основании проведенных исследований и вводимой классификации подходов, методов и технологий реинжиниринга ИС, становится возможным охарактеризовать текущее состояние в области его методологического и технологического обеспечения.

Так, несмотря на наличие большого количества работ, охватывающих данную проблематику с различных точек зрения, следует признать, что:

- понятие «реинжиниринг ИС» различными исследователями до сих пор трактуется по разному, существует множество близких понятий, наличие которых приводит к появлению внешне отличающихся, но по сути схожих подходов, методов и технологий;
- существующие методы и технологии не позиционируются в контексте других существующих решений, не интегрированы на уровне методологий и технологий;
- наблюдается разрыв между решениями концептуального характера и решениями, направленными на решение конкретных прикладных задач;
- отсутствует четкая взаимосвязь между методами/технологиями реинжиниринга и методологиями разработки ИС «с нуля».

В сложившейся ситуации актуальной задачами становятся:

- унификация, а возможно, и стандартизация понятия «реинжиниринг ИС», других связанных с ним понятий;
- разработка целостных методологий реинжиниринга ИС;
- разработка средств адаптации методологий реинжиниринга ИС в реальных проектах;
- создание инструментальных средств, обеспечивающих комплексное решение задач по реинжинирингу ИС;
- интеграция методологий разработки «с нуля» и методологий реинжиниринга, в том числе и на уровне поддерживающих их инструментальных средств.

Литература

1. *John Bergey, William Hefley, Walter Lamia, Dennis Smith* A Reengineering Process Framework, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1995.
2. *Santiago Comella-Dorda, Kurt Wallnau, Robert C. Seacord, John Robert* A Survey of Legacy System Modernization Approaches, Software Engineering Institute (Technical Note CMU/SEI-200-TN-003, 00tn003.pdf), Pittsburgh, 2000.
3. *Rick Kazman, S. Jeromy Carriere* Playing Detective: Reconstructing Software Architecture from Available Evidence. Technical Report CMU/SEI-97-TR-010. Pittsburgh, 1997.
4. *Rick Kazman, S. Jeromy Carriere* View Extraction and View Fusion in Architectural Understanding, Proceedings of the Fifth International Conference on Software Reuse (ICSR), June, 1998, Victoria, BC.
5. *Кротов А.А. и Лупян Е.А.* Обзор методов реструктуризации и интеграции информационных систем, http://d902.iki.rssi.ru/students/alekro/Dissertation/Papers/Reengineering/my_review.html
6. "Автоматизированный реинжиниринг программ" Сборник статей под ред. А.Н.Терехова и А.А.Терехова Издательство С.-Петербургского университета, 2000.
7. *Guo G. Y., Atlee J. M., Kazman R.* A Software Architecture Reconstruction Method, Department of Computer Science, University of Waterloo, Software Engineering Institute, Carnegie Mellon University.
8. *John Bergey, Dennis Smith, Nelson Weiderman* DoD Legacy System Migration Guidelines, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, September 1999.
9. *John Bergey, Dennis Smith, Nelson Weiderman, Steven Woods* Options Analysis for Reengineering (OAR): Issues and Conceptual Approach, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, September 1999.
10. *Weiderman N., Nelson H., Bergey John K., Smith Denis B., & Tilley Scott R.* Approaches to Legacy System Evolution, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1997 (CMU/SEI-97-TR-014)
11. *Ransom J., Sommerville I., & Warren I.* A Method for Assessing Legacy Systems for Evolution, Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering (CSMR98), 1998
12. *Bisdal Jesus, Lawless Deirdre, Wu Bing, Grimson Jane, Wade Vincent, Richardson Ray, & O'Sullivan D.* An Overview of Legacy Information System Migration, APSEC 97, ICSC 97, 1997

13. *John K. Bergey, Linda M. Northrop, Dennis B. Smith* Enterprise Framework for the Disciplined Evolution of Legacy Systems, SEI CMU October 1997.
14. *Энн Мак-Крори* Что такое унаследованные системы?, Computerworld, США, 1998.
15. *Michael L. Brodie, Michael Stonebraker* Migrating Legacy Systems. Gateways, Interfaces & The Incremental Approach, Morgan Kaufmann Publishers, Inc., 1995
16. *Weiderman N., Northrop L., Smith D., Tilley S., Wallnau K.* Implications of Distributed Object Technology for Reengineering, CMU/SEI-97-TR-005, SEI CMU June 1997.
17. *Tilley S.* A Reverse-Engineering Environment Framework, SEI CMU April 1998
18. *Bergey J., Smith D., Tilley S., Weiderman N., Woods S.* Why Reengineering Projects Fail, SEI CMU April 1999.
19. *Kazman R., O'Brein L., Verhoef Ch.* Architecture Reconstruction Cuidelines, SEI CMU August 2001.
20. *Kazman R., Woods S., Carriere S.* Requirements for Integrating Software Architecture and Reengineering Models: CORUM II, SEI CMU, October 1998.
21. *Carriere S.J., Woods S. Kazman R.* Software Architectural Transformation, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, October 1999.
22. *John Bergey, Dennis Smith, Nelson Weiderman* DoD Software Migration Planning, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, August 2001.
23. www.ispras.ru
24. *Kruchten P.* The Rational Unified Process: an introduction. Reading: Addison Wesley, 1999.
25. Rational Unified Process, version 2002.05.00.25, Rational Software Corporation.
26. *Ducasse S., Richner T, Nebbe R.* Type-Check Elimination: Two Object-Oriented Reengineering Patterns, Proceedings WCRE, 1999.
27. *Demeyer S., Ducasse S. , Nierstrasz O.* Object-Oriented Reengineering Patterns, Morgan Kaufmann Publishers, Inc., 2002.
28. *Ducasse S. , Demeyer S., Nierstrasz O.* A Pattern Language for Reverse Engineering, Proceedings of EuroPLoP, 2000.
29. *Pooley R., Stivens P.* Software Reengineering Patterns, University of Edinburg, Department of computer science, http://www.reengineering.ed.ac.uk/.
30. *Dewar R.* Characteristics of Legacy System Reengineering, The University of Edinburgh, Division of Informatics, http://www.reengineering.ed.ac.uk/.
31. *Olsem M. R., Sittenauer Ch.* Reengineering, Software Technology Support Center, Technology Report, Volume 2, April 1995, http://www.stsc.hill.af.mil/reng/.
32. *Ducasse S., Lanza M, Tichelaar S.* The Moose Reengineering Environment, University of Berne, Software Composition Group, 2001.
33. Software Reengineering Assessment Handbook, Technical Report, Version 3.0, 1997, http://www.stsc.hill.af.mil/
34. *Sander T.* Modeling Object-Oriented Software for Reverse Engineering and Refactoring, Thesis, University of Bern, 2001.
35. *Ducasse S.* R´etro-Conception d'Application `a Objets Reengineering Object-Oriented Applications, Universit´e Pierre et Marie Curie, 2001.
36. The FAMOOS Object-Oriented Reengineering Handbook, http://www.iam.unibe.ch/_famoos/handbook/.
37. *Olsem M. R., Sittenauer Ch.* Reengineering, Software Technology Support Center, Technology Report, Volume 1, April 1995, http://www.stsc.hill.af.mil/reng/.
38. IEEE Computer Society TCSE, 1990, http://tcse.org/.
39. Стандарт ANSI/IEEE Std. 729-1983.

40. Joint Logistic Commanders Computer Resources Management group (JLC/CRM), 1992, http://www.stsc.hill.af.mil/reng/.

Приложение А. Глоссарий понятий и терминов.

Реинжиниринг бизнес-процессов (Business Process Reengineering)

Фундаментальное переосмысление и радикальное перепроектирование бизнес-процессов для достижения существенных улучшений в таких ключевых для современного бизнеса показателях результативности, как затраты, качество, уровень обслуживания и оперативность [40].

Рационализация имен данных (Data name rationalization (DNR))

Унификация именования данных, являющаяся специальным случаем реинжиниринга данных. Заключается во введении унифицирующих соглашений по именованию во всех программных системах [40].

Реинжиниринг данных (Data Reengineering)

Выполнение всех функций реинжиниринга, соотносимых с исходным кодом, но применительно к файлам данных [40].

Восстановление результатов проектирования (Design recovery)

Подмножество обратного инжиниринга, в котором знание проблемной области, внешняя информация, логические выводы или нечеткие суждения принимаются во внимание при обследовании исходной системы. Целью восстановления результатов проектирования является выявление значимых высокоуровневых абстракций, в дополнение к тем, которые были получены в процессе непосредственного исследования (изучения) системы [38].

Прямой инжиниринг (Forward engineering)

Традиционный процесс перехода от высокоуровневых абстракций и логического, независящего от реализации проектирования к физической реализации системы [38].

Представляет собой переход от требований к высокоуровневому проектированию, и далее к низкоуровневому проектированию и реализации [36].

Представляет собой множество видов деятельности по инжинирингу системы, на вход которым для производства новой целевой системы поступают продукты и артефакты, производные от унаследованных программных средств, и новые требования [40].

Комментарий

Основное отличие прямого инжиниринга от реинжиниринга заключается в том, что в случае реинжиниринга «стартуют» от существующей реализации (существующей системы).

Основное отличие прямого инжиниринга от инжиниринга в целом заключается в том, что на вход прямого инжиниринга поступают результаты реинжиниринга [40].

Редокументирование (Redocumentation)

Форма реструктуризации, где результирующее семантически эквивалентное представление системы является альтернативным взглядом, предназначенным для его восприятия человеком [38].

Процесс анализа системы с целью создания различного рода сопровождающей ее документации. Включает в себя как создание руководств пользователя, так и переформатирование листинга исходного кода [40].

Реинжиниринг (Reengineering)

Исследование (изучение, обследование) и перестройка исходной системы с целью ее воссоздания в новой форме с последующей реализацией этой новой формы [38].

Комментарий

Процесс реинжиниринга включает в себя такие подпроцессы, как обратный инжиниринг, реструктуризация, редокументирование, прямой инжиниринг и переориентация [40]. Как правило, предполагает модернизацию системы в целях обеспечения ее соответствия возникающим новым требования [36].

Рефакторинг (Refactoring)

Специальный вид реструктуризации, а именно реструктуризации на уровне программного кода, имеющей объектно-ориентированный контекст. Является процессом изменения программной системы, направленным на улучшение внутренней структуры программного кода, но не изменяющим внешнего поведения программы [34, 36].

Реструктуризация (Restructuring)

Трансформация системы из одной формы представления в другую на одном и том же уровне абстракции. Новое представление сохраняет семантику и внешнее поведение (функциональность) оригинала [38, 40].

Переориентация (Retargeting)

Процесс трансформации и перевода (переноса) существующей системы в новую конфигурацию [40].

Комментарий

Например, перенос на новую аппаратную платформу, под новую операционную систему, под новое CASE - средство.

Обратный инжиниринг (обратное проектирование) (Reverse engineering)

Процесс анализа исходной системы, преследующий 2 цели – выявить компоненты системы и отношения между ними, и создать представление системы в другой форме или на более высоком уровне абстракции [38].

Процесс достижения понимания системы, ее анализа и абстрагирования по направлению к новой форме представления, соответствующей более высокому уровню абстракции [40].

Процесс извлечения информации из существующей программной системы [36]. В общем случае обратное проектирование применяют для извлечения информации на высоком уровне абстракции, например, извлечение информации уровня проектирования из программного кода.

Сопровождение программных продуктов (Software maintenance)

Модификация программного продукта после его поставки в целях исправления ошибок, улучшения производительности и других атрибутов качества, или адаптации продукта к изменениям окружения (внешней среды) [39].

Трансляция исходного кода (Source Code Translation)

Трансляция исходного кода с одного языка программирования на другой или с одной версии в другую на том же самом языке программирования [40].

Инжиниринг систем (Systems Engineering)

Высокоуровневый процесс инжинирии систем, направленный на достижение соответствия системы всем выдвигаемым к ней требованиям [40].