УДК 629.7.024.01

Методы и технологии обмена данными систем автоматизированного проектирования

А.А. Буряков

При современных все ускоряющихся темпах научно-технического прогресса динамика проектирования сложных объектов, таких как самолет, содержащий 10^6-10^8 *деталей, является одной из важнейших его характеристик, и сокращение сроков и стоимости проектирования становится одним из главных требований. Использование современных систем автоматизированного проектирования меняет как средства, так и методы решения проектных задач. Для получения оптимального результата проектирования необходимо использование ряда функциональных возможностей одной или нескольких существующих прикладных систем и дополнительно разработанных специализированных модулей. В статье проводится обоснование и выбор методов и технологии обмена данными систем автоматизированного проектирования.*

Последние десятилетия характеризуются появлением на мировом рынке систем автоматизированного проектирования (САПР, в английской нотации Computer Aided Design - CAD), решающих широкий спектр задач геометрического моделирования, инженерного анализа, электронного документооборота и т.д. Условно структуру существующего рынка CAD-систем можно представить в виде пирамиды. На нижнем уровне этой пирамиды находятся такие системы как КРЕДО, AutoCAD, T-Flex; на следующем - системы среднего уровня (SolidWorks, Solid Edge, Autodesk Inventor и т.д.); и на вершине - системы высшего уровня (CATIA, Unigraphics, Pro/Engineer и т.д.) [1].

Рис. 1. Структура рынка CAD-систем

В этих системах реализован довольно широкий набор математических и технических методов и алгоритмов, позволяющий применять их в диапазоне от простого построения модели (системы AutoCAD, КРЕДО) до построения и расчета характеристик даже такого сложного изделия как самолет или подводная лодка (системы CATIA, Unigraphics, CADAM). Не вдаваясь сейчас подробно в анализ данных систем и их функциональных возможностей, можно тем не менее сделать вывод, что наиболее полные возможности проектирования, естественно, имеют системы высшего уровня, но, специалист, использующий эти системы, кроме высокой квалификации в своей узкой области знаний, должен быть профессионалом и в области работы с компьютером. Кроме того, стоимость таких систем весьма значительная. Системы среднего уровня в настоящее время по своим функциональным возможностям приближаются к системам верхнего уровня, сохраняя более низкую стоимость.

Разные системы имеют отличающиеся функциональные возможности и для улучшения качества проектирования, более многостороннего анализа изделия, часто необходимо использовать функциональный аппарат сразу нескольких систем. В таких случаях возникает необходимость обмена данными о геометрической модели между системами. При этом важно, чтобы модель, полученная после передачи данных из одной системы в другую, была идентична исходной.

В результате проведенного анализа наиболее распространенных CAD-систем, было выявлено следующее (рис. 2):

1. Существует очень ограниченное множество систем, поддерживающих возможность прямого чтения форматов данных другой системы (например, системы Solid Edge и Unigraphics) или возможность передачи данных с использованием конвертирования, полностью сохраняющего структуру взаимосвязей изделия;

2. Существует класс систем, поддерживающих нейтральные форматы обмена данными STEP и IGES, но часто данные форматы реализованы в различных системах по-разному и различном объеме [2];

Рис. 2. Схема возможного обмена данными между системами
(сплошные линии – полная поддержка формата, широкий пунктир – поддержка форматов STEP и IGES; узкий пунктир – поддержка формата DWG)

3. Для обмена данными (преимущественно для их загрузки) с системами, не поддерживающими форматы STEP и IGES (например, с AutoCAD), некоторые фирмы встраивают в свои продукты поддержку их внутреннего формата;

4. С развитием систем происходит одновременное развитие и усложнение как их внутренних форматов, так и нейтральных форматов обмена данными, таких как STEP и IGES.

Нейтральные форматы данных IGES и STEP находятся в состоянии развития – по мере совершенствования систем, в новые версии формата IGES включается все большее и большее число поддерживаемых геометрических элементов, у формата STEP по мере стандартизации частей, входящих в его состав, происходит их последовательное включение в состав спецификаций, поддерживаемых разработчиками CAD/CAM/CAE-систем. При использовании этих форматов возможно возникновение следующих проблем – искажение значений параметров из-за ошибок конвертации в

случае, если способ представления объекта в системе отличается от описанного в спецификации формата, потери данных в случаях, если геометрические элементы не поддерживаются в формате явно и происходит их аппроксимация. Рисунок 3 иллюстрирует изменения, происходящие с моделью при передаче ее из системы SolidWorks посредством формата IGES: при загрузке файла в систему CATIA происходит замена всех элементов на поверхности, их ограничивающие (замена твердотельного представления поверхностным), сама модель поворачивается на 90^0 (рис. 3, слева), при загрузке файла в систему SolidWorks в модели происходит разрушение связок между элементами и замена всех «сборок» простыми «деталями» (рис. 3, справа).

Таким образом, при обмене данными между системами, использование механизмов прямого чтения данных других систем и конвертирования, сохраняющего структуру взаимосвязей изделия, позволяют получить модель, идентичную исходной. Использование других видов конверторов и файлов нейтральных форматов может привести к появлению ошибок.

Рис. 3. Передача данных о модели посредством IGES-файла

При всем многообразии функциональных возможностей, CAD/CAM/CAE-системы не содержат отдельно выделенных специализированных модулей, которые необходимы проектировщику.

Очевидно, решение данной проблемы возможно в трех направлениях:

- разработка системы под конкретный вид деятельности «с нуля»;
- разработка дополнительного модуля интегрированного в конкретную систему;

- разработка внешнего специализированного модуля, выполняющего необходимые функции и решение вопроса об обеспечении обмена данными с рядом CAD/CAM/CAE-систем.

При традиционном подходе к проектированию САПР для определенной области производства «с нуля», группа программистов должна обеспечить разработку модели системы и взаимное увязывание всех ее модулей в одно единое целое. При этом необходимо четко описать и согласовать входные и выходные параметры каждого модуля, входящего в ее состав. При этом, основными недостатками такого подхода являются:

- длительный срок разработки;
- высокая стоимость разработки;
- высокая сложность проектирования и реализации методов;
- «замкнутость» разрабатываемой системы относительно внешних приложений;
- сложность внесения изменений и исправления ошибок;
- система может морально и физически устареть до окончания написания.

Известен пример разработки комплексной САПР по традиционному подходу - система «Visual CAPDA», разрабатываемая под руководством профессора Х. Хаберланда в Берлине. Все модули, начиная с геометрических и кончая расчетными, разрабатываются группой специалистов специально для нее. Разработка системы ведется уже более 10 лет, но часть модулей все еще находится в разработке.

Второе направление – разработка дополнительного модуля интегрированного в конкретную систему, лишено тех недостатков, которые были отмечены у первого направления. Но вместе с тем, известно, что новые версии систем появляются 1-2 раза в год, появляются новые фирмы и происходит слияние существующих, и трудно спрогнозировать, продукт какой из фирм станет лидером на рынке. Привязка к продукту конкретной фирмы может существенно снизить привлекательность дополнительного программного обеспечения (ПО). Это является существенным аргументом в пользу создания ПО, инвариантного относительно конкретной CAD/CAM/CAE-системы.

Таким образом, третий вариант представляет собой компромисс между первыми двумя – с одной стороны, разработка модулей ведется безотносительно к конкретной системе – сокращаются сроки разработки и снижается стоимость системы, с другой стороны, необходимо решить вопрос наиболее рационального обмена данными с внешними системами, которые выбираются в соответствии с определенной методикой (например, обеспечение максимального набора функциональных возможностей).

В настоящее время существует несколько способов осуществления взаимодействия между приложениями:

- работа посредством файлов внутреннего формата;
- работа посредством командных файлов или макросов;

- взаимодействие посредством COM- и CORBA-технологий.

Эффективность этих способов и сложность их реализации очень сильно зависят от конкретной системы.

Файлы внутренней структуры приложения

Работа с файлами внутренней структуры заключается в прямой записи файла одним приложением и чтением его другим (рис. 4). Этот вариант обеспечивает высокую скорость передачи данных в случаях, если формат хранения данных достаточно прост и(или) не претерпевает изменений. С другой стороны, это самый рисковый способ взаимодействия с внешним приложением - если от версии к версии происходит изменение или развитие внутренней структуры файла или формат файла закрыт, то с выходом новой версии внешнего приложения возникают сложности решения задачи анализа новой структуры файла и разработки новых процедур, реализующих сохранение данных в этом формате, с одновременной поддержкой форматов предыдущих версий (новая версия может поддерживать только чтение форматов предыдущих версий, а запись вести только в новом формате). При реализации этого подхода, основная нагрузка приходится на процесс анализа внутренней структуры файла.

Рис. 4. Взаимодействие между приложениями посредством внешних файлов

При таком взаимодействии невозможно исключить человеческий фактор в процессе взаимодействия систем, т.к. нет эффективного способа обнаружить, закончило ли второе приложение обработку данных или нет. Кроме того, постоянное развитие и усложнение форматов данных приводят к необходимости многократной переработки процедур передачи и чтения данных для обеспечения совместимости с несколькими версиями приложений, что увеличивает стоимость реализации проекта.

Командные файлы и макросы

Отличительной особенностью такого подхода, как правило, является реализованная в системе возможность записи и выполнения макроса в процессе работы, что позволяет в дальнейшем вручную вносить изменения в его текст и добиваться требуемого результата. Примером могут служить макросы в SolidWorks, CATIA, вариации языка Visual Basic в системах Word, Excel и т.д., командные файлы для систем COSMOS/M, ANSYS, NASTRAN и др. Используя данный метод автоматизации, пользователь остается жестко привязанным к конкретной системе проектирования, что вполне устраивает разработчиков систем. Такой способ подходит для автоматизации небольших

локальных задач или в том случае, если применение других способов затруднено, и характеризуется очень низкой скоростью передачи данных и построения модели, так как выполнение команд ведется последовательно в режиме реального времени и без поддержки возможности фонового исполнения, а в случае сложной геометрии изделий командный файл может занимать до нескольких мегабайт, передачу которых необходимо осуществлять по сети для исполнения. При реализации этого подхода необходимо обратить внимание на разработку процедур и функций, осуществляющих расшифровку и обработку результатов вычислений – они являются наиболее узким местом в реализации данного подхода. Также как и при взаимодействии посредством файлов внутренней структуры, данный метод в первую очередь опирается на возможности файловой структуры ОС, в которой сохраняется файл с командами и результаты расчетов.

Укрупнено схема взаимодействия с внешними приложениями посредством командных файлов представлена на рис. 5.

Рис. 5. Схема взаимодействия посредством командных файлов

COM- и CORBA-технологии

Выгодно отличаясь от первых двух вариантов взаимодействия, данные технологии позволяют осуществлять взаимодействие между приложениями на уровне объектов [3], когда можно работать со свойствами конкретного объекта, а не только с примитивами, его составляющими. С помощью объектных моделей множество объектов приложения, в том числе и на различных платформах, взаимодействуют друг с другом и реализуют бизнес–процессы, создавая видимость единого целого.

Основное назначение CORBA и COM - поддержка разработки и развертывания сложных объектно-ориентированных прикладных систем.

Архитектуру распределенных вычислений, получившую название Common Object Request Broker Architecture (CORBA) поддерживает консорциум OMG, в который входят такие компании как IBM, Sun Microsystems, HP, DEC и ряд других производителей. Развиваясь с 1991 года, сегодня она применяется многими крупными организациями на базе Unix-серверов и мэйнфреймов.

Другое технологическое направление представляет Microsoft, создавшая распределенную компонентную объектную модель Distributed Component Object Model (DCOM), которая встраивается в операционные системы семейства Windows и объединяет в себя следующую серию спецификаций – COM, COM+, DCOM.

Функции CORBA и COM - это функции промежуточного программного обеспечения объектной среды. Архитектуры реализуют несколько базовых принципов:

- независимость от физического размещения объекта;
- независимость от платформы;
- независимость от языка программирования.

CORBA и COM - это клиент–серверные технологии, в которых функциональность объекта предоставляется клиенту посредством обращения к абстрактным интерфейсам. Интерфейс определяет набор методов, которые реализуют функции, присущие данному классу объектов. Интерфейс дает клиенту возможность только вызывать тот или иной метод, скрывая от него все детали его реализации. Клиент получает доступ к объекту только путем вызова метода, определенного в интерфейсе объекта - реальные действия выполняются в адресном пространстве объекта, возможно, удаленном по отношению к процессу клиента. Сокрытие деталей реализации и позволяет в конечном итоге добиться слаженного взаимодействия компонентов в независимости от того, где и на какой платформе они реализованы и какой язык программирования для этого использовался.

Рис. 6. Механизм вызова удаленной процедуры

В обеих технологиях взаимодействие между клиентским процессом и сервером объекта, то есть процессом, который порождает и обслуживает экземпляры объекта, использует механизм объектный вариант вызова удаленной процедуры (RPC, remote procedure call). На рис. 6 показана типичная структура RPC - старейшей из технологий промежуточного программного обеспечения. Механизм RPC реализует схему передачи сообщений, в соответствии с которой в распределенном клиент - серверном приложении процедура-клиент передает специальное сообщение с параметрами вызова по сети в удаленную серверную процедуру, а результаты ее выполнения возвращаются в другом сообщении клиентскому процессу.

Для того чтобы реализовать эту схему, на стороне клиента и на стороне сервера поддерживаются специальные компоненты, носящие название клиентский и серверный суррогаты. На них возлагаются функции преобразования аргументов и результатов в универсальное, не зависящее от конкретной архитектуры представление. Для того чтобы вызвать ту или иную функцию, клиент обращается к клиентскому суррогату, который упаковывает аргументы в сообщение-запрос и переда-

ет их на транспортный уровень соединения. Серверный суррогат распаковывает полученное сообщение и в соответствии с переданными аргументами вызывает нужную функцию, или нужный метод объекта, если речь идет об объектном варианте RPC.

Важной характеристикой обеих технологий является четкое разграничение интерфейсов, которые - суть абстрактный набор связанных методов, и конкретных реализаций этих методов. Клиенту доступно описание интерфейса объекта, через которое он получает доступ к методам, то есть функциональности данного объекта. Детали реализации методов от клиента полностью изолированы. Метод вызывается по ссылке, и реальные действия выполняются в адресном пространстве объекта.

Однако за этим фундаментальным сходством начинаются значительные различия между конкретным воплощением в двух моделях понятий объектов, интерфейсов, вызова по ссылке и т.д.

В CORBA интерфейс объекта задается с помощью определенного OMG языка описания интерфейсов (Interface Definition Language). Тип объекта - это тип его интерфейса. В модели CORBA определен базовый тип для всех объектов - CORBA::Object. Объект поддерживает тип своего непосредственного интерфейса и, по принципу наследования, все его базовые типы.

В COM объект характеризуется своим классом. Класс - это реализация некоторого множества интерфейсов. Для всех интерфейсов существует базовый интерфейс - IUknown. Клиент получает доступ к объекту с помощью указателя на один из его интерфейсов (interface pointer).

В CORBA различные языки программирования поддерживаются благодаря заданным отображениям между описаниям типов данных на IDL в соответствующие определения на конкретном языке. CORBA IDL задает определения, которые могут отображаться в множество различных языков, не требуя при этом никаких изменений от целевого языка. Эти отображения реализуются компилятором IDL, который генерирует исходные коды на нужном языке. В настоящий момент поддерживается отображение в C/C++, SmallTalk, Ada95, Cobol и Java.

По числу поддерживаемых языков COM уступает CORBA, в которой предусмотрен гибкий механизм отображения описаний интерфейсов в терминах IDL в соответствующие структуры того или иного языка программирования. Стандартно в COM поддерживаются такие языки программирования как Visual C++, Java, Visual Basic, Delphi, Borland C++ Builder и т.д., но в рамках ОС Windows множества языков программирования, поддерживаемых данными технологиями, практически совпадают.

Обе технологии имеют солидный багаж проектов на их основе. Примеры конкретных реализаций систем на базе CORBA группируются по отраслям промышленности, и их список очень внушителен - банковское дело и финансы, реклама и маркетинг и другие. COM также может похвастаться значительным числом инсталляций. До недавнего времени область ее использования ограничивалась преимущественно настольными системами и сетями масштаба рабочей группы или

подразделения. Подобные COM-приложения доказали свою надежность и эффективность. Без Windows-систем сейчас не обходится большинство предприятий. С ростом производительности современных ПК расширяется и область применения Windows, поэтому COM/DCOM неизбежно будет важным элементом корпоративных архитектур.

CORBA обеспечивает реальную многоплатформенность. Реализации CORBA многочисленны и принадлежат множеству производителей. Эти продукты поддерживают обширный диапазон аппаратных платформ, в том числе мэйнфреймы и Unix-системы. Однако разнообразие реализаций имеет и свои недостатки, прежде всего потенциальную проблему несовместимости.

Скорость передачи данных при использовании данных технологий несколько ниже, чем при работе с файлами внутренней структуры, т.к. она ведется в режиме реального времени, но зато этот способ более эффективен с точки зрения универсализации стыковки с разными версиями одного и того же программного продукта, так как он обращается к единому набору функций и методов, которые прописываются в системном реестре Windows или, в случае CORBA, описаны в брокере объектных запросов и мало меняются от версии к версии.

Следует отметить, что посредством данных технологий возможен обмен данными в автоматическом режиме, т.е. без привлечения человека.

При работе по COM-технологии, предпочтительно использование продуктов пакета Visual Studio, так как эта технология разрабатывалась компанией Microsoft. Хотя COM по числу поддерживаемых платформ отстает от CORBA, на платформе Windows множества языков программирования, поддерживаемых обеими технологиями, практически совпадают и в этом случае более рациональным представляется использование технологии COM, интегрированной с Windows. Не следует забывать, что COM, в отличие от CORBA, является официально бесплатной технологией. Для взаимодействия по технологии CORBA с системами, работающими на платформах, отличных от Windows, можно использовать программные продукты фирмы Borland, входящей в консорциум.

Таким образом, предусмотрев при создании внешнего модуля средства передачи данных посредством COM- и CORBA-технологий, можно обеспечить проектировщику возможность достаточно быстрого построения моделей в большей части CAD/CAM/CAE-систем, и как следствие, их функциям анализа (рис. 7). При этом, за счет использования нейтральных форматов обмена данными, можно обеспечить следующий принцип взаимодействия с системами – на первом этапе обеспечивается взаимодействие с системами, обеспечивающими в совокупности наибольший набор функциональных возможностей (системы CAD 1 и CAD 2 на рисунке), тогда как для проведения вспомогательного анализа можно осуществить передачу данных в другие системы посредством файлов нейтрального формата (CAD 3, CAD 4, …, CAD n). Кооперативный, синергетический эффект, связанный с созданием единой информационно-технологической среды должен существенно повысить экономический эффект от использования систем.

Рис. 7. Схема итерационной передачи данных о моделях

Резюме. Осуществлен анализ форматов, позволяющих осуществлять обмен данными между CAD/CAM/CAE-системами. Использование специализированных конверторов является наиболее предпочтительным. Определены приоритеты использования технологий при передаче данных в CAD/CAM/CAE-системы из внешних приложений. Наиболее эффективным является взаимодействие посредством технологий COM и CORBA.

Список литературы:

1. *Куприков М.Ю.* Автоматизация проектно-конструкторских работ – фундаментальный фактор обеспечения качества жизненного цикла изделий в машиностроении // «Новые информационные технологии» Тезисы докладов X юбилейной Международной студенческой школы-семинара. 2002 г. - Москва. - с. 48-53.

2. *Lee K.* Principles of CAD/CAM/CAE Systems. 1999 г. – Addison-Wesley, – 560 с.

3. *Кенту М.* Delphi 7: Для профессионалов. 2004 г. – СПб. – 1101 с.

Сведения об авторах

Буряков Александр Александрович, аспирант кафедры «Инженерная графика» Московского авиационного института (государственного технического университета), научный сотрудник ГУ НПО «Спецтехника и связь»; Телефон: (095)415-80-48 (дом.), (095)673-90-50 (раб.); *e-mail*: alex_twims@rambler.ru