второй допустимый порог.

Алгоритм работы дефектоскопа следующий. Блок обработки задает номер канала и формирует сигнал сброса для буферной памяти FIFO. Сигнал сброса служит одновременно и сигналом запуска измерений. Сигнал запуска обеспечивает старт формирования тестового сигнала в выбранном канале генератора. После передачи в образец тестового сигнала микропроцессор генераторного блока формирует сигнал старта для блока оцифровки. Этот сигнал устанавливает RS-триггер в блоке приемника, который разрешает тактирование запуска АЦП и синхронную запись результатов преобразования в буферную память FIFO. После заполнения буферная память формирует сигнал сброса RS-триггера, запрещая дальнейшее тактирование АЦП и FIFO. Этот же сигнал активизирует прием блоком обработки данных из буферной памяти. После усиления принятого сигнала (эхо или донного) производится первичная фильтрация посредством корреляции опорного сигнала в виде одного периода частоты зондирующего импульса и принятого пакета, имеющего ту же частоту заполнения. При этом учитывается, что частота ультразвука может изменяться оператором. Полученная реализация возводится в квадрат и сглаживается. Затем выполняется повторная корреляционная обработка с использованием в качестве опорного сигнала функции вида "синус в квадрате". При появлении импульсных помех значительной амплитуды дополнительно производится накопление заданного количества импульсов от различных зондирований объекта контроля. Экспериментально установлено, что принятый алгоритм обработки позволил повысить чувствительность прибора примерно в 10 раз.

Выводы. Показана высокая эффективность применения методов обработки информации в приборах неразрушающего контроля. Разработан новый ЭМА дефектоскоп, чувствительность которого повышена в 10 раз.

Список литературы: **1**. *Патон Б.Є., Троїцький В.О., Посипайко Ю.М.* Неруйнівний контроль в Україні // Інформ. бюл. Українського товариства неруйнівного контролю та технічної діагностики. – 2003. – № 2 (18). – С. 5–9. **2**. *Сучков Г.М.* О главном преимуществе ЭМА способа // Дефектоскопия. – 2000. – № 10. – С. 67–70. **3**. *Ермолов И.Н.* Теория и практика ультразвукового контроля. − М: Машиностроение, 1981. – 240 с. **4**. *Сучков Г.М.* Новые методы ультразвукового контроля ЭМА способом на основе адаптации радиолокационных технологий // Техническая диагностика и неразрушающий контроль. – К., 2005. − № 3. − С. 38−42. **5**. *Сучков Г.М.* Обработка информации. Повышение возможностей корреляционного анализа в ЭМА приборах // Контроль. Диагностика. – М., 2004. − № 12. − С. 13−16. **6**. *Сучков Г.М.* Возможности линейной частотной фильтрации в ЭМА приборе // Контроль. Диагностика. – М., 2004. − № 10. − С. 20−21. **7**. *Сучков Г.М.* Обработка информации. Возможности корреляционного анализа при толщинометрии ЭМА способом // Контроль. Диагностика. – М., 2002. − № 8. − С. 37−40. **8**. *Сучков Г.М.* Возможности современных ЭМА-толщиномеров // Дефектоскопия. – 2004. – № 12. – С. 16–25. **9**. *Сучков Г.М.* Современные возможности ЭМА дефектоскопии // Дефектоскопия. – 2005. – № 12. – С. 24 - 39. **10**. Неразрушающий контроль: Справочник: В 7 т. Под общ. ред. *В.В. Клюева*. Т. 3: Ультразвуковой контроль / *И.Н. Ермолов, Ю.В. Ланге*. – М.: Машиностроение, 2004. – 864 с.

Поступила в редакцию 10.11.2006

УДК 651.326

Н.В. ТКАЧУК, д-р техн. наук, НТУ "ХПИ",
А.А. ЗЕМЛЯНОЙ, НТУ "ХПИ",
В.А. ЧУГАЙ, НТУ "ХПИ"

ПРИМЕНЕНИЕ МОДЕЛИ COCOMO II В ЗАДАЧАХ УПРАВЛЕНИЯ СИСТЕМНЫМИ ТРЕБОВАНИЯМИ ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Розглянуто сучасні системи управління вимогами та їх типову функціональність. На основі обзору інформаційних джерел запропоновано розширення типової функціональності таких систем можливістю знаходження конфігурацій вимог, що задовольняють ряду обмежень. Даний підхід реалізується шляхом використання моделі COCOMO II та шаблонів специфікацій на базі стандартів IEEE .

The proposed paper includes a review of the modern requirements management systems. Based on the information sources review the extension of the system's typical functionality is proposed. The approach introduces a possibility of finding configurations of system requirements within defined constraints. Its implementation is based on the cost model COCOMO II and typical requirements specification extracted from IEEE standards.

Постановка проблемы. Разработка программного обеспечения (ПО), рассматриваемая как система производственных и управленческих процессов, в настоящее время достаточно подробно освещена и стандартизирована во множестве официальных и корпоративных документов таких организаций как ISO (International Organization for Standardization), IEEE (Institute of Electrical and Electronics Engineers), SEI (Software Engineering Institute), а также таких ведущих производителей ПО как Microsoft, Agile, IBM и ее подразделения Rational. Так, например, фактический стандарт в области программной инженерии SWEBOK (Software Engineering Body of Knowledge) определяет следующие основные составляющие процесса создания ПО: *разработка требований* (Software Requirements), *проектирование ПО* (Software Design), *реализация ПО* (Software Construction), *тестирование* (Software Testing), *сопровождение и поддержка* (Software Maintenance) [1].

Первый этап, разработка требований к ПО, во многом определяет успех последующих, зависимых от него этапов [2]. Очевидно, что проектирование и реализация ПО проводится на основе разработанных спецификаций системных требований (СТ), а тестирование должно гарантировать соответствие произведенного продукта СТ, предъявляемым к нему со стороны заказчика. Ошибки, допущенные на этапе разработки требований, дают мультипликативный эффект на последующих этапах. От степени точности СТ зависит и основная характеристика разрабатываемого ПО – его качество.

Управление СТ связано и с другими процессами, непосредственно связанными и неотъемлемыми от производства ПО – процессы управления проектами, управления конфигурациями, управления системой качества производства.

Анализ литературы. Развитие индустрии программного обеспечения привело к появлению ряда официальных и фактических стандартов в этой области. Их разработчиками являются такие организации как IEEE, ISO, SEI и другие. Предметом таких стандартов выступают составляющие процесса создания ПО, управления программными проектами, разрабатываемые программные продукты и их качество.

Один их фундаментальных программных стандартов ISO является ISO 12207 Software Life-Cycle Processes [3], который определяет структуру процесса создания ПО, структуру жизненного цикла ПО. Организацией IEEE было разработано руководство по промышленному внедрению ISO 12207 под названием IEEE/EIA 12207.1 – Industry Implementation of International Standard ISO/IEC 12207 [4]. ISO/IEC 90003:2004 Software Standard [5] является руководством по применению системы качества ISO 9001:2000 в сфере производства программного обеспечения.

На основе ряда стандартов для программной индустрии организацией IEEE была создана работа под названием SWEBOK – Software Engineering Body of Knowledge [6]. Для SWEBOK также были использованы адаптированные IEEE стандарты ISO. Руководство SWEBOK задает границы дисциплины программной инженерии и определяет 10 ассоциированных с ним областей знаний, в том числе и разработку требований. Эта область знаний определяет понятие программного требования, основные типы требования, различия между программными и системными требованиями. Рассматривается процесс разработки требований вместе с такими этапами как извлечение требований, анализ, специфицирование, утверждение.

Извлечение требований связано с определением источников, из которых требования могут быть получены, а также методов и технологий, при помощи которых это будет происходить. Процесс анализа СТ включает определение и разрешение конфликтов между требованиями, определение области применения ПО и внешней по отношению к нему среды, разграничение системных и программных требований и дальнейшую детальную проработку последних. Проводится классификация требований, концептуальное моделирование, проектирование архитектуры и привязка требований к спроектированным программным артефактам.

Следующий шаг – специфицирование требований. Спецификация должна не только представлять перечень требований в структурированном виде, но и предоставлять возможность его регулярного изменения и согласования. Существуют рекомендуемые структуры спецификации, закрепленные в международных и национальных стандартах.

Целью данной статьи является анализ современных стандартов и технологий для управления СТ, синтез одного их возможных вариантов эталонной архитектуры для *систем управления требованиями* (СУТ) и разработка подхода к решению задачи расширения типовой функциональности

таких систем за счет возможности поиска конфигураций требований, удовлетворяющих ряду определенных ограничений.

Современные технологии и инструментальные средства управления требованиями. Приведенные выше составляющие процесса управления требованиями являются предметом автоматизации. В настоящее время на рынке существует ряд инструментов такого рода от известных производителей. Общее их название – система управления требованиями (СУТ, Requirements Management System – RMS). Общее число доступных в продаже инструментов составляет более десятка: Teamcenter SLATE, Borland CaliberRM, Requirement Tracing System, Prosareq Requirements Manager, RaQuest, Requirements Mgmt Database, Requisite Pro, Telelogic Doors, SoftREQ и другие. Среди них можно выделить явных лидеров, системы, наиболее богатые с функциональной точки зрения. Это Borland CaliberRM [7], Requisite Pro [8], Telelogic Doors [9]. Borland CaliberRM является корпоративной системой управления требованиями, что предполагает взаимодействие различных групп пользователей. В Borland CaliberRM предусмотрено центральное хранилище данных с проектными требованиями. Доступ к центральному хранилищу может производиться как при помощи локально установленного у пользователя GUI приложения, так и при помощи веб-интерфейса. Основное предназначение системы – ускорение процесса определения требований и повышение его эффективности. Поддерживается ряд функций для анализа требований и управления процессом их разработки. Имеется возможность интеграции с другими инструментами для разработки ПО – инструментами управления проектами, проектирования, реализации, тестирования. Проектные требования представляются в виде древовидной иерархической структуры. Сами требования хранятся в виде текстово-графической информации без какой-либо формализации, есть возможность задавать метаинформацию в виде набора атрибутов. Помимо иерархической структуры и трассировки, к сожалению, нет достаточно гибкого способа задания отношений между требованиями. Имеется возможность создавать шаблоны спецификаций для их быстрого повторного применения в других проектах.

Функциональность инструмента IBM RequisitPro во многом схожа с описанной выше базовой функциональностью CaliberRM. Также организуется центральное хранилище с возможностью удаленного и локального доступа при помощи GUI и веб-интерфейсов, реализовано интегрирование с внешними инструментами разработки. Имеются возможности структурирования, задания и прослеживания связей между требованиями. Требования представляются в виде текстово-графической информации, есть возможность задания атрибутов. Одним из преимуществ RequisitPro является возможность визуально определять схожие требования в рамках одного или нескольких проектов и применять готовые апробированные решения в новом проекте. Имеются возможности создания требований при помощи стандартных или создаваемых

текстовых шаблонов. В готовом виде предусмотрены шаблоны для выпуска документации в соответствии со стандартами IEEE, ISO, SEI CMM и Rational Objectory Process. Так же, как и две рассмотренных ранее системы, Telelogic DOORS реализует хранилище требований и предоставляет интерфейс доступа к нему. В отличие от CaliberRM и RequisitPro здесь нет возможности подключения при использовании веб-интерфейса, имеется только GUI. Основная ставка делается на создание документов требований в формате текста, дополнительных встраиваемых элементов (рисунки, диаграммы) и определение атрибутов объектов. Особенностью Telelogic DOORS является то, что одну и ту же информацию можно просматривать с различных точек зрения, для этого инструмент предоставляет концепцию видов. Кроме того, возможность связывания объектов при помощи направленных ссылок дает возможность создавать не только иерархические структуры требований, но и структуры сетевого типа. Имеются средства анализа связей. Поддерживается возможность обмена документами с внешними приложениями.

Результаты изучения можно обобщить в виде схемы типичной функциональности системы управления требованиями. Она включает в себя два вида функций – инструментальные и аналитические. Составляющие каждого из видов представлены на рисунке.

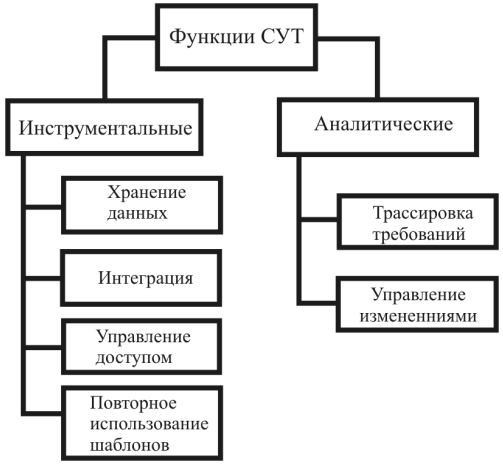

Рис. Типичная функциональность СУТ

Несмотря на то, что типовые функции СУТ достаточно развиты, наблюдается явный недостаток функций аналитического типа, которые бы позволили более тесно интегрировать процесс управления требованиями и другие процессы создания ПО. Это позволило бы эффективно решать ряд задач, применяя спецификации требований. К таким задачам, например,

относится поиск допустимой конфигурации требований при ограничениях бюджета на реализацию проекта.

Расширение типовой функциональности СУТ. В настоящее время фирмы-разработчики ПО для эффективной реализации проектов создания информационных систем отдают предпочтение использованию современных методов расчета трудозатрат и стоимости ПО. Одной из ключевых проблем при осуществлении данных расчетов является четкая спецификация СТ и адекватное соответствие спецификации входным параметрам расчетных моделей.

COCOMO II (Constructive Cost Model II) – это наиболее полная, конструктивная и широко применяемая модель оценивания трудозатрат. Модель ориентирована на порционность поступления информации для оценивания на протяжении всего периода разработки ПС и является трехуровневой [10]:

1. Предварительная модель (Application Composition Model).
2. Предпроектная модель ((Early Design Model).
3. Детальная модель (Post Architecture Model).

В общем случае формула для вычисления алгоритмической оценки стоимости (трудозатраты человек/месяц) записывается следующим образом:

$$PM = A * size^{B} * M \,, \qquad (1)$$

где

$$M = PERS * RCPX * RUSE * PDIF * PREX * FCIL * SCED \,, \qquad (2)$$

$A = 2{,}45$ – постоянный коэффициент, который зависит от организации, выполняющей проект, и типа разрабатываемого программного обеспечения; $size$ – может соотноситься либо с размером кода программы, либо с функциональной оценкой, выраженной в количестве объектных или функциональных точек; показатель степени B может варьироваться в пределах от 1 до 1,5, он отображает объем работ, требующийся для реализации больших проектов; множитель M является произведением семи показателей, характеризующих проект и процесс создания ПО, а именно: *RCPX* (product reliability and complexity) – надежность и уровень сложности разрабатываемой системы; *RUSE* (required reuses) – повторное использование компонентов; *PDIF* (platform difficulty) – сложность платформы разработки; *PERS* (personnel capability) – возможности персонала; *PREX* (personnel experience) – опыт персонала; *SCED* (scheduling constraints) – график работ и *FCIL* (support facilities) – средства поддержки.

На уровне детальной модели формируется 17 параметров из 7 параметров предпроектной модели. Проанализировав все параметры модели COCOMO II, можно условно разделить их на две группы: организационно-технологические и параметры СТ. Ко второй группе принадлежат параметры, показанные в таблице.

Таблица

Предпроектная модель	Детальная модель
Сложность платформы разработки (*PDIF*).	Показатели, ограничивающие время исполнения (*TIME*). Возможность изменения платформы разработки (*PVOL*). Ограничение объема памяти (*STOR*).
Надежность и уровень сложности разрабатываемой системы (*RCPX*).	Требуемая надежность системы (*RELY*). Сложность системных модулей (*CPLX*). Объем необходимой документации (*DOCU*). Размер используемой базы данных (*DATA*).

На основе стандарта IEEE Std 830-1993 [11], можно сформировать универсальный шаблон *SRS* (Software Requirements Specification). Разделы требований, которые описаны в данном шаблоне, находят отображение в параметрах СТ предпроектной и детальной модели COCOMO II.

Исходя из того, что бюджет проекта также может рассматриваться как одно из нефункциональных макро-требований заказчика системы, то компания-разработчик должна выполнить проект с максимальной функциональностью в рамках установленного бюджета. В простейшем случае такую задачу можно решить, используя аппарат линейного математического программирования [12]. Определим целевую функцию:

$$f = \sum_{i=1}^{n} a_i x_i \longrightarrow \max, \qquad (3)$$

где a_i – коэффициент важности реализации *i*-го СТ (задается экспертом); x_i – степень реализации *i*-го СТ, $x \in [0;1]$, $i = \overline{1,n}$.

Определим ограничения:

$$\sum_{i=1}^{n} c_i x_i \le B, \quad i = \overline{1,n}, \qquad (4)$$

где c_i – стоимость полной реализации *i*-го СТ; B – бюджет проекта (стоимость разработки ПО).

$$x_i \ge q_i, \ \overline{i = 1,n}, \qquad (5)$$

где q_i – нижняя граница степени реализации *i*-го СТ; n – количество СТ.

Бюджет проекта в данной постановке задачи рассчитывается по модели COCOMO II и сравнивается с бюджетом заказчика. Стоимость полной

реализации *i*-го СТ рассчитывается на основании общего бюджета проекта, определенного моделью COCOMO II. Количество системных требований определяется экспертом по шаблону *SRS*. Решением поставленной задачи является вектор $\bar{x}(x_1,...,x_n)$ – степень реализации функциональной полноты каждого СТ.

Приведем расчетный пример, иллюстрирующий данный подход. Подставим в целевую функцию 7 детальных СТ универсального шаблона *SRS*. Определим коэффициенты важности для них:

1. Внешние интерфейсы – степень важности максимальна, так как основной целью является логически правильная организация экранной формы относительно технического задания на разработку ($\alpha_1 = 10$).

2. Функции системы – степень важности данного требования максимальна, так как генерация отчетов и параметризированный поиск являются неотъемлемыми компонентами ИС ($\alpha_2 = 10$).

3. Требования исполнения – достаточно минимальны, так как задачи быстродействия не ставились ($\alpha_3 = 2$).

4. Требования логики БД – приближены к максимальному значению, но возможно на начальном этапе разработки ИС сокращение и упрощение модели данных ($\alpha_4 = 8$).

5. Надежность – должна быть приближена к максимальной оценке ($\alpha_5 = 9$).

6. Безопасность – определена на номинальном уровне. Однако необходимо предусмотреть паролирование доступа стандартными средствами СУБД ($\alpha_6 = 5$).

7. Ремонтопригодность – требование минимально, которое может проявляться в изменениях кода запросов, формата составления отчетов, изменении пароля системы ($\alpha_7 = 1$).

Тогда целевая функция примет вид:

$$f = 10x_1 + 10x_2 + 2x_3 + 8x_4 + 9x_5 + 5x_6 + x_7 \longrightarrow \max \qquad (6)$$

Сформируем вид ограничений для данной задачи на основе универсального шаблона *SRS* и расчета трудозатрат на реализацию исследуемых параметров модели COCOMO II: внешние интерфейсы (*TIME*=1, *PVOL*=1, *STOR*=1), функции (*TIME*=1, *PVOL*=1, *STOR*=1, *RELY*=1, *CPLX*=1, *DOCU*=1, *DATA*=1), требования времени исполнения (TIME=1), требования логики БД (*RELY*=1, *DATA*=1), надежность (*RELY*=1), безопасность (*RELY*=1, *CPLX*=1), ремонтопригодность (*DOCU*=1).

Так же установим нижнюю границу степени реализации *i*-го СТ: $q_i \ge 0$.

Ограничения будут иметь вид:

$$3x_1 + 7x_2 + x_3 + 2x_4 + x_5 + 2x_6 + x_7 \le 15; \quad (7)$$

$$0 \le x_i \le 1, \ i = \overline{1,7}. \quad (8)$$

Решая эту задачу одним из известных методов, например, методом искусственного базиса [12], находим вектор, который показывает степень реализации функциональной полноты каждого СТ в рамках бюджета заказчика $\bar{x}(1; \frac{6}{7}; 1; 1; 1; 1; 0)$. Найденный вектор значений неизвестной переменной представляет собой конфигурацию СТ, удовлетворяющих заданным ограничениям. Можно выделить ряд факторов, которые оказывают влияние на полученный результат: опыт экспертов, оценка недоработки проекта и его функциональности, правильное и полное сопоставление параметров модели COCOMO II шаблону SRS, содержание объема работ для каждого СТ относительно полученного вектора результатов реализации функциональной полноты СТ. Данный подход дает возможность управлять СТ в рамках бюджета разрабатываемого проекта – нефункционального макро-требования заказчика.

Выводы. В работе поставлена и решена проблема расширения типовой функциональности СУТ возможностью поиска конфигураций требований с учетом ограничений. Научная новизна предложенного подхода состоит в совместном использовании модели COCOMO II и шаблонной спецификации на основе стандартных документов IEEE, а также представленной математической модели. Практическая значимость подхода состоит в повышении продуктивности аналитиков и менеджеров программных проектов в процессах разработки и управления системными требованиями. К недостаткам работы можно отнести то, что не предлагается способов автоматизации перехода от детальных спецификаций требований к параметрам модели COCOMO II. Для этого необходима особая формализация представлений требований, что является перспективой дальнейших исследований.

Список литературы: 1. *Соммервил И.* Инженерия программного обеспечения. – М.: Вильямс, 2002. – 624 с. **2.** *Бабенко Л.П., Лаврищева К.М.* Основи програмної інженерії. – К.: Знання, 2001. – 269 с. **3.** ISO 12207 Software Life-Cycle Processes. – 1995. – 57 с. **4.** IEEE/EIA 12207.1 – Industry Implementation of International Standard ISO/IEC 12207. – 1997. – 36 с. **5.** ISO/IEC 90003:2004 Software Standard**. –** 2004. – 175 p. **6.** IEEE Software Engineering Body of Knowledge Guide. 2004 – 200 с. **7.** Borland CaliberRM, www.borland.com/de/products/caliber. **8.** IBM Requisite Pro, www3.ibm.com/software/awdtools/reqpro. **9.** Telelogic Doors, www.telelogic.com. **10.** *Андон Ф.И., Коваль Г.И., Коротун Т.М.* Основы инженерии качества программных систем. – К.: Академпериодика, 2002. – 502 с. **11.** ISO 830-1998 Recommended Practice for Software Requirements Specifications. 1998. – 37 p. **12.** *Зайченко Ю.П.* Исследование операций. – К.: Вища шк., 1988. – 320 с.

Поступила в редакцию 10.10.2006

УДК 681.3

В.А. ШЕХОВЦОВ, канд. техн. наук, НТУ "ХПИ",
А.В. КОСТАНЯН, НТУ "ХПИ"

ОБРАБОТКА НЕФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ НА СТАДИЯХ АНАЛИЗА И ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

У статті розглянуто підхід до аналізу нефункціональних вимог на перших стадіях розробки програмного забезпечення. Підхід базується на моделі KCPM, розширюючи її для обробки нефункціональних вимог до системи. Ці вимоги трактуються як сквозні інтереси. Представлені розширення моделі KCPM та нові правила відображення при використанні асиметричного AOSD підходу. Розглянуто також питання з використання методів NLP у даному підході.

This paper presents approach for non-functional requirements processing on early development stages. It's based on the Klagenfurt Conceptual Predesign Model (KCPM) allowing taking into account the non-functional requirements to the system. These requirements are treated as crosscutting concerns. The extensions of the KCPM schema and the new mapping rules are introduced for the case of asymmetric AOSD approach. Also questions about NLP support for our approach are considered.

Постановка проблемы. В процессе разработки программного обеспечения (ПО) всё более актуальной становится проблема обработки нефункциональных требований наряду с основными задачами, для решения которых данное ПО предназначается. Это обусловлено увеличивающейся сложностью создаваемых систем, временными ограничениями и повышением требований к самим программным продуктам, связанным со скоростью изменения окружающего мира. Под нефункциональными требованиями понимаются такие понятия как производительность, надёжность, эффективность, применимость, портируемость, тестируемость, воспринимаемость и модифицируемость программных систем. Проблема определения и обработки таких требований очень непроста. В работах [1, 2] были попытки описать некоторые подходы решения данного вопроса, но единой концепции не было. Положение изменилось с появлением АОРПО (Аспектно-Ориентированной Разработки Программного Обеспечения), которая выделяет нефункциональные требования как аспекты или сквозные интересы.

В данной статье мы опишем подход для выделения и конвертирования нефункциональных особенностей на этапе анализа требований в промежуточную модель, с последующим построением на её основе концептуальной модели системы. В качестве промежуточной модели мы используем расширенную KCPM (Klagenfurt Conceptual Predesign Model).

Анализ литературы. Опишем коротко основные положения и принципы используемых нами концепций. Таковыми являются АОРПО и KCPM. Дадим вначале определение АОРПО. Основной целью АОРПО является реализация средств систематической идентификации, разделения, представления и