УДК 519.683+519.688+519.71+519.1

ПРОГРАММНЫЙ КОМПЛЕКС, ПРЕДНАЗНАЧЕННЫЙ ДЛЯ ЛОГИКО-СМЫСЛОВОГО АНАЛИЗА СЛОЖНЫХ ПРОГРАММНЫХ СИСТЕМ

Абасов Н. В. – канд. техн .наук, ведущий науч. сотрудник, e-mail: cell@sifibr.irk.ru (Институт систем энергетики им. Л. А. Мелентьева СО РАН); *Чернышов М. Ю.* – канд. физ. наук, зав. научно-методической частью, e-mail: Michael_Yu_Chernyshov@mail.ru (Президиум Иркутского научного центра СО РАН)

Обсуждается проблема, связанная с отсутствием технологий эффективного анализа программных систем. Описывается программный комплекс, разработанный на основе предложенной авторами технологии вычислительного моделирования и предназначенный для автоматического анализа программ путем статического анализа кода текста объектной программы без её выполнения. Глубина анализа может варьировать от определения поведения операторов, отдельных функций – до анализа всего исходного кода. Результаты анализа используются для оперативного выявления ошибок в объектной программе и проверки её соответствия спецификации. Реализована оригинальная технология, предполагающая автоматическое выявление функциональных связей в текстах программ и построение многообразия их модельных графических представлений. Она также допускает возможности изменения проанализированной граф-структуры и дополнения её дополнительными кластерами.

The problem bound up with the absence of a technology intended for efficient analysis of software systems is discussed. Described is a software complex developed on the basis of the technology of computational modeling proposed by the authors and intended for solving the problems of automatic analysis of software systems by static analysis of the object software text code without execution of the program. The analysis depth may vary from defining the behavior of separate software operators, separate functions – to analysis of the total initial code. Results of analysis are used for time-optimal finding the errors in the object program and verification of its correspondence to the specification. A unique technology, which presumes automatic finding out functional relations in software texts and constructing a diversity of their representations, has been implemented. It also presumes the possibilities of changing the graph-structure analyzed and its complementation with additional clusters.

Ключевые слова: надёжные вычислительные системы; вычислительная технология эффективного анализа программ; статический анализ и модификация кода; графы связей; технология автоматического построения графов связей; метод вычислительного моделирования.

Введение и постановка задач исследования

В рамках одного из проектов РАН была поставлена задача создания гибких аналитических программных комплексов (АПК) высокой производительности, которые могли бы быть использованы как в практическом анализе программ и программных систем, так и в обучении будущих системных аналитиков. Эта задача решалась бы относительно просто, если бы создаваемые компьютерные программы строились на неких единых и универсальных принципах. Это могли бы быть, например, единые принципы модельной логико-смысловой организации программ или, иначе говоря, принципы технологии программирующего моделирования. При таком подходе необходимо договориться о том, чтобы все программисты мира строили только прозрачные программные системы, такие, в которых каждая из подпрограмм строилась бы как органичный и легко вычленимый логико-смысловой элемент в единой логико-смысловой модели целостной программной системы. Однако в условиях капитализма, предполагающего заинтересованность программиста и компании, которую он представляет, в коммерческом результате своей работы, такой подход недостижим. Строя современные программные системы, программисты стремятся сделать их непрозрачными, чтобы максимально усложнить их анализ и не дать другим программистам возможность воспользоваться результатами их труда.

Именно в таких условиях пришлось решать задачу создания гибкого АПК, т.е., по существу, задачу создания супер-хакерской аналитической программы, которая позволила бы выполнять автоматизированный анализ почти любых программ и программных систем, написанных на языках, для которых существует понятие функция. Совершенно ясно, что создание такого АПК невозможно (1) без создания технологии глубокого семантического анализа объектных программных систем (ОПС), точнее, без технологии смыслового анализа (а) семантических функций ОПС и (б) предназначения ОПС согласно спецификации, т.е. без знания принципов смыслового анализа, и (2) без разработки вычислительной технологии смыслового анализа программных систем, которая и должна быть основой такого АПК.

Предполагалось, что созданный АПК должен стать инструментом практического анализа программ и, кроме того, эффективным средством обучения будущих программистов и квалифицированных пользователей.

Об этапах реализации нашего проекта

Ранее была разработана технология автоматического анализа программ и программных систем [1]. В ходе её воплощения был последовательно решен ряд задач. На первом этапе эта технология нашла воплощение в форме прак-

тически работающего АПК первого этапа (АПК-1) [2], явившего собой попытку реализовать принцип программирующего моделирования и технологию логико-смысловой организации программ, когда строится прозрачная программная система, и каждая из её подпрограмм строится как легко вычленимый элемент в единой логико-смысловой модели программной системы. АПК-1 уже можно было применять в анализе доступных программ. Инструментальными основами АПК-1 стали (1) разработанная авторами универсальная инструментальная среда программирования ЗИРУС [3], (2) язык ОЛФИС [3] и (3) подход, предполагающий представление программ в виде моделей данных и набора отношений [4]. Этот подход основан на общих принципах теории баз данных, применении теории графов для модельного отображения объектных программных систем (или их модулей). Он был рассчитан на формирование многообразных внешних текстовых и графических отображений программ.

На этой основе была предпринята попытка решить задачу второго этапа, создав АПК-2, назначение которого – обеспечение возможности повторного использования модулей старых программ в создаваемых программных продуктах. В ходе воплощения идеи, связанной с моделированием, предполагавшим построение графов функциональных связей внутри анализируемой ОПС по её исходному коду, была разработана аналитическая система, предназначенная для автоматизированного логико-смыслового анализа обширного класса модулей программ и программных систем в целях изучения возможности их повторного использования. АПК-2 пригоден для таких операций, как оперативное выявление мест в ОПС, содержащих ошибки, идентификация свойств ОПС, проверка соответствия спецификации.

Позднее, уже на основе АПК-2, был разработан аналитический программный вычислительный комплекс АПК-3, предназначенный для исследования и модификации программных систем с учётом итогов их логико-смыслового анализа, который понимается весьма широко, предполагая возможность смыслового анализа не только текстов программ. В рамках АПК-3 реализован универсальный метод, предполагающий семантическую интерпретацию текстов анализируемых компьютерных программ в терминах смысловых отношений [5]. АПК-3 обладает новыми возможностями, главной из которых является способность модифицировать (изменять) объектные программы в соответствии с возникшими потребностями. Такая модификация подразумевает, в частности, возможности изменения граф-структуры проанализированной программы и дополнения её новыми кластерами, специально созданными аналитиком. Применение предустановленных шаблонов различных представлений графов, а также дополнение графов новыми кластерами осуществляется во время работы анализатора.

Таким образом, подход, предполагавший практическое поэтапное воплощение технологии вычислительного (программирующего) моделирования в форме программных комплексов АПК-1 – АПК-3 оправдал себя. Опыт показал, что на вышеуказанных основах (в том числе на принципе, предполага-

ющем смысловой анализ, многозначное моделирующее отображение модулей объектных программ, и связанном с построением графов отношений) можно построить эффективный АПК, рассчитанный не только на автоматический анализ, но и на преобразование исходных текстов программ. Эффективность такого АПК в значительной степени зависит от его функционального наполнения, т.е. от множества функций, которые он может выполнять.

О назначении АПК, предназначенного для функционально-смыслового анализа текстов программ и программных систем, и о его функциональном наполнении

Важно, чтобы АПК мог выполнять четко определенное множество задач, связанных с его функциональным назначением.

К основным общим задачам АПК в нашем случае относятся:

- выявление элементов объектной программы описанных, но практически не используемых;
- исследование исходного кода объектной программы в целях последующего переноса программного продукта на другую платформу;
- реструктурирование исходного кода объектной программы с помощью материала, накапливаемого в специально созданном хранилище моделей программ;
- исследование исходного кода в целях последующей модернизации объектной программы и получения нового программного продукта.
- семантический анализ исходного кода объектной программы, на который отсутствует документация, в целях определения её функционально-смыслового (интенционального) назначения.

Говоря о конкретном ***функциональном наполнении АПК***, рассчитанного на возможность анализа текстов простых программ и больших программных систем, важно отметить следующий принципиальный момент. Известно, например, что модель данных большой программной системы, реализованной на языках программирования С и С++, имеет форму множества директорий, содержащих различные по предназначению (использованию) файлы: файлы с исходными кодами, файлы с описанием типов объектов и вспомогательные файлы. С учётом этого, одним из эффективных подходов к выполнению смыслового анализа простых программ и больших программных систем может быть подход, реализующий моделирующее отображение их модулей, которое предполагает построение графов смысловых отношений. При этом графы модельного представления могут отличаться по уровню отображения внутренних объектов, их смыслов и связей между объектами и смыслами.

Перечень конкретных функций, которые выполняются уже АПК-2, включает в себя:

- возможность построения графа связей файлов системы (важно иметь возможность видеть, каким образом исходные файлы проекта подключают друг друга);
- возможность отображения иерархии наследования классов;

- возможность отображения диаграммы вызовов функций (такая диаграмма показывает, каким образом управление передается выбранной пользователем или системой функции и куда оно передается из нее; связь на диаграмме соответствует вызову функции);
- возможность просмотреть участки кода, в которых данный элемент используется, для любого модуля программы;
- возможность преобразования исходных файлов исследуемой программы в модель и помещения модели в хранилище данных;
- возможность выделения из всей анализируемой программы только необходимой функциональной части и сборка только этой части.

Об особенностях реализации аналитического программного комплекса второго этапа

В рамках проекта второго этапа был разработан вычислительная аналитический программный комплекс АПК-2, предназначенный для простейшего автоматизированного логико-смыслового анализа обширного класса модулей программ и программных систем в целях изучения возможности их повторного использования. Его идеология построена на основе: (1) технологии программирующего моделирования, реализованной с использованием идеологии теории графов [1], (2) технологии смыслового анализа объектных программ, построенной на общих принципах смыслового анализа [5, 6] и (3) вычислительной технологии автоматического смыслового анализа программ [3]. Система, созданная на основе этой идеологии, предполагает автоматический анализ текстовых форм программ и получение графического отображения итогов анализа (Рис. 1).

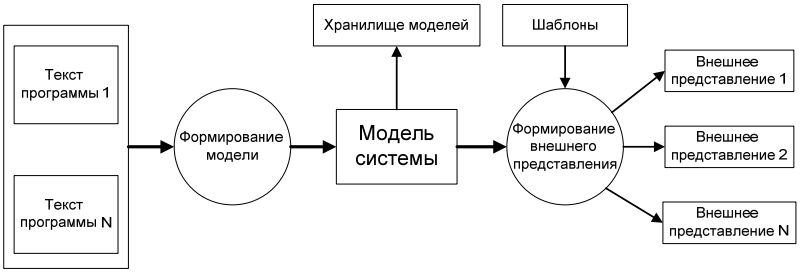

Рис. 1. Схема процесса анализа программной системы, построения её модели и отображения этой модели в различные её внешние представления

АПК-2 позволяет осуществлять наглядный анализ ОПС, даже если они имеют весьма большой размер. Такой анализ предполагает построение графа функциональных связей в анализируемой ОПС по её исходному коду, что уже обеспечивает наглядную (и удобную при обучении) и эффективную навигацию аналитика по файлам с выявлением средствами системы и наглядным отображением основных свойств ОПС (типов, методов, функций, использованных в ОПС). Всё это создает информационную основу для поис-

ка возможных способов корригирующего преобразования исходного текста ОПС в последующих модификациях комплекса (версия АПК-3). Общая идеология решения задачи по преобразованию исходных текстов программ как раз и представлена на рис. 1.

Относительная простота идеологии системы и её реализации в АПК-2 обеспечивает возможность эффективного обучения пользованию этим программным комплексом.

Логическая основа АПК-2 была написана на скриптовых языках Lua и Python. АПК-2 предусматривал решение задачи статического анализа кода анализируемой программы с помощью набора оригинальных технологий, предполагающих выявление и отображения функционально-смысловых связей в исходном тексте ОПС. Такой подход к решению задачи включал следующие этапы: 1) статический анализ кода объектной программы (глубина анализа, выполняемого АПК-2, может варьировать от определения поведения отдельных операторов программы, отдельных её функций и далее – до анализа всего исходного кода); 2) реализуется оригинальная технология автоматического построения графов функциональных связей в тексте анализируемой программы; 3) АПК-2 пригоден для операций, предполагающих простейший смысловой анализ: оперативное выявление мест в объектной программе, содержащих ошибки, идентификация свойств объектной программы, проверка её соответствия спецификации и т.п.

О возможностях, предоставляемых статическим анализом исходного текста анализируемой программной системы

Поясним, что означает *статический анализ кода* (САК). Заметим, что данный термин обычно применяют к анализу, производимому специальным программным обеспечением. Заметим, что САК полезен при исследовании и проверке надежности и свойств программного обеспечения, применяемого в современных вычислительных системах высокой производительности. САК производится без реального выполнения исследуемой программы, в большинстве случаев – над версией её исходного кода, хотя в принципе анализу может быть подвергнут какой-нибудь один вид объектного кода. Рис. 2 детализирует общий технологический принцип статического анализа исходных текстов (кода) программ, выполняемого АПК-2.

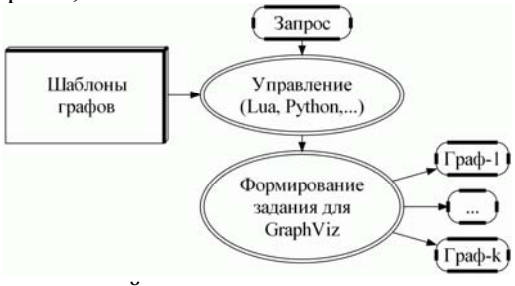

Рис. 2. Технологический принцип статического анализа исходных текстов программ (по существу это – технология формирования графов связей)

Заметим, что существенный прорыв в решении задачи анализа текстов больших программных систем был обеспечен за счёт технологического решения, связанного с найденным авторами подходом к семантической декомпозиции и, как следствие, раздельному представлению модулей весьма сложных алгоритмов больших объектных программ. Раздельное представление, а затем оригинальное решение задачи сборки (know-how), способствовали эффективному решению задачи логико-семантического анализа.

Действительно, если оказывается возможной декомпозиция системы на основании некоторого принципа или правила (например, правила применения кластеризации к графу), то можно найти подсистемы, подчиняющиеся этому принципу/правилу (кластеру). Такой подход к анализу архитектуры программных систем позволяет не только (1) упростить понимание внутреннего устройства программной системы, но и (2) осуществить её рефакторинг, т.е. выполнить моделирование архитектуры объектной программной системы с целью её улучшения. В свою очередь, наличие модели ОПС позволяет (3) выявить наличие неоправданного дублирования подсистем в ОПС, (4) обнаружить в ОПС сильно связанные компоненты и т.п. Доступность таких уникальных возможностей, предоставленных АПК-2 исключительно важна. Кроме того, наличие работающего АПК-2 дает возможность научить аналитиков использовать указанные функции в практике.

Важным в аспекте понимания принципа функционирования АПК-2 является то, что он предполагает возможность обращения в Хранилище моделей ранее проанализированных программ и модулей программ, с помощью этих моделей выполнить статический анализ исходного текста анализируемой программы и, в итоге, сформировать по задаваемым шаблонам (назовём их учебными шаблонами) графы связей различного внешнего представления и различной степени детализации.

Для усвоения технологического принципа САК важно иметь обучающую систему, наглядно демонстрирующую возможности системы и этапы применения статического анализатора кода в анализе. Необходимость в такой системе подсказала мысль о необходимости создания оригинальной технологии наглядного отображения. Ею стала разработанная нами оригинальная технология автоматического формирования графов связей в текстах анализируемых программ и программных систем.

О применении теории графов и создании оригинальной технологии отображения

Применение теории графов в качестве аппарата отображения модели анализируемой ОПС делает её модельное представление наглядным и при этом позволяет, как выяснилось, существенно упростить последующую интерпретацию её сложных структур и алгоритмов. Оба эти фактора имеют большое значение с точки зрения решения задач не только анализа, но и обучения.

В ходе работы АПК-2 реализуется идея применения предустановленных шаблонов различных представлений графов. Анализ позволяют получить представление об отношениях в модели: (1) о функциональных связях (Рис. 3), (2) о связях по глобальным объектам, (3) о связях по описателям объектов. Это позволяет получить исчерпывающую низкоуровневую (не смысловую) информацию об ОПС. Она является основой в семантическом анализе.

Схема функциональных связей (СФС), идентифицируемых в модели ОПС дает информацию о наличии описания различных функций в файлах, расположенных в разных директориях ОПС, а также о том, где и когда именно эти функции вызываются (каждая связь на диаграмме соответствует вызову функции). Каждая функция имеет адрес вида «Директория, Файл, Функция». Для построения подграфа связей, соответствующего конкретной функции, делается соответствующий запрос. На схеме (Рис. 3) показано, как выглядит рекурсия (см. рекурсию для функции N из файла 1 директории 1).

Кроме СФС реализуются ещё два модельных представления: граф связей по глобальным объектам (ГСГО) и граф связей по описателям объектов структуры программы (ГСОО). ГСГО похож на СФС, но отображает использование глобальных переменных, констант и объектов. ГСОО отображает зависимости иерархии для типов, классов, структур, расположенных в различных частях текста (файлах). Например, связь вида «Тип1 → Тип2» означает, что Тип 2 является одним из составляющих Типа1. В терминах классов это – отношение вида «Потомок → Родитель». Полное представление о системе модельных отображений функций АПК-2 представляет исключительный интерес с точки зрения понимания технологии программирующего моделирования. Его описание требует отдельной статьи.

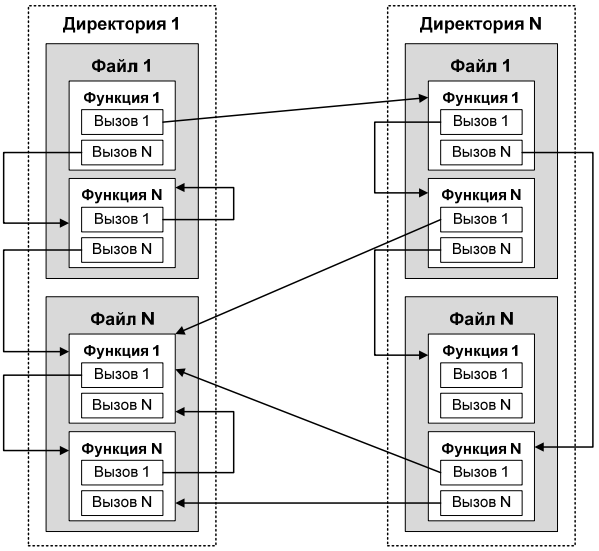

Рис. 3. Схема функциональных связей, идентифицируемых в модели ОПС

Заключительный этап формирования аналитического графа заключается в создании его текстового описания и последующей передаче этого писания компоненту Graphviz, осуществляющему визуальное отображение графа.

Заключение. Возможности АПК-2

Итак, реализуя вышеописанную технологию, АПК-2 выполняет статический анализ текста (исходного кода) ОПС и формирует по шаблонам, задаваемым пользователям (а при обучении – по учебным шаблонам), графы связей различного внешнего представления и необходимой степени детализации (см. рис. 4, где представлены итоги частичного анализа связей внутри исходного кода и связей с множеством функций типа sys).

Выполненный анализ позволяет аналитику, использующему АПК-2, оценить иерархию связей типа «Потомок → Родитель» в анализируемой программе, обнаружить неиспользуемые объекты, рекурсии, оперативно выявить места в ОПС, содержащие ошибки, идентифицировать свойства объектной программы, проверить её соответствие спецификации.

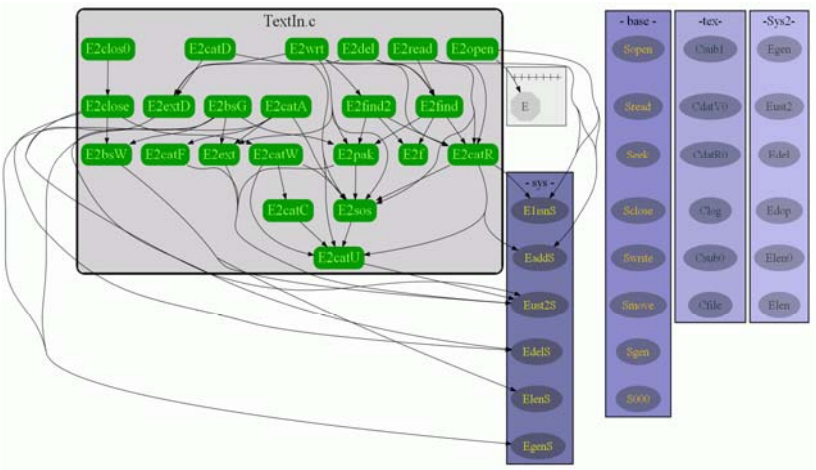

Рис. 4. Граф функциональных связей, построенный в итоге первого этапа анализа текста

Библиографические ссылки

1. *Чернышов М. Ю.* О реализации вычислительной системы, предназначенной для логико-смыслового анализа программ и программных систем / М. Ю. Чернышов, Н. В. Абасов // Информационные технологии и высокопроизводительные вычисления. Материалы международной конференции (Хабаровск, 4–6 октября 2011 г.). Хабаровск: Изд-во ТОГУ, 2011. С. 116–124.

2. *Абасов Н. В.* Основы универсальной среды программирования ЗИРУС / Н.В.Абасов // Вестник ИрГТУ. 2006. № 2(26). С. 62–68.

3. *Абасов Н. В.*, *Чернышов М. Ю.* На пути к вычислительной технологии для эффективного исследования программ и программных систем, основанной на принципах логико-смыслового анализа / Н. В. Абасов, М. Ю. Чернышов // Информационные технологии и высокопроизводительные вычисления. Материалы международной конференции (Хабаровск, 4–6 октября 2011 г.). Хабаровск: Изд-во ТОГУ, 2011. С. 3–15.

4. *Абасов Н. В., Миронов В. О.* Применение графов для анализа и реструктуризации исходных текстов программ / Н. В. Абасов, В. О. Миронов // Труды XIV Всероссийской Байкальской конф. “Информационные и математические технологии в науке и управлении”. Т.3. Иркутск: ИСЭМ СО РАН, 2009. С. 173–178.

5. *Чернышов М. Ю.* Вербальные средства выражения мыслей-скрепов как медиаторов смысловой интегративности текста / М. Ю.Чернышов. М.: Наука и право, 2010. 168 с.