Bartashevskiy S.Ye., Bartashevskaya L.I.
National Mining University

JUSTIFICATION OF CHOICE PERSPECTIVE CHEMICAL CURRENT SOURCES FOR MINE ELECTRIC LOCOMOTIVES

Summary
The paper concerns basic ways to improve the efficiency of locomotive transport periodic action by increasing its autonomy. Comparative analysis the energy and economic efficiency batteries of different electrochemical systems. The classification of fuel cells by operating temperature and type of fuel used. The physicochemical properties of employed fuels and problems, arising during storage and use are analyzed. Selected the most perspective type of fuel cells, which is the operating temperature and their fuel types will exploit it on electric locomotives in Ukrainian coal mines.
Keywords: mine locomotive, electric battery, autonomy, electrolyte, battery, a fuel cell.

УДК 004.415.2

РЕИНЖИНИРИНГ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ГЕОМЕТРИЧЕСКОЙ МОДЕЛИ КОРПУСА СУДНА

Давыденко Е.А.
Черноморский государственный университет имени Петра Могилы

Рассмотрено концепцию создания мобильного программного обеспечения. Предложен метод многоуровневого структурного проектирования программ. Проанализирована послойная модель программного обеспечения. Проведен реинжиниринг САПР ДЕЙМОС. В результате было сохранено действующую архитектуру и добавлен новый функционал.
Ключевые слова: САПР, ДЕЙМОС, реинжиниринг, программное обеспечение, ACIS.

Постановка проблемы. В последнее время наблюдается тенденция к увеличению продолжительности жизненного цикла успешных программных проектов. Как следствие, растет объем унаследованного кода, поддерживаемого сообществом разработчиков [1]. Именно это объясняет исключительную важность задач, связанных с облегчением сопровождения и развития существующего программного кода. В то же время, этим задачам уделяется недостаточное внимание со стороны научного сообщества и разработчиков инструментальных средств. Как следствие, современные методики переоценивают значение начальной фазы жизненного цикла программной системы и практически игнорируют ее дальнейшую эволюцию. Таким образом, в настоящее время существует явный недостаток методик и эффективных инструментов поддержки работы с существующим кодом.

Так же, в последнее время наметился перелом ситуации: стали вызывать значительный интерес вопросы систематического использования трансформаций как центрального организующего принципа процесса развития и сопровождения существующего программного обеспечения (ПО). Однако большинство исследователей рассматривает трансформации достаточно узко – как трансформации на уровне исходного кода – рефакторинг [2]. Тем не менее, в настоящее время практически не существует исследований, посвященных трансформации на более высоком уровне абстракции – уровне архитектуры ПО. В то же время, многие сценарии сопровождения и развития существующего кода подразумевают изменение архитектуры существующей системы. В связи с этим, большой интерес вызывает разработка методологии и сопровождающих ее инструментальных средств, нацеленных на организацию предсказуемого и управляемого процесса изменения архитектуры ПО.

Анализ предметной области. Развитие автоматизации проектных работ на начальных этапах внедрения вычислительной техники (ВТ) происходило в направлении автоматизации сложных инженерно-технических расчетов. Это обеспечивало достаточно быструю окупаемость внедрения ВТ. Отдельной задачей автоматизации стояли чертежно-графические работы. Для их решения создавались специальные компьютерные комплексы (Benson, Konsberg и др.), которые принимали данные через магнитные ленты и выполняли вычерчивание сложных чертежей с большой точностью на чертежных столах (для работ по проектированию габаритных технических объектов в 70-80 годах прошлого столетия их использование было просто необходимо). Естественно, что развитие информационных технологий привело инженеров-программистов к идее использования моделей конструкций. Однако ограниченность ресурсов компьютеров существенно сдерживала объемы моделей и количество задач, которые можно было решать при использовании этих моделей. Для расширения моделей и соответственно задач, решаемых на их основе, стали применяться различные системы управления базами данных (СУБД). При этом, в условиях жесткого ограничения по ресурсам (оперативная память многих ЭВМ была 1-4 МБ), огромное количество кода было написано с использование низкоуровневых языков программирования (Ассемблер). Программирование инженерных расчетов традиционно выполнялось на языке Fortran, а различных экономических задач – на языке Cobol. Системные задачи реализовывались на языке PL/1. В 80-х годах прошлого столетия уже стало очевидна необходимость создания интегрированных систем автоматизации, но для ее решения необходимы были единые подходы к созданию программ и интегрирующие средства. Попытки изменения моделей

приводили к огромным изменениям в программах по решению тех или иных задач, что приводило к большим ресурсным затратам.

Ситуация изменилась с появлением языка С в 80-х годах. Он сочетал в себе как возможности языков высокого уровня, так и языков низкого уровня. С его появлением наблюдается резкий количественный рост различных систем автоматизации (СА) и повышение их качественных показателей. Например, среди СА проектирования судов появились такие системы как Foran, Tribon, Autoship, которые и на сегодня являются лидерами в судостроении. Правда, нельзя этот резкий скачок относить только к языку С – в этот же период происходят существенные изменения и в элементной базе, и в программно-аппаратных платформах. Преобладающей на то время была парадигма структурного программирования, а модель ПО – блочно-модульная (рис. 1).

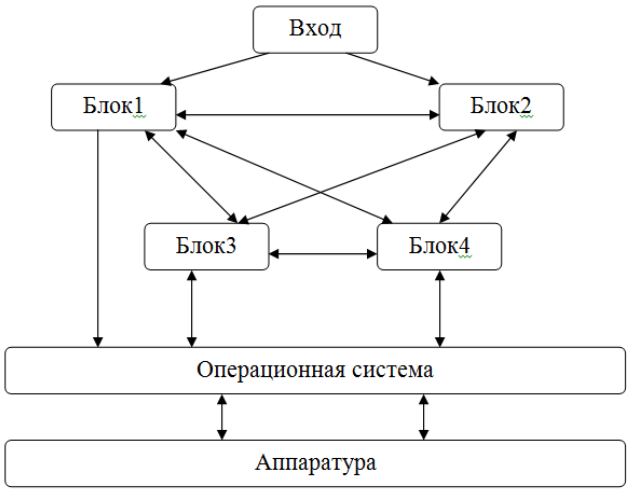

Рис. 1. Сборочная блочно-модульная структура ПО

Этот подход к разработке ПО состоит в организации системы как набора процедур, каждую из которых может вызывать любая пользовательская процедура. Была создана теоретическая база так называемого *сборочного программирования* и инструментальная система автоматизации проектирования программ. Однако такая сборочная структура не обеспечивает изоляции данных; в разных участках кода используется информация об устройстве всей системы. Расширение такого программного продукта затруднительно, так как изменение некоторой процедуры может вызвать ошибки в других частях системы, на первый взгляд не имеющих к ней отношения. Кроме того, отладка и поиск ошибок в такой системе всегда имеет более длительный промежуток времени. Перенос же такого программного продукта на другую платформу вообще становится сложной задачей.

Поскольку объемы кода ПО резко возрастало вслед за соответствующим повышением технических характеристик ЭВМ, то вписывать его в рамки монолитной модели становилось крайне затруднительно и неэффективно, то на смену этой методологии пришла методология *многоуровневого послойного структурного проектирования* программ. Сначала эта методология нашла теоретическую базу для микропрограммного уровня. Так, в [3] теоретическая база основана на аппарате САА-М (систем алгоритмических алгебр, предложенных в 1965 году академиком В.М. Глушковым), которая позволила предложить метод многоуровневого структурного проектирования программ и его инструментарий. Однако этот аппарат нашел применение только для микропрограммирования.

Что касается ПО систем автоматизированного проектирования (САПР), то к этому времени в качестве технического обеспечения появились рабочие станции, на которых устанавливались ОС Unix и которые имели мощность мини-ЭВМ и даже выше, при этом они имели широко развитый графический интерфейс. При этом, изначально предлагались для разработки стандартизированные мобильные средства. Сама ОС Unix на тот момент предполагала послойную организацию программного кода, внизу которого располагалось ядро ОС [4].

При послойной модели ПО подход к структурированию системы предполагает разделение ее на модули, наслоенные один поверх другого. Каждый модуль предоставляет набор функций, которые могут вызываться другими модулями (рис. 2). Код, расположенный в некотором слое, вызывает код только из нижележащих слоев. В некоторых случаях, многослойная организация кода принудительно обусловлена аппаратурой и ОС – например VAX/VMS [5].

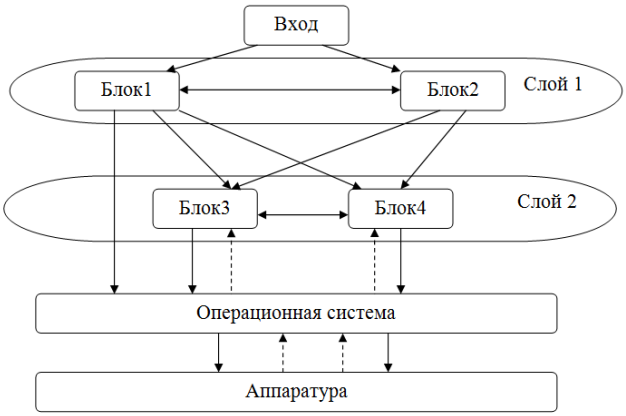

Рис. 2. Послойная структура ПО

Одним из преимуществ послойной организации кода является то, что код каждого слоя получает доступ только к необходимым ему интерфейсам (и структурам данных) нижележащих слоев: таким образом, уменьшается объем кода, обладающего неограниченной властью. Кроме того, такая структура позволяет начинать процесс отладки с самого нижнего слоя и добавлять по одному уровню до тех пор, пока вся система не станет работать правильно. Послойная структура облегчает и расширение системы: можно целиком заменить любой уровень, не затрагивая остальных частей.

В 90-х годах прошлого столетия появилась совершено новая по своей концептуальной сущности концепция программирования – *объектно-ориентированное программирование* (ООП). Это результат 30-летнего опыта программирования и развития таких языков и технологий программирования как Simula-67, Lisp, SmallTalk, ObjectiveC, Java и C/C++. Эта методология является стилем программирования, который фиксирует поведение реального мира таким образом, что внутренняя реализация скрыта [6]. Особый эффект получился при добавлении принципов объектно-ориентированного программирования в язык С. В результате появился язык С++, который стал мощнейшим средством программирования, который обладает, благодаря возможностям самого языка С, мощностью языка низкого уровня и, сверх того, выразительностью и компактностью языка высокого уровня.

Основной целью разработки мобильной программной системы с ориентацией на данные является создание ПО, которое можно было бы легко (и дешево) изменять. Насколько важна способность к модификации, становится ясно, если принять во внимание часто приводимую статистику, согласно которой 70% цены программного продукта падает на сопровождение. Сопровождение включает в себя введение новых возможностей, модификация форматов данных, исправление ошибок и адаптацию к новому оборудованию. Один из способов минимизации необходимых изменений в объектно-ориентированных программах – это сокрытие физического представления данных внутри объектов. *Объект* – это структура данных, физический формат которой скрыт в определении типа. Он имеет набор свойств, называемых *атрибутами*, и с ним работает группа сервисов.

Учитывая тот факт, что Украина является страной с большим судостроительным потенциалом, была разработана и утверждена Кабинетом министров национальная программа развития судостроения. В ней была предусмотрена разработка отечественной САПР судов, чтобы судостроительные конструкторские бюро и заводы в определенной степени чувствовали себя независимыми от иностранных фирм-разработчиков САПР и могли бы обеспечить выпуск конструкторско-технологической документации в соответствии с национальными нормативными документами. В особой степени это относится к САПР корпуса судна, поскольку эта подсистема является системообразующей составляющей САПР судна в целом. Имея такую базовую подсистему и развитые адаптивные интерфейсные средства, можно обеспечить передачу информации в другие, смежные системы автоматизации, а также можно обеспечить конвертацию информации из других (зарубежных) САПР судов, если они используются на отдельных предприятиях. В рамках вышеупомянутой программы создана САПР корпуса судна ДЕЙМОС [7], которая внедрена на многих предприятиях Украины и России (разработчиками и владельцами являются ОАО «НИИ «Центр» и СП «ГРАСКО», г. Николаев). Только в ОАО «НИИ «Центр» при непосредственной эксплуатации этой системы разработано более 30 проектов судов, которые были построены и спущены на воду.

Выделение нерешенных ранее частей общей проблемы. В связи с недостаточным финансированием и даже его полным отсутствием, в течение большого периода не представлялось возможности разработать или усовершенствовать некоторые модули ПО. В том числе модуль интеграции геометрической модели САПР ДЕЙМОС с некоторыми международными стандартами представления данных.

Постановка задачи. Целью данной работы является реинжиниринг САПР ДЕЙМОС для решения проблемы файлообмена геометрических моделей в соответствии с международными форматами, такими как IGES, SAT и др.

Изложение основного материала исследования. Необходимость изменения (реинжиниринга) существующего ПО возникла в ходе решения широкого круга задач по его модернизации [8, 9]. В общем случае изменения существующего ПО способны затронуть не только его код, но и все остальные артефакты, связанные с программной системой, которая трансформируется. Одна из наиболее существенных разновидностей – *это изменение архитектуры программного обеспечения.* В качестве примеров можно привести такие показатели, которые требуют изменений архитектуры существующего ПО:

· *Превращения, обусловленные функциональными изменениями ПО*. При этом желательно, чтобы внедрение новой функциональности не повлияло на существующую логику системы. Также желательно, чтобы сложность внедрения новой функциональности в существующую систему не превысило существенным образом сложность реализации этой функциональности в рамках нового проекта. Красивая архитектура позволяет достичь поставленных целей [3]. Таким образом, изменение существующей архитектуры – важный шаг на пути внедрения новой функциональности, который, к тому же, облегчает последующую эволюцию системы.

· *Изменение платформы ПО*. Крайне желательно, чтобы изменение платформы ПО как можно меньше повлияло на существующий код, и, чтобы можно было ограничиться изменениями только в узкой платформо-зависимой прослойке системы. Выделение такой прослойки – архитектурная задача. Ее решение всегда связано с необходимостью изменения архитектуры.

· *Превращения, которые связаны с реорганизацией компании-разработчика ПО*. Примером такой реорганизации может стать перевод реинжиниринга существующего ПО на аутсорсинг. Решение о привлечении аутсорсинга – типичный шаг для оптимизации производства в общем и программных продуктов в частности. К сожалению, этот шаг часто осложняется проблемой выделения и передачи компонентов для внешней разработки.

ACIS это объектно-ориентированная C++ геометрическая библиотека, состоящая из более чем 35 DLL-файлов и включает каркасные структуры, поверхности и твердотельное моделирование. Она дает разработчикам программ богатый выбор геометрических операций для конструирования и манипулирования сложными моделями, а также полный набор булевых операций. Ее математический интерфейс Laws Symbolic и основанная на Non-Uniform Rational B-Splines (NURBS) деформация, позволяют интегрировать поверхностное и твердотельное моделирование. Ядро ACIS осуществляет вывод в форматы файлов SAT и IGES, которые любая программа, поддерживающая ACIS может читать напрямую. Именно эти форматы очень часто используются в судостроении. Поэтому целью исследований было решить задачу интеграции САПР ДЕЙМОС с ядром ACIS 3D Geometric Modeler.

Одной из главных проблем САПР ДЕЙМОС является невозможность импорта некоторых файлов формата IGES. В ходе исследований было решено конвертировать IGES-файл с помощью Transmagic Innovate CAD Converters в формат SAT. Реализация этого метода является наиболее целесообразным, поскольку SAT – это стандарт, с которым работает ACIS, поэтому не возникнет никаких проблем с поддержкой файла.

Для реализации этого метода предлагаются следующие шаги (рис. 3):

- добавить поддержку Transmagic Innovate CAD в проектные файлы ДЕЙМОС.
- добавить функцию конвертирования из IGES-файла в SAT-файла.
- скачать SAT-файл и продолжать работу с документом.

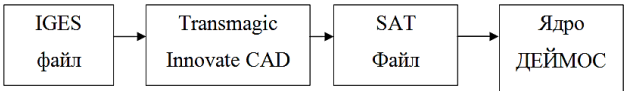

Рис. 3. Загрузка IGES-файлов с помощью Transmagic Innovate CAD

Программное обеспечение, реализующее указанную интеграцию, представляет собой класс Converter. Этот класс имеет статический метод ConvertFromIges2Sat, который осуществляет преобразование формата IGES в формат STEP. На рис. 4 представлена диаграмма классов в нотации UML, иллюстрирующую процесс загрузки IGES файла.

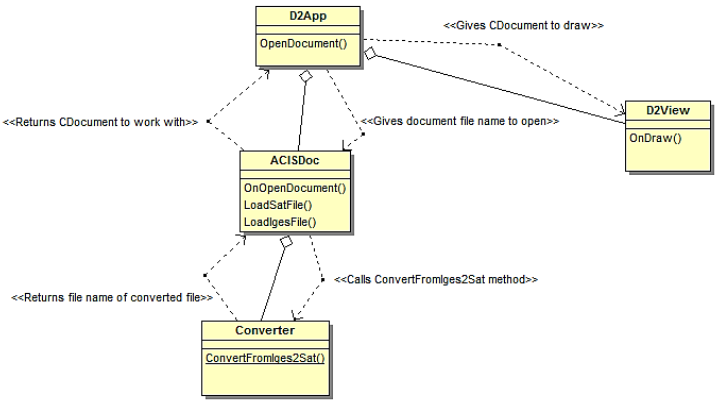

Рис. 4. Диаграмма классов процесса загрузки IGES файла

Выводы. В результате решена одна из самых больших проблем САПР ДЕЙМОС – отсутствие возможности импортирования IGES файла. Поскольку наиболее подходящий формат для работы с ядром данной САПР это SAT, поэтому принято решение использовать конвертер фирмы Transmagic для создания временного SAT файла, который затем будет загружен в программу. Выбор стороннего ПО обусловлен тем, что стандарт формата SAT сейчас лицензированный, и написать собственный конвертер затруднительно. Во время тестирования работы ПО не обнаружено никаких проблем с имплементацией ядра.

Список литературы:

1. A. van Deursen. Research Issues in The Renovation of Legacy Systems, A. van Deursen, P. Klint, C. Verhoef, CWI research report P9902, April 1999.
2. Фаулер М. Рефакторинг: улучшение существующего кода. – Пер. с англ. – СПб: Символ-Плюс, 2003. – 432 с.
3. Ющенко Е. Л. Многоуровневое структурное проектирование программ. Теоретические основы, инструментарий / Е. Л. Ющенко, Г. Е. Цейтлин, В. П. Грицай, Т. К. Терзян. – М.: Финансы и статистика, 1989. – 208 с.
4. Биков Д. П., Фісун М. Т. Пошарова модель розробки програмного забезпечення САПР суден // Наукові записки НУКМА, т. 22 у трьох частинах. Частина III: Природничі науки. – Київ: Вид. дім «КМ Академія». – 2003. – С. 477-483.
5. Mansurov D. Campara, Klocwork, Managed Achitecture of Existing Code as a Practical Transition towards MDA, № 1 Chrysalis Wat, Ottawa, Canada.
6. Бьерн Страуструп. Язык программирования С++. Специальное издание – СПб: Бином, Невский Диалект, 2004. – 1104 с.
7. Дубів І. І. Система деталювання й моделювання корпусу судна ДЕЙМОС: основні принципи та загальна структура. – Миколаїв: УДМТУ, 2001. – В зб. «Труди УДМТУ», вип. 43. – С. 32-39.
8. Давыденко Е. А. Использование методов системного анализа при реинжиниринге программного обеспечения САПР // Universum: Технические науки: электрон. научн. журн. 2013. № 1(1). URL: http://7universum.com/en/tech/archive/item/786 (дата обращения: 25.12.2013).
9. Фісун М. Т., Давиденко Є. О. Реінжиніринг програмного забезпечення модуля генерації керуючих програм в САПР ДЕЙМОС / М. Т. Фісун, Є. О. Давиденко [Текст] // Сборник научных трудов SWorld. – Выпуск 4. Том 12. – Иваново: МАРКОВА АД, 2013 – С. 30-34.

Давиденко Є.О.
Чорноморський державний університет імені Петра Могили

РЕІНЖИНІРИНГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ГЕОМЕТРИЧНОЇ МОДЕЛІ КОРПУСУ СУДНА

Анотація
Розглянуто концепцію створення мобільного програмного забезпечення. Запропоновано метод багаторівневого структурного проектування програм. Проаналізована пошарова модель програмного забезпечення. Проведено реінжиніринг САПР ДЕЙМОС. В результаті було збережено діючу архітектуру і доданий новий функціонал.
Ключові слова: САПР, ДЕЙМОС, реінжиніринг, програмне забезпечення, ACIS.

Davydenko Y.O.
Petro Mohyla Black Sea State University

REENGINEERING OF GEOMETRIC MODEL OF SHIP HULL

Summary
Considered a concept of creating mobile software. Proposed a method of multilevel structural design programs. Analyzed layered software model. Was conducted reengineering CAD DEIMOS. As a result, the saved existing architecture and adds new functionality.
Keywords: CAD, DEIMOS, reengineering, software, ACIS.