Организация взаимодействия независимо разработанных приложений в расширяемых графических САПР

Малюх В.Н., Никитин А.Г., Виноградов А.В., Канюс С.С.
Институт Систем Информатики СО РАН, Новосибирск, Россия
vmalukh@propro.ru, nick@complex-a5.ru, vini@propro.ru, staskan@gmail.com

Аннотация

Работа посвящена разработке архитектуры базовой САПР, обеспечивающей автоматическое взаимодействие пактов прикладных расширений с заранее неизвестной структурой данных и функциональным поведением. В качестве метода решения задачи предлагается в интерфейсе прикладного программирования САПР предоставить разработчикам приложений возможность программно переопределять в прикладных модулях программные методы и классы, реализующие геометрическую и функциональную модель базовой САПР. Также рассматриваются ряд архитектурных решений упрощающих программную реализацию такой системы.

***Ключевые слова**: САПР, API, прикладные пакеты, функциональная модель, архитектура системы.*

1. ВВЕДЕНИЕ

Одной из характерных тенденций развития САПР в последнее десятилетие является создание специализированных прикладных пакетов на основе универсальных САПР общего назначения. К настоящему времени созданы и находятся в эксплуатации десятки специализированных систем, однако архитектура, структуры данных и методы организация взаимодействия данных, традиционно применяемые в них, имеют ряд узких мест.

В частности автоматизация каждой новой прикладной задачи, как правило, решается путем разработки специализированных модулей, для работы которых требуются структуры данных, не предусмотренные для обработки ядром базовой системы, вследствие чего возникают коллизии в использовании ранее или параллельно разработанных модулей. Проблема усугубляется тем, что на этапе разработки базовой САПР достоверно не известны все задачи, для которых она будет применяться в будущем. Крайне затруднено, зачастую невозможно, взаимодействие прикладных систем, созданных независимыми разработчиками на базе одного и того же базового ядра.

Целью настоящей работы является создание архитектуры инструментального ядра САПР и выбор базовых алгоритмических решений, обеспечивающих гибкое взаимодействие независимо разработанных приложений, ориентированных на использование в расширяемых прикладных системах автоматизированного проектирования..

2. ОБЗОР ОГРАНИЧЕНИЙ СУЩЕСТВУЮЩИХ АРХИТЕКТУР

Наиболее распространенной типовой схемой построения современной графической САПР является создание универсальной системы геометрического моделирования, обеспечивающей создание и модификацию каркасных объектов, поверхностей и твердых тел, оснащения ее базовым набором встроенных инструментов и интерфейсом программирования, позволяющим комбинировать предопределенные геометрические сущности и создавать новые инструменты для их создания и модификации.

Наиболее характерным примером такой САПР является самый распространенный на текущий момент программный пакет AutoCAD, разрабатываемый компанией Autodesk [1] с 1981 года. Система построена по классической архитектуре, обладает весьма развитыми встроенными средствами моделирования геометрии, включая каркасные объекты, поверхности и твердые тела. Для разработчиков прикладных систем имеются интерфейсы программирования (API) для различных систем программирования. Эти API позволяют решать достаточно широкий круг задач: настраивать пользовательский интерфейс, создавать новые инструменты для редактирования моделей, модификации предопределенных базовой системой геометрических сущностей. Однако у этой архитектуры есть ряд существенных ограничений, связанных с созданием и редактированием новых геометрических сущностей с отличным от предопределенного базовой системой поведением.

Система AutoCAD разработана достаточно давно и развивалась путем последовательного добавления функциональности с использованием того же самого API, кроме того, значительное внимание уделялось сохранению функций предыдущих версий для обеспечения преемственности с ранними реализациями. Таким образом система всегда сохраняла архитектурные решения, свойственные ее ранним версиям. Как следствие, имеется недостаточная взаимосвязь между подсистемами каркасного черчения, поверхностного и твердотельного моделирования. Например, отсутствуют возможности взаимодействия между трехмерными моделями, построенными с использованием поверхностей и твердых тел, так как объекты этих типов имеют совершенно независимые описания геометрических данных и, как следствие не взаимодействуют друг с другом. Это фактически делает невозможным и наложение параметрических связей между контурными объектами, поверхностями и твердыми телами, тем самым сильно ограничивая функциональные возможности системы.

Каждая специфическая прикладная задача, как правило, решается путем разработки специализированных модулей, для работы которых требуются структуры данных, не предусмотренные для обработки ядром системы. API дает возможность привязки дополнительных данных к существующим типам объектов и их сборок, однако не обеспечивает обратной связи с этими данными для встроенных инструментов. Таким образом, во-первых, даже для выполнения простых геометрических преобразований разработчики сложных прикладных пакетов должны создавать практически заново весь набор средств редактирования, что заметно увеличивает трудоемкость и стоимость разработки. Во-вторых, необходимо обеспечивать защиту от применения к вновь созданным сущностям

встроенных инструментов редактирования, иначе пользователи сталкиваются с тем обстоятельством, что редактируя прикладные объекты привычными встроенными инструментами, они нарушают тем самым связь между геометрической формой и специфическими прикладными данными и, тем самым делают невозможной дальнейшую корректную работу с проектом с использованием специальных инструментов.

Вследствие этого возникают коллизии в использовании параллельно разработанных прикладных модулей. Так, например, используя пакет приложений Mechanical Desktop (MDT), разработанный самой Autodesk, можно создавать весьма сложные машиностроительные проекты, однако объекты, полученные с помощью МДТ несовместимы с твердыми телами, созданными средствами базовой системы. Аналогичная несовместимость имеет место между проектами, созданными в MDT и в других пакетах приложений, например Architectural Desktop (ADT), что вызывает значительные трудности в их совместном использовании, например при комплексном проектировании промышленных сооружений и наполнении их моделями производственного оборудования. Также, несмотря на использование схожего описания геометрии твердых тел, системы параметрических связей в каждом комплексе приложений работают независимо и не могут взаимодействовать. Проблема усугубляется тем, что на этапе разработки как базовой САПР, так и специализированного прикладного пакета на ее базе, заранее достоверно не известен весь круг задач, для которых они будут применяться в практике и, соответственно не определен набор применяемых совместно приложений.

В качестве характерного примера САПР следующего поколения рассмотрим пакет SolidWorks [2], впервые выпущенный одноименной компанией (с 2003 г – подразделение Dassault Systems) в 1996 году. Система SolidWorks изначально построена на базе универсального параметрического геометрического ядра Parasolid, которое обеспечивает мощные средства трехмерного твердотельного моделирования. Таким образом в SolidWorks радикально решена проблема взаимодействия геометрических описаний объектов и наложения параметрических связей как между самими объектами, так и между их отдельными элементами. Аналогично другим современным САПР SolidWorks имеет развитой API, обеспечивающий разработку приложений. Благодаря наличию встроенного в геометрическое ядро механизма обработки ограничений SolidWorks обеспечивает создание сложных параметрических моделей. Однако, оборотной стороной применения универсального геометрического решателя в качестве средства создания поведенческих связей в модели является то, что система весьма требовательна к вычислительны ресурсам а сам процесс наложения связей является достаточно трудоемким для конечного пользователя.

Также, возможности ядра Parasolid и API системы не предусматривают автоматического изменения топологии объектов и сборок, эта задача решается либо интерактивно (т.е. редактированием модели вручную) либо путем использования специализированных приложений, которые также активируются пользователем. Взаимосвязь между прикладными объектами, разработка структур данных которых производилась независимо, также ограничена и обеспечивается лишь на уровне взаимодействия аппроксимаций поверхностей и твердых тел, без учета специфических прикладных особенностей. Кроме того, для разработчика прикладным модулей остаются ограниченными возможности по способу отображения новых сущностей. По сути они сводятся к композиции отображений объектов, предопределенных геометрическим ядром.

Принципиально иной подход к построению архитектуры САПР используется в узкоспециализированных системах, предназначенных для решения только заранее предопределенного круга задач. Характерным примером такой САПР является пакет ArchiCAD [3], предназначенный для выполнения архитектурно-строительного проектирования. В ArchiCAD исходно все объекты, составляющие модель проектируемого изделия, суть прикладные сущности – стены, перекрытия, кровля, окна, двери и т.п. Система геометрических, топологических и функциональных взаимосвязей различных объектов спроектирована заранее, на этапе формирования структуры данных всей системы. Это обеспечивает весьма эффективное функциональное взаимодействие объектов и значительно упрощает работу конечного пользователя, работающего с естественными для него прикладными сущностями. Однако такое решение, несмотря на имеющийся развитой API фактически делает невозможным создание объектов с поведением, принципиально отличающимся от того, что заложено в базовую систему. Это обстоятельство значительно затрудняет решение смежных задач проектирования (например прокладку инженерных коммуникаций) и сопряжение проектов, созданных в ArchiCAD и ему подобных системах с другими САПР. Также остаются ограниченными возможности по изменению способа визуального отображения проектируемых объектов.

Следует отметить, что принципы, положенные в основу существующих САПР во многом предопределены технологией программирования больших систем, имевшей место на рубеже конца 80-х начала 90-х, а именно так называемой компонентной парадигмы расширения систем.. Разработка ПО САПР весьма трудоемка и изменение архитектуры сопряжено со значительным затратами, вследствие этого развитие существующих САПР в течение значительного промежутка времени осуществляется эволюционным путем, не затрагивая базовых концепций. В то же время, в сфере технологий разработки ПО за последнее десятилетие произошло внедрение ряда кардинальных нововведений, результатом которых стали существенные, принципиально отличные возможности по формированию структурного облика сложных программных систем. В первую очередь это относится к достижениям в области технологии объектно-ориентированного программирования, в частности с появлением и широким распространением платформы .NET [4], разработанной корпорацией Microsoft.

3. АРХИТЕКТУРА ПЕРСПЕКТИВНОЙ БАЗОВОЙ САПР

Основываясь на всем вышесказанном, представляется целесообразной разработка на новой технологической основе архитектуры базовой САПР, обеспечивающей организацию автоматического взаимодействия независимо разрабатываемых приложений.

Основная идея, предлагаемая в работе заключается в разработке архитектуры базовой САПР [5], обеспечивающей инкапсуляцию геометрических и функциональных описаний

объектов, методов хранения и обеспечения функционального взаимодействия прикладных объектов в единой среде за счет следующих основополагающих решений:

- Средства инкапсуляция в проектные данные геометрических, литеральных и функциональных описаний объектов есть базовая сущность системы.
- Наборы базовых геометрических представлений и используемых алгоритмов сознательно минимизированы и унифицированы.
- Разработчикам приложений предоставляется возможность программно переопределять в прикладных модулях методы и классы базовой САПР, реализующие геометрическую и функциональную модель.

3.1 Унификация геометрического представления объектов.

В описываемой архитектуре САПР предлагается использовать максимально унифицированное описание геометрии объектов с одной стороны, и изначально накладывающее как можно меньше ограничений на возможные варианты описания данных с другой. Это сознательное "упрощение" ограничивает функциональные возможности базовой САПР, но позволяет пополнять ее сколь угодно сложной функциональностью за счет разработки приложений.

В качестве универсального базового элемента геометрии, взамен традиционного набора специализированных сущностей используется ***пространственный композитный контур***, представляющий собой сложный цельный объект, состоящий из произвольной последовательности прямых, эллиптических дуговых и сплайновых сегментов (NURBS разного порядка). Сегменты, следующие друг за другом, имеют одну и только одну общую точку. Контур может иметь свойство замкнутости. Такое представление достаточно традиционное для САПР, в описываемой реализации дополнено существенным расширением: в узлах контура могут быть определены условия сопряжения сегментов – угол и радиус.

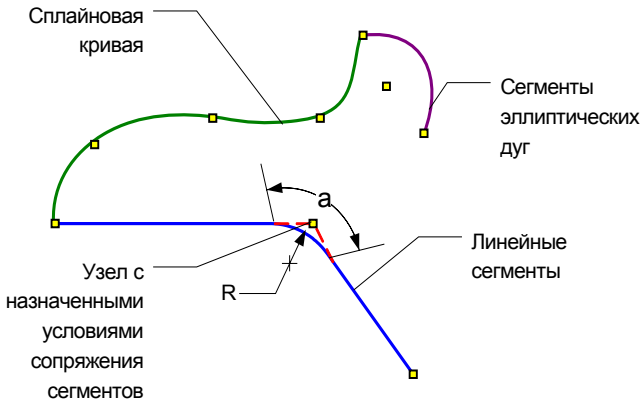


Рис. 1. Композитный контур

На базе такого универсального описания формируются частные случаи контурных объектов, как двумерных таки пространственных – окружности, эллипсы, прямоугольники, правильные многоугольники и т.п.

В качестве описания **поверхностей** выбран метод *кинематического описания*, заключающийся в том, что поверхности определяются набором образующих и направляющих контуров и способом их применения. В структуре данных, описывающих поверхность при таком описании, в отличие от традиционно принятых методов, разделены контекст построения, используемый для хранения и модификации информации о поверхности и ее аппроксимация, используемая для визуализации и, частично, для промежуточных вычислений. В качестве унифицированного набора избраны поверхности четырех основных кинематических классов (рис 2): поверхности протяжки (поз a), поверхности Кунса (поз b) поверхности вращения (поз. c) и плоские поверхности (поз d). Существенным обстоятельством, предопределившим выбор унифицированных классов поверхностей, является то, что поверхности одного класса могут быть преобразованы без искажения формы в поверхности или композицию из поверхностей других классов. Способ алгоритмической аппроксимации поверхностей вынесен за рамки инструментального ядра и может быть выбран разработчиком приложений. На практике наиболее часто используются либо полигональная аппроксимация плоскими многоугольниками либо NURBS.

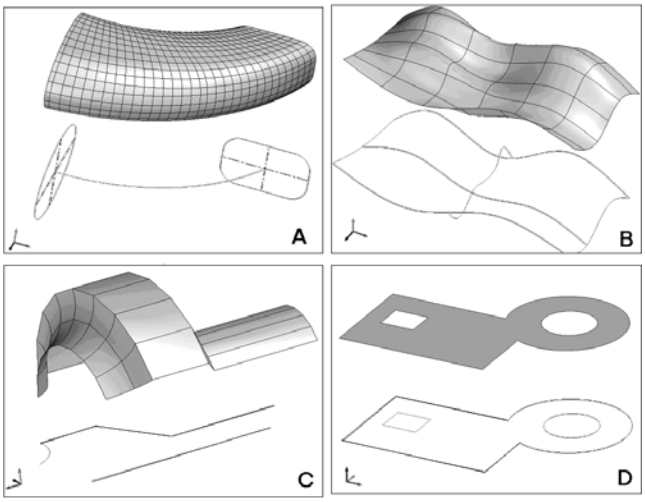


Рис. 2. Классы поверхностей

Твердые тела описываются в рамках представляемой архитектуры комбинацией из граничного представления элементарных односвязных тел, т.н. BREP (Boundary REPresentation) и последовательностью булевых операций над ними, т.н. CSG-tree (Constrictive Solid Geometry Tree). Суть BREP-представления заключается в том, что твердое тело описывается замкнутая пространственная область, ограниченная набором элементарных бесконечно тонких поверхностей (граней), с общими образующими контурами (ребрами) на границе поверхностей и признаком внешней или внутренней стороны поверхности, а также следующим рядом операций, определенных над телами:

Для описания сложных тел, получаемых, например, обработкой материала или неразъемной сборкой используется иерархическая структура, описывающая последовательность применения булевых операций над набором элементарных твердых тел – так называемое CSG-дерево (Constructive Solid Geometry tree). Для составных твердых тел определены операции, дополняющие набор операций, обязательных для элементарных тел: вычитание, объединение, пересечение.

3.2 Универсальная логическая структура проектных данных

Традиционно в САПР используется одна из двух систем группирования и разделения объектов проекта. Первая, с использованием т.н. слоев сложилась исторически по аналогии с чертежами, элементы которых вычерчены на нескольких листах прозрачного носителя. Такой подход позволяет эффективно и интуитивно ясно управлять видимостью элементов в сложных проектах. Следует оговорить, что под видимостью полагается понятие более широкое, нежели просто отображение объекта на носителе информации, а именно – набор динамически настраиваемых групповых прав доступа для произвольного набора объектов.

Второй подход представляет собой помещение объектов в иерархическое дерево, аналогично тому, как это устроено в файловой системе с директориями и файлами. Такой подход чрезвычайно удобен для описания сборок, особенно в случае использования внешних ссылок на элементы сборок, описанные в отдельных проектах.

В полной мере достоинства обоих методов сочетает, предлагаемый в настоящей работе комплексный подход, в котором в качестве основной структуры используется сборочная иерархия, а в качестве способа управления видимостью частей проекта как документа - модифицированная иерархия слоев. Суть модификации послойного подхода заключается в том, что любой из объектов сборочной иерархии (как геометрический, литеральный, так и собственно узел дерева) может ссылаться не на один слой, а на произвольное количество слоев одновременно. Таким образом, расширяется количество возможных комбинаций выставляемых признаков видимости объектов.

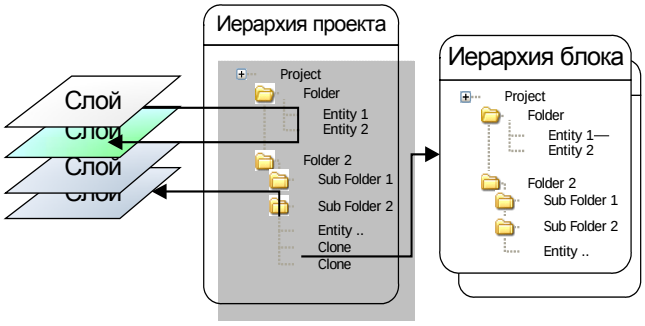


Рис. 3. Логическая структура проекта

В результате внешние ссылки в предлагаемой архитектуре реализуются достаточно традиционным способом, путем сохранения в объекте вставки ссылки на расположение внешнего проекта с типовой внутренней структурой и, в целях достижения переносимости проектов и быстродействия базы данных проекта, кэширования экземпляра содержимого блока.

3.2 Организация и система управления прикладными данными в моделях объектов.

3.2.1 Инкапсуляция произвольных прикладных данных

Для создания прикладных систем ключевым моментом является наличие возможности расширять структуру базы данных проекта и их взаимодействие, насыщая ее прикладной информацией, заранее (на этапе разработки инструментальной системы) неизвестной структуры и поведения. В качестве решения этой задачи предлагается присоединять к стандартным объектам головной системы контейнеры, представляющие собой обычную байтовую последовательность произвольного размера, снабженную идентификатором. С этой точки зрения он подобен файлу на диске. Роль головной системы сводится к автоматическому сохранению контейнеров при записи и чтении проекта из файла, копировании и переносе объектов внутри проекта и при обмене между проектами.

Количество контейнеров, прикрепленных к одному объекту неограниченно. Последний вариант упрощает задачу идентификации контейнеров, сводя ее к простой индексации, и тем самым, улучшая производительность системы в плане поиска и доступа к данным. Однако, в этом случае становится невозможно гарантировать отсутствие дублирования доступа к прикладным данным со стороны независимо разрабатываемых прикладных систем. Для предотвращения использования контейнеров с данными другими приложениями каждый контейнер предлагается снабдить уникальным идентификатором, создаваемым инструментальными средствами. Для создания идентификаторов целесообразно использовать технологию GUID (Global Unique Identifier), гарантирующая всемирную уникальность 128-разрядных идентификаторов. В простейшем случае применение вышеописанного метода внедрения прикладных данных в базу данных проекта позволяют создать *пассивную* расширяемую систему, в которой вся обработка прикладных данных производится путем исполнения прикладных программ, лишь функции хранения данных приложений обеспечиваются головной системой.

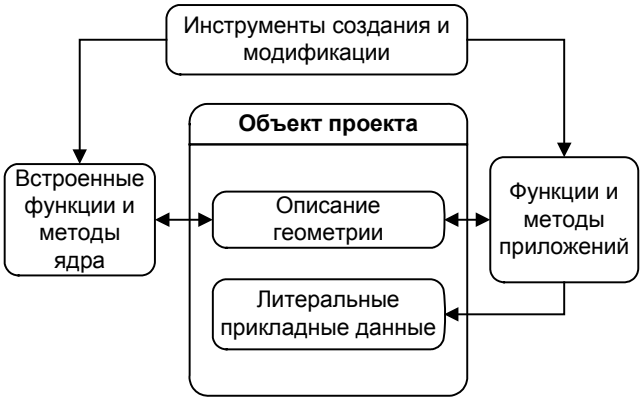


Рис. 4. Структура работы с прикладными данными в пассивно расширяемой системе

Расширение за счет инкапсуляции пассивных данных позволяет при небольших трудозатратах на разработку решать часть задач, в первую очередь связанных с постобработкой данных проекта, хранение в проекте специализированных прикладных данных, не предусмотренных разработчиками базовой САПР.

3.2.2 Инкапсуляция геометрических трансформируемых данных

В ряде случаев необходимо модифицировать геометрические данные приложений синхронно с редактирование объектов встроенными средствами и инструментами головной системы. Наиболее типичным случаем таких модификаций

являются аффинные геометрические преобразования: смещение, поворот, масштабирование или зеркальное или осесимметричное отражение. В практике трехмерной графики и САПР, применяемые к объектам геометрические трансформации выражаются обобщенной матрицей преобразований в однородных координатах, содержащей в своих элементах результат смещение, поворот и масштабирование.

Для обеспечения синхронизации геометрических трансформаций предлагается кроме независимых контейнеров, данные в которых не изменяются при модификации объектов встроенными средствами головной системы предусмотреть также контейнеры специального типа, содержащие структурированную геометрическую информацию в виде последовательности трехмерных координат опорных точек [6].

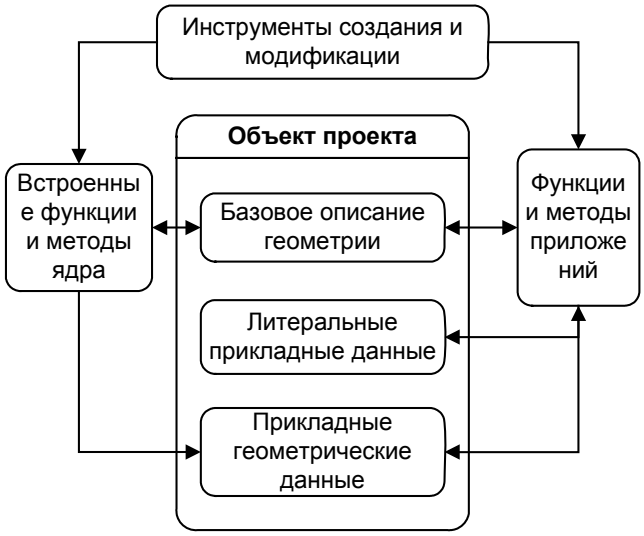


Рис. 5. Структура работы с прикладными данными в полуактивной расширяемой системе

Так как встроенные данные головной системы модифицируются аналогичным образом то прилагать матрицу трансформаций к контейнерам с прикладной геометрией достаточно автоматически при каждом применении общесистемной трансформации. Таким образом, изменения геометрической информации, описанной в прикладных данных, гарантированно происходят синхронно с преобразованиями геометрии основных объектов. Применение этого метода внедрения прикладных данных в базу данных проекта позволяют создать *полуактивную* расширяемую систему, в которой вся специфическая обработка литеральных прикладных данных, связанных с целевым назначением производится путем исполнения прикладных программ а геометрические трансформации – могут быть выполнены автоматически головной системой.

3.2.3 Активные компоненты функциональной модели

Описанных выше ограничений лишен предлагаемый в настоящей работе способ расширения данных, который заключается не только в хранении в объекте прикладных данных, но и таблицы переопределенных базовых методов, автоматически вызываемых системой при совершении операций над объектом. В том случае, если при разработке прикладного объекта метод не переопределен, базовая система использует встроенные методы модификации, аналогично способам описанным ранее.

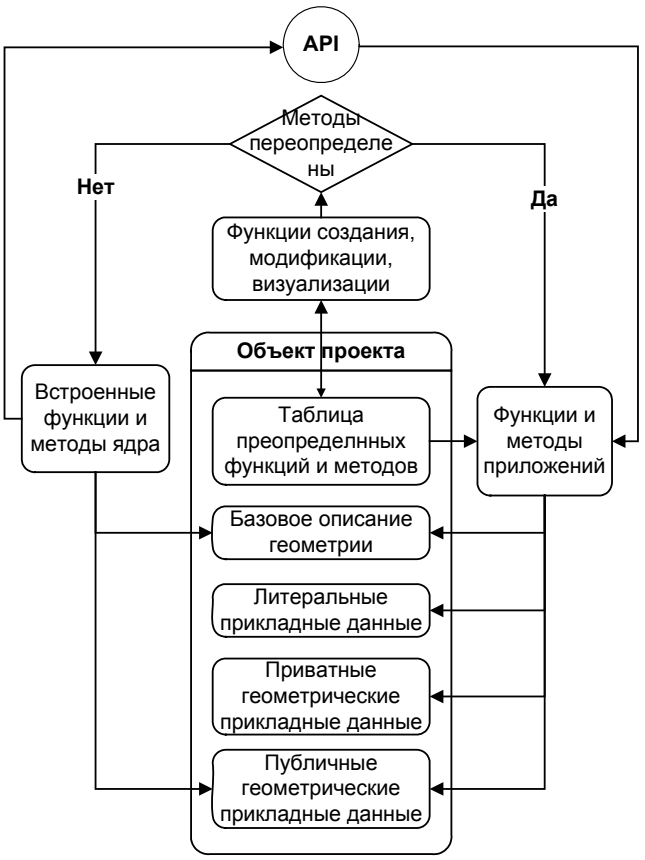


Рис. 6. Структура работы с прикладными данными в активной расширяемой системе

Таким образом, при всяком интерактивном или программном изменении, вносимом в базу данных объекта, их перевычисление может производиться в соответствии с поведенческим алгоритмом, заложенным разработчиком приложений, что обеспечивает расширяемость функциональности базовой системы и обеспечивает корректную модификацию всех компонент объекта. Более того, в таблице подмены методов, реализующих объект, могут быть переопределены не только методы, изменяющие объект, но и, например, методы его отображения в зависимости от требуемого контекста – тем самым решаются вопросы генерализации и условных обозначений в ГИС и общеинженерных проектах, что принципиально расширяет возможности, предоставляемые разработчику прикладных систем.

Наличие активные компонент в архитектуре прикладных систем позволяют естественным образом сформировать не только геометрическую модель, но наборы литеральных объектов, описывающих ***ограничения и связи***, установленных между литералами и параметрами геометрических объектов[7]. Каждый экземпляр установленной связи также содержит интервал возможных значений параметра.

4. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

4.1 Общие требования к системе

На базе описанных выше архитектурных решений разработана инструментальная среда bCAD [8]. Проектирование и разработка программной реализации системы велись исходя из следующих общих технических требований к дизайну и внутренней организации программного обеспечения:

- независимое изменение исходных кодов и отладку частей системы несколькими программистами одновременно
- возможность совместного использования значительной части исходных кодов, как самой системой, так и какими-либо иными уже существующими или будущими программными системами.
- возможность расширения функционала системы, без каких-либо изменений исходных текстов базовой системы и соответственно перекомпиляции базового исполняемого модуля.
- возможность простой установки и регистрации новых приложений в уже установленную у пользователя систему (без переинсталляции системы).
- как система, так и расширения должны быть защищены от возможных повреждений своих внутренних данных или нештатного поведения, в результате их несовместимости.

Учитывая достаточно большие объемы данных, хранимые системой в оперативной памяти, и имеющиеся данные по способам работы пользователей с системой (по опыту предыдущей версии системы), категорически неприемлемо аварийное завершение работы системы с потерей всех текущих данных, например, при возникновении недостатка оперативной памяти для завершения текущей операции. Следовательно, реализация системы спроектирована таким образом, что бы максимально защитить пользователя не только от потери, но и от нарушения целостности его данных при возникновении *не фатальных* исключительных ситуаций.

Система проектировалась с использованием необходимой декомпозиции проекта в целом на набор *слабо связанных* библиотек. Для поддержания возможности расширения функционала и обеспечения гибкости конфигурирования системы под нужды различных категорий пользователей в системе объявлено несколько основных интерфейсов для ***системных*** и ***прикладных*** расширений.

Реализовано несколько таких расширений в основном ориентированных на добавление чтения/записи дополнительных форматов хранения изображений, импорта/экспорта объектной модели в различные графические форматы, а также реализации специализированных прикладных объектов. Реализация системных расширений в виде динамически подгружаемых библиотек (DLL) обеспечивает возможность независимой их разработки, отладки и распространения при условии неизменности интерфейса "система-расширение". Базовая библиотека поддержки интерфейса пользователя содержит механизм проверки на совместимость версий системы и расширения, основанный на проверке номеров версии и номеров сборки (build) системы и загружаемого расширения, и базовые механизмы настройки графических элементов управления под нужды пользователя.

4.2 Переопределяемые программные методы

В программной реализации предложенной архитектуры ряд базовых методов выполнены переопределяемыми при реализации объектов, не предусмотренных базовой системой. Список этих методов, в зависимости от реализации, может пополняться, не ограничиваться приведенным, на текущий момент выделены следующие:

- ***Generate*** – создание объекта по контексту построения или модификации
- ***Transform*** – реакция объекта на применение аффинных и квазиаффинных преобразований
- Verify – проверка объекта на корректность
- Save – запись объекта в файл
- Load – чтение объекта из файла
- Delete – удаление объекта из базы данных проекта
- Display – отображение объекта по требуемому контексту (на экране, принтере, в 3D, 2D и т.п.)
- Get/set properties – получение и назначение значений данных объекта.

Приведем пример, иллюстрирующий сложные изменения, затрагивающие не только геометрию объекта в процессе исполнения операции, но и обеспечивающие изменение топологии и создание новых объектов, входящих в сборку. При масштабировании пространственной ферменной конструкции (изменении ее длины) необходимо произвести не простое геометрическое увеличение, но и изменение количества поперечных элементов в соответствии с предопределенными инженерными правилами (рис 7), сохраняя их поперечный размер неизменным, что невозможно при простом применении матрицы аффинных преобразований.

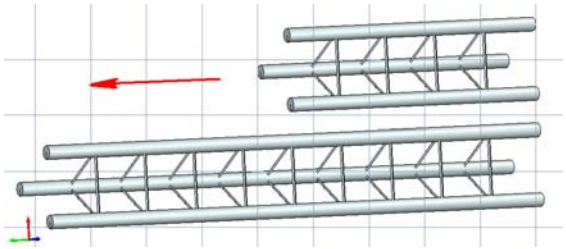
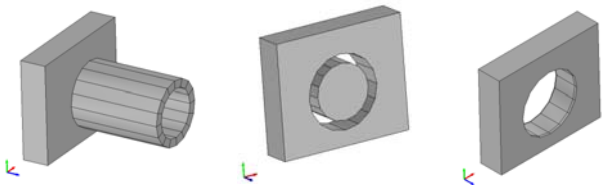


Рис. 7. Перестроение топологии объекта при геометрическом масштабировании

- ***Boolean*** – реакция объекта на применение его в булевых операциях

В качестве иллюстрации переопределения метода булевой операции вычитания для случая создания отверстия в преграде, геометрически не тождественного геометрии вырезаемого объекта. Характерным примером такой ситуации является проем в перегородке, необходимый для прохождения через него трубопровода (рис. 8). Геометрия трубы описывается телом, имеющим полость, поэтому в результате прямого применения исходного тела в качестве булева операнда в результате получается вырез, некорректный с точки зрения конструкторских требований (рис. 8 б). В переопределенном методе Boolean для выполнения операции вычитания автоматически создается временное тело, представляющее собой эквидистанту к внешней поверхности трубы, и это тело предается в булеву

операцию. Результатом является полное отверстие с необходимым зазором (рис 8 в).



а) б) в)

Рис. 8. Булева операция вычитания с исходным (б) и переопределенным (б) операндом

Показательным примером переопределения метода Display служит отображение специализированных объектов в схематическом виде. На рис. 9 приведено изображение проекта системы трубопроводов в макетном виде и в виде его сборочной иерархии с использованием условных обозначений элементов, предусмотренных стандартом.

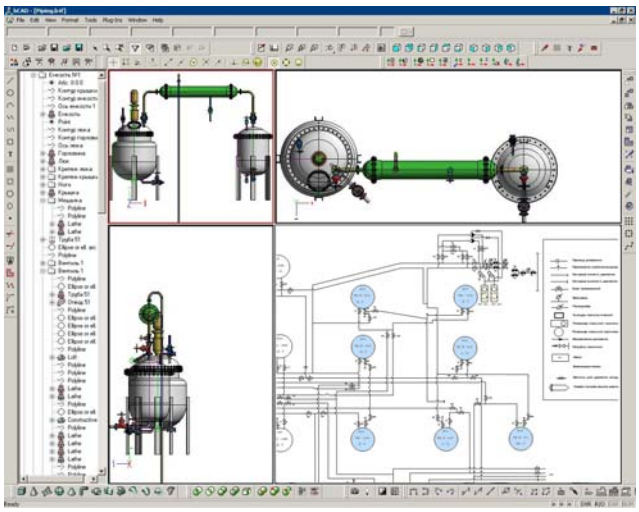


Рис. 9. Фрагмент сложной трубопроводной системы

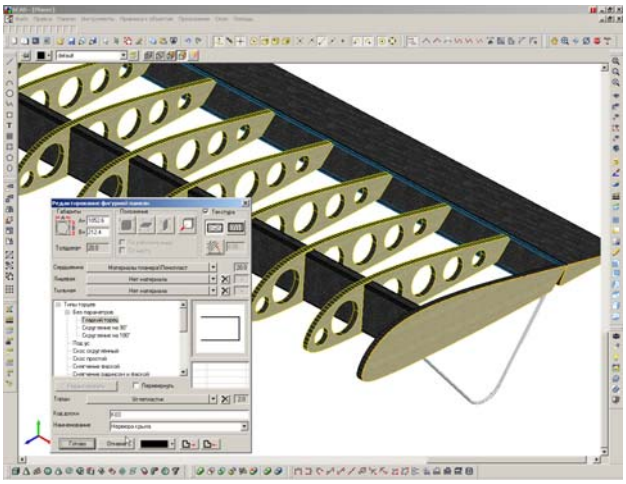


Рис. 10. Специализированный диалог редактирования свойств гнутой листовой детали

В отличие от традиционного подхода как объемное, так и схематическое изображение есть представление самого объекта, а не производный документ, получаемый методом сложной постобработки.

На рис.10 приведен пример автоматического вызова специализированного диалога редактирования свойств (Properties) для сложного конструктивного элемента.

5. ЗАКЛЮЧЕНИЕ

На базе предлагаемой архитектуры и организации модели данных разработана универсальная САПР bCAD и на ее основе ряда прикладных систем для создания проектов архитектурных сооружений, конструирования мебели и торгового оборудования, моделирования кинематики, сложных трубопроводных коммуникаций, выполнения проектно-компоновочных работ в авиации и судостроении. Опыт разработки, внедрения и эксплуатации этих систем, на практике подтверждает эффективность описанных архитектурных решений.

6. БИБЛИОГРАФИЯ

1. MacAuley C. Programming AutoCAD in ARX. Autodesk. 2000.
2. Мюррей Д., SolidWorks. М. Изд-во «Лори», 2003
3. Титов С. ArchiCAD 9. Справочник с примерами. М. Изд-во «КУДИЦ-Образ», 2005
4. Box D., Sells C. Essential .NET. Volume 1. The Common Language Runtime. Microsoft .NET Development Series, 2002.
5. Бахтин И.Н., Малюх В.Н., Архитектура интерфейса прикладного программирования в САПР bCAD. "САПР и Графика" М. Изд-во Компьютер Пресс, №3, 2000, с 20-26.
6. Малюх В.Н. Разработка специализированных приложений в САПР bCAD. «Автоматизация проектирования», М., изд. РАН, № 2, 2000 с15-18,
7. Malukh V.N. Intelligent capabilities for small CAD systems: problems and perspectives. Сборник докладов на международной конференции isicad-2004, Новосибирск, июнь 2004, с. 311-312.
8. Malukh V.N., Nickitin A.G. Modern Architecture of lightweight CAD. Сборник докладов на международной конференции Graphicon -2005, Новосибирск, июнь 2005, с. 111-113.

Об авторах

Владимир Николаевич Малюх – директор ЗАО «ПроПро Группа», м.н.с. ИСИ СО РАН

Адрес: 630090, Новсибирск-90, а/я 346

e-mail vmalukh@propro.ru

Алексей Германович Никитин – ведущий разработчик ООО «Комплекс 5», м.н.с. ИСИ СО РАН.

e-mail nick@complex-a5.ru

Алексей Валерьевич Виноградов – ведущий сотрудник ЗАО «ПроПро Группа», ст. преподаватель НГТУ.

e-mail vini@propro.ru

Станислав Сергеевич Канюс – программист «Интел», аспирант ИСИ СО РАН.

e-mail staskan@gmail.com

Third-party application packages interaction in extendable graphics CAD systems

Abstract

Work is describing architecture of basic CAD system, providing automatic interaction between third-party applications, developed independently. To provide system with such ability it is suggested that third-party developers are allowed to redefine basic program methods and classes implementing CAD project objects. Also, there are few minor architecture solutions, simplifying such architecture implementation. There are several samples if real implementations described.

Keywords: *CAD, 3D, 2D, third-party applications, plug-ins, functional model, system architecture*.

About the authors

Vladimir Malukh, CEO ProPro Group, research fellow of Institute of Informatics System of Russian Academy of Science, Siberian Branch, his e-mail is vmalukh@propro.ru

Alexey Nickitin senior developer of "Complex-5" company, research fellow of Institute of Informatics System of Russian Academy of Science, Siberian Branch, his e-mail is nick@complex-a5.ru

Alexey Vinogradov senior architect of ProPro Group, senior lecturer of Novosibirsk State technical university, his e-mail is vini@propro.ru

Stanislav Kanus, Intel developer, post-graduate of Institute of Informatics System of Russian Academy of Science, Siberian Branch his e-mail is staskan@gmail.com