OSTIS-2013
(Open Semantic Technologies for Intelligent Systems)

УДК 004.823

ПОДХОД К АВТОМАТИЗАЦИИ АНАЛИЗА И СИНТЕЗА ТЕХНИЧЕСКОЙ ДОКУМЕНТАЦИИ

Заболеева-Зотова А.В.[*], Орлова Ю.А.[*], Петровский А.Б.[**]

[*]*Волгоградский государственный технический университет, г. Волгоград, Российская Федерация*

zabzot@gmail.com

yulia.orlova@gmail.com

[**]*Институт системного анализа РАН, г. Москва, Российская Федерация*

pab@isa.ru

В работе рассматривается подход к автоматизации начальных этапов проектирования программного обеспечения. Разработан программный инструментарий, предназначенный для автоматического семантического анализа технической документации, автоматического построения моделей, синтеза структуры и естественно-языкового описания программного обеспечения.
Ключевые слова: проектирование; семантический анализ; синтез структуры; техническая документация.

Введение

Проектирование программного обеспечения (ПО) представляет собой плохо формализованный трудоемкий процесс, требующий от пользователя глубокого знания предметной области и навыков в проектировании. Наиболее известные из коммерческих продуктов, используемых для проектирования программного обеспечения, предназначены, в основном, для визуализации промежуточных и конечных результатов процесса проектирования. Некоторые из них позволяют полностью автоматизировать последние этапы проектирования: генерация кода, создание отчетной и сопровождающей документации и т.д. В тоже время задача анализа технического задания, выделение на его основе функциональной структуры программы, задача синтеза функциональной структуры программы на основе построенной модели цели практически не решаются. Ни одно из известных CASE-средств не позволяет пользователю проектировать программное обеспечение даже на ограниченном естественном языке [Заболеева-Зотова и др., 2010b].

Повышение эффективности проектирования многокомпонентного программного обеспечения может быть обеспечено за счет автоматизации начальных этапов процесса. Для этого необходимо:

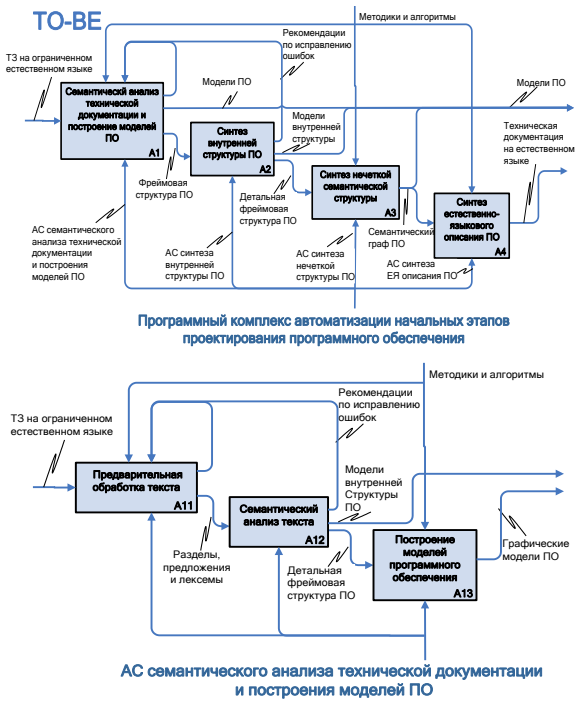

Рисунок 1. Процесс проектирования программного обеспечения

– провести анализ процесса концептуального проектирования многокомпонентного ПО, моделей семантического анализа текста и формализмов представления знаний;

– формализовать процедуры автоматизации начальных этапов проектирования многокомпонентного программного обеспечения;

– разработать алгоритмы семантического анализа технической документации и синтеза структуры многокомпонентного программного обеспечения, используя аппарат мультимножеств;

– реализовать разработанные формализмы, методику и алгоритмы в виде автоматизированной системы естественно-языкового описания ПО;

– провести апробацию разработанных средств при проектировании многокомпонентного ПО.

На рисунке 1 приведена диаграмма «to be», которая служит иллюстрацией процесса проектирования программного обеспечения с использованием предлагаемой концепции.

1. Анализ существующих CASE-систем

В современных информационных технологиях важное место отводится инструментальным средствам, системам разработки и сопровождения ПО. Эти технологии и среды образуют CASE-системы. Широко известные CASE-системы, такие как BPWin, ERWin, OOWin, Design/IDEF, CASE-Аналитик, Silverrun, Rational Rose, Vantage Team Builder, S-Designor и другие, позволяют частично автоматизировать процесс проектирования программного обеспечения. Однако, как показал анализ, данные системы автоматизируют конечные этапы проектирования программного обеспечения, такие как создание отчетной и сопровождающей документации, генерация кода и т.д. Начальные этапы проектирования – формирование и анализ технической документации, синтез внутренней структуры программного обеспечения – выполняются аналитиком.

Важное значение в процессе разработки ПО имеют также средства спецификации проектов ПО. На начальных этапах проектирования ПО строится функциональная модель системы, которая описывает совокупность выполняемых системой функций. Построение функциональных спецификаций выполняется аналитиком на основе текста технической документации и является в настоящий момент не автоматизированной процедурой.

Как показали специальные исследования, 41% обнаруженных ошибок в сложном проекте программного обеспечения составляют ошибки, сделанные на начальных этапах проектирования. Причиной 56% всех ошибок являются ошибки, сделанные на стадии анализа требований [Заболеева-Зотова и др., 2010а]. Такие ошибки чрезвычайно дорого исправлять.

Проведенный анализ позволяет сделать следующие выводы:

- Автоматизированы в основном отдельные этапы проектирования.
- Отсутствуют методы автоматизированного анализа технической документации.
- Отсутствует автоматизация процесса построения моделей программного обеспечения.
- На начальных этапах проектирования программного обеспечения допускается много ошибок.

Таким образом, автоматизация начальных этапов проектирования программного обеспечения остается открытой задачей, требующей для своего решения создания специальных программных средств.

2. Методика автоматизированного анализа и синтеза технической документации

Часть компонентов технической документации содержит информацию, которая по своему характеру является нечеткой, что обусловлено вариативностью и подвижностью границ языковой нормы и статистическим характером отдельных видов информации. Неточность информации, содержащейся в компонентах технической документации, относится к семантическому и предметно-зависимому уровню и обусловлена сложностью процесса формализации описываемых явлений. Рекомендации по проведению такой формализации формулируются на ограниченном естественном языке (ЕЯ) и могут трактоваться по-разному различными специалистами в зависимости от языковой интуиции человека,

Практически непреодолимой причиной неполноты лингвистической информации является открытость и постоянное развитие ЕЯ: появление новых языковых единиц, изменение свойств существующих единиц и правил их сочетаемости. Такая динамика особенно заметна в подъязыках новых предметных областей с неустоявшейся терминологией.

Еще одной причиной неполноты лингвистической информации является наличие нюансов и языковых особенностей отдельных носителей языка, описать и формализовать которые на сегодняшний день не представляется возможным.

Часть информации, содержащейся в технической документации, особенно в техническом задании, может быть ошибочной. Ошибочная информация отличается от неточной тем, что для неточной информации известно, насколько она может не соответствовать действительности. Ошибочная информация может быть маркирована даже как точная, но в то же время полностью противоречить реальной ситуации.

Предлагаемая методика анализа и синтеза технической документации (рис. 2) содержит формализмы, необходимые для представления семантики требований к программному обеспечению на ранних этапах проектирования.

В соответствии с методикой разрабатываемая программная система рассматривается как черный ящик, а предъявляемые к ней требования представляются в виде спецификации функций и определения потоков входных и выходных воздействий.

Рисунок 2. Методика анализа и синтеза технической документации

Методика анализа и синтеза текста технической документации состоит из четырех этапов: семантическая обработка текста, создание фреймовой структуры, создание моделей программного обеспечения, описанного в технической документации, синтез текста описания построенных моделей.

Для семантической обработки текста разработана семантическая модель текста технической документации, включающая требования, сформулированные в виде документов на ограниченном естественном языке. Фреймовая структура является внутренним представлением требований. Модель программного обеспечения задается в виде описания требований на графических языках Data Flow Diagrams и UML.

Семантическая модель текста технической документации содержит разработанные расширенные нечеткие атрибутные грамматики над фреймовой структурой формальных документов «Техническое задание» и «Технический проект».

Расширенная нечеткая атрибутная грамматика, необходимая для автоматизированного анализа текста технического задания, определена в виде:

$$AG=(N, T, P, S_0, B, F, A, D(A))$$

где N – конечное множество нетерминальных символов; T – множество терминальных символов, непересекающееся с множеством N; P – конечное множество правил; S_0 – выделенный символ из множества N, называемый начальным символом; B – множество лингвистических переменных $\beta_{k,i}$, соответствующих терминальным символам T (переменная i на k уровне); F – множество функций принадлежности $f_{k,i}$, определяющих степень принадлежности $\mu_{k,i}$ лингвистических переменных $\beta_{k,i}$; $A=Asin \cup Asem$ – множество атрибутов, где $Asin$ – синтаксические атрибуты, $Asem$ – семантические атрибуты; $D(A)$ – конечное множество семантических действий.

Лингвистические переменные из множества $B=\{\beta_{k,i}\}_{k,i}$ используемые для анализа текста технического задания описывается следующей пятеркой:

$$\beta_{k,i} = (\beta, T(\beta), U, G, M)$$

где β – название лингвистической переменной (наименование и область применения, основание для разработки, назначение разработки, технические требования к программному изделию, стадии и этапы разработки и т.д.); $T(\beta)$ – языковые выражения. Для лингвистических переменных верхнего уровня они являются лингвистическими переменными, соответствующими терминалам правой части правила. Для лингвистических переменных нижнего уровня – нечеткими переменными, то есть выражениями естественного языка. U – универсум, $T(\beta) \subset U$. G – правила морфологического и синтаксического описания языковых выражений, которые определяют атрибуты $Asin$. M – семантическое правило для лингвистических переменных, которое индуцируется морфологическими и синтаксическими правилами, так как смысл терма в T частично определяется его синтаксическим деревом, и семантическими атрибутами $Asem$.

Методы представления связей между правилами транслируются на язык нечеткой математики. При этом связи представляются нечеткими отношениями, предикатами и правилами, а последовательность преобразований этих отношений – как процесс нечеткого вывода.

Лингвистические переменные верхнего уровня являются составными, то есть включают лингвистические переменные нижнего уровня. Благодаря этому можно построить дерево лингвистических переменных и установить зависимость между ними.

Функции принадлежности из множества $F=\{f_{k,i}\}_{k,i}$ лингвистических переменных $\{\beta_{k,i}\}_{k,i}$ необходимы для построения нечеткого вывода. В частности, каждому правилу грамматики из множества P ставится в соответствие функция принадлежности $f_{k,i}$. Эта двойственная система подстановок используется для вычисления смысла лингвистической переменной.

В грамматике используются следующие синтаксические атрибуты $Asin$: ’Название’ – текст, представляющий собой наименование раздела; ’Содержимое’ – текст, представляющий собой содержимое раздела; ’Клауза’ – фрагмент; ’Клауза ТИРЕ’ – фрагмент с тире; ’Группа ГЕНИТ_ИГ’ – именительная группа, связанная родительным падежом и др.

Семантические атрибуты, используемые в грамматике, содержат название атрибута $Asem$ и

следующие семантические действия $D(A)$: “Фрейм СИСТЕМА=Создание” – создается фрейм СИСТЕМА; ”Слот НАЗВАНИЕ СИСТЕМЫ= Присваивание” – значение присваивается слоту НАЗВАНИЕ СИСТЕМЫ; “Фрейм ПОТОК ДАННЫХ=Создание” – создается фрейм ПОТОК ДАННЫХ и др.

3. Информационная модель

Рассмотрим систему, описанную в техническом задании, как совокупность входных воздействий, функций и выходных воздействий:

$$S = (N_S, F_S, \boldsymbol{I}_S, \boldsymbol{O}_S),$$

где N_S – название системы, F_S – множество функций системы, $\boldsymbol{I}_S$ – мультимножество входных воздействий, $\boldsymbol{O}_S$ – мультимножество выходных воздействий. Входные воздействия такой системы S представляют собой мультимножество

$$\boldsymbol{I}_S = \{k_I^1 \cdot I_S^1, \ldots, k_I^n \cdot I_S^n, \ldots, k_I^N \cdot I_S^N\},$$

где функция кратности k_I^n характеризует число одинаковых входов вида I_S^n.

Выходные воздействия такой системы S представляют собой мультимножество

$$\boldsymbol{O_S} = \{k_O^1 \cdot O_S^1, \ldots, k_O^m \cdot O_S^m, \ldots, k_O^M \cdot O_S^M\},$$

где функция кратности k_O^m характеризует число одинаковых выходов вида O_S^m.

Множество функций системы F_S реализуются подсистемами $F_S^1, \ldots, F_S^l, \ldots, F_S^L$, каждая из которых описывается следующим образом:

$$F_S^l = (N_F^l, \boldsymbol{I}_F^l, D_F^l, G_F^l, H_F^l, \boldsymbol{O}_F^l),$$

где N_F^l – имя функции F_S^l, $\boldsymbol{I}_F^l$ – мультимножество входных воздействий функции F_S, D_F^l – название действия, выполняемого функцией, G_F^l – объект, над которым выполняется действие, H_F^l – ограничения на функцию, $\boldsymbol{O}_F^l$ – мультимножество выходных воздействий функции F_S.

Мультимножества $\boldsymbol{I_S}$, $\boldsymbol{O}_S$, $\boldsymbol{I}_F^l$, $\boldsymbol{O}_F^l$ представляют потоки данных, которые состоят из единичных порций, имеющих одинаковую структуру:

$$DF = (N_{DF}, D_{DF}, T_{DF}),$$

где N_{DF} – название потока данных, D_{DF} – направление потока данных, T_{DF} –тип данных в потоке.

Операции над мультимножествами [Петровский, 2000], [Петровский, 2003], [Петровский, 2004] можно использовать синтеза структуры программных системам.

При синтезе текста описания построенных моделей программного обеспечения требования к программной системе представляются в виде набора диаграмм потоков данных на графическом языке Data Flow Diagrams. В диаграммах отображается общая структура системы с указанием ее входных, выходных потоков и функции, выполняемые системой, с их входными, выходными потоками.

Работа частично поддержана Российским фондом фундаментальных исследований (проекты 11-07-00398, 12-07-00266, 12-07-00270).

Библиографический список

[Заболеева-Зотова и др., 2010a] Автоматизация семантического анализа текста технического задания: монография. / Заболеева-Зотова А.В., Орлова Ю.А. - Волгоград, ИУНЛ. 2010. - 155 с.

[Заболеева-Зотова и др., 2010b] Автоматизация начальных этапов проектирования программного обеспечения / Заболеева-Зотова А.В., Орлова Ю.А. // Изв. ВолгГТУ. Серия «Актуальные проблемы управления, вычислительной техники и информатики в технических системах»: межвуз. сб. науч. ст. - Волгоград, ВолгГТУ. 2010. - Вып. 8, № 6. - С. 121-124.

[Петровский, 2000] Петровский А.Б. Комбинаторика мультимножеств. // Доклады Академии наук. 2000, Т.370, №6, С.750-753.

[Петровский, 2003] Петровский А.Б. Пространства множеств и мультимножеств. – М.: Едиториал УРСС, 2003.

[Петровский, 2004] Петровский А.Б. Многокритериальное принятие решений по противоречивым данным: подход теории мультимножеств. // Информационные технологии и вычислительные системы. 2004, №2, С.56-66.

[Розалиев и др., 2010] Применение нейронных сетей и грануляции при построении автоматизированной системы определения эмоциональных реакций человека / Розалиев В.Л., Бобков А.С.,. Федоров О.С. // Изв. ВолгГТУ. Серия «Актуальные проблемы управления, вычислительной техники и информатики в технических системах»: межвуз. сб. науч. ст. - Волгоград, ВолгГТУ. 2010. - Вып. 9, № 11. - С. 63-68.

APPROCH TO AUTOMATION OF ANALYSIS AND SYNTHESIS OF TECHNICAL DOCUMENTATION

Zaboleeva-Zotova A.V.[*], Orlova Y.A.[*], Petrovsky A.B.[**]

[*]*Volgograd State Technical University, Volgograd, Russian Federation*
zabzot@gmail.com
yulia.orlova@gmail.com

[**]*Institute for Systems Analysis, Russian Academy of Sciences, Moscow, Russian Federation*
pab@isa.ru

The paper considers an approach to automation of the initial stages of software design. Program tools, which provide automated semantic analysis of technical documentation, automated construction of models, synthesis of structure and natural language description of the program software, are developed.

The method of analysis and synthesis of text and technical documentation consists of four phases: semantic word processing, creating frame structure, the creation of software models, as described in the technical documentation, the synthesis of the description text of the constructed models.

This work was partially supported by the Russian Foundation for Basic Research (projects 11-07-00398, 12-07-00266, 12-07-00270).