УДК 658.512.011.056

А.В. Булыгин, В.Ю. Гранин,
О.С. Гурова, М.Л. Тюрина

Инструментальные средства компьютеризации инженерных знаний в САПР

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ»

Ключевые слова: интеллектуальная система проектирования, метамодель, база знаний, метод базы знаний, модуль инженерных знаний, семантическая сеть.
Ключові слова: інтелектуальна система проектування, метамодель, база знань, метод бази знань, модуль інженерних знань, семантична мережа.
Keywords: intellectual design system, knowledgebase, knowledgebase method, knowledge engineering module, semantic network.

В первой статье нашего цикла [1] были рассмотрены причины недостаточной эффективности современных машиностроительных САПР, в числе которых отмечены недостаточный уровень формализации методик проектирования и низкая производительность традиционных технологий разработки программного обеспечения САПР. Эффективным способом решения указанных проблем признаны средства инженерии знаний. САПР должна способствовать решению проектных задач, сформулированных в терминах предметной области, в том числе и не имеющих заранее известного алгоритма их решения. Такие задачи традиционно относятся к числу интеллектуальных, в связи с чем требование обычно называют принципом интеллектуализации САПР.

К настоящему времени наиболее мощные CAE/CAD/CAM-системы оснащены модулями работы с инженерными знаниями, например, Unigraphics Knowledge Fusion (современное название Design Logic) и CATIA Knowledgeware, известны убедительные примеры их эффективного применения при проектировании самолетов Су и Ан. Среди систем проектирования среднего уровня до недавнего времени примерами могли служить лишь российские разработки T/FLEX САПР ИМ (САПР изделий машиностроения) и интегрированная инструментальная среда СПРУТ (Система проектирования универсальной технологии) с подсистемой генерации баз знаний СПРУТ-БЗ.

Однако в течение последних двух-трех лет, как отмечено в работе [2], становится заметным все более широкое использование такого инструментария в системах проектирования среднего уровня (mid-range). Важная особенность новых средств компьютеризации знаний состоит в том, что они, в отличие от упомянутых выше баз знаний предшествующего поколения, обычно реализуются как CAD-независимые системы. В той же статье приведено мнение одного из пользователей знание-ориентированных технологий (Knowledge-Based Engineering, KBE), работника аэрокосмической промышленности США: «Для деталей сборок, при проектировании которых применима KBE-технология, MCAD-система не является центральным звеном процесса, она просто обеспечивает вывод информации. Меня крайне огорчает тесная интеграция KnowledgeWare с CATIA V5 и Knowledge Fusion с

NX. Поэтому я предпочитаю CAD-независимое приложение Rule Stream, которое не сковывает меня в выборе CAD-системы». Эта тенденция также была принята авторами во внимание при разработке инструментария интеллектуальных САПР.

Однако ни одна из таких систем, за исключением лишь двух упомянутых российских разработок, не решает проблем сокращения трудоемкости разработки САПР и привлечения к их созданию непрограммирующих экспертов предметной области, поскольку разработка баз знаний сопровождается программированием на специальных языках. Практически единственным способом разрешения этого противоречия может служить оснащение САПР инструментальными средствами расширения функций, позволяющими непрограммирующему профессионалу пополнять систему новыми методами проектирования.

Цель настоящей работы состоит в рассмотрении принципов построения и основных функций модуля генерации баз знаний, входящего в состав интеллектуальной инструментальной среды разработки специализированных САПР. Под знаниями понимаются: система понятий предметной области и их связь с формальными представлениями; структура данных информационной модели предметной области; математические модели, используемые при проектировании; правила принятия проектных решений.

Как отмечено в [1], при создании комплекса инструментальных средств авторами было принято решение не разрабатывать собственные универсальные программные средства (СУБД, CAD- и CAM-модули и др.), а обеспечить совместимость создаваемого инструментария с готовыми программными средствами путем организации взаимодействия с внешними базами данных, CAD-системами и подключением внешних программ. При этом минимально достаточным был признан следующий состав инструментальных средств: модуль моделирования структуры изделия и проектных альтернатив, модуль генерации баз знаний и модуль взаимодействия с CAD-системами. С их использованием на основе создаваемых экспертом информационных моделей и баз инженерных знаний формируются прикладные программы модульной структуры, открытые для последующей модификации без привлечения профессиональных программистов.

Одним из важнейших инструментов создаваемого комплекса является модуль генерации баз знаний, предназначенный для выполнения следующих функций:

- взаимодействие с пользователем в терминах предметной области;
- формирование выводов на основе имеющихся знаний, представленных как в алгоритмической (процедурной), так и в экспертной (например, продукционной) форме;
- автоматический синтез алгоритмов и программ на основе сформулированной пользователем постановки задачи.

Заметим, что под выводом в данном случае понимают не обязательно только логический вывод, реализуемый экспертными системами. Как отмечено в работе [3], вывод на функциональной сети эквивалентен

построению расчетной цепочки из начальной вершины (исходных данных) в целевую (результат).

Следовательно, модуль объединяет функции таких компонент программного обеспечения интеллектуальной системы, как интеллектуальный интерфейс, решатель задач и интеллектуальная система программирования. Тенденция к такому объединению функций характерна и для инструментальных программных средств общего назначения – в качестве одного из перспективных направлений рассмотрено создание интеллектуального инструментария и переход от технологий CASE (Computer-Aided Software Engineering) к BASE (Brain-Assisted Software Engineering) [4].

Ближайшими аналогами разрабатываемого модуля, как уже отмечалось, были признаны подсистемы T/FLEX САПР ИМ и СПРУТ-БЗ. Несмотря на некоторые различия между ними, основные идеи и принципы построения этих систем можно считать общими, они достаточно полно изложены в работе [5]. Такие идеи были положены и в основу разработки нашего прототипа интеллектуальной инструментальной среды. Речь идет прежде всего о многоуровневой структуре базы знаний и основных типах модулей знаний. Анализ содержания типовых процессов проектирования показал, что возможностей продукционной модели представления знаний вполне достаточно для любых видов проектных операций (расчеты по формулам, извлечение данных из таблиц и др.). Вместе с тем было признано целесообразным дополнить проектные операции некоторыми процедурами управления процессом проектирования, прежде всего циклами.

Структура базы знаний. Первичной единицей знаний является модуль инженерных знаний (МИЗ). Он представляет собой функциональный блок, содержащий входы, выходы, управляющие реакции и механизм, который может быть представлен согласно стандарту IDEF0 схемой, изображенной на рис. 1. Входами и выходами служат данные, а механизмом – средства их преобразования (формулы, выборки из таблиц и базы данных и др.).

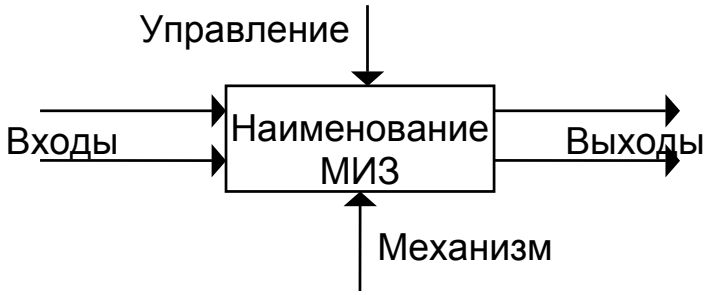

Рис. 1. Модуль инженерных знаний как типовой функциональный блок

Для решения комплексных задач, требующих использования нескольких функциональных блоков, элементарные блоки (МИЗ) объединяются по входам и выходам в сложную функцию – метод базы знаний. Маршруты проектирования технических объектов, т.е. последовательности проектных

операций, выраженные в виде модулей и методов базы знаний, как правило, нелинейны: решения, принимаемые в процессе проектирования, могут зависеть от тех или иных условий; могут использоваться итерационные расчетные схемы и др. Поэтому модули инженерных знаний объединяются в семантическую сеть. Поименованными узлами такой сети являются сами модули знаний, а ориентированными ребрами – входные и выходные переменные, наименования которых соответствуют свойствам проектируемого объекта.

Метод базы знаний позволяет определить методику проектирования некоторого класса объекта машиностроения. Для обеспечения возможности разработки методик решения связанных задач была введена ещё одна единица – событие. Определив реакции на события (в качестве реакций выступают другие методы, определенные в той же базе знаний), можно реализовать циклически взаимосвязанный расчет нескольких классов объектов, т.е. определить методику расчета связанной задачи. Наличие событий позволяет организовать любой другой событийный алгоритм произвольной сложности.

В автоматическом режиме создаются два события – начало и окончание работы расчетного метода. Реакции на эти события также определяются автоматически: реакция на событие начала расчета служит для построения объекта с параметрами по умолчанию в CAD-системе, а реакция на событие окончания метода позволяет запустить на выполнение расчётные методы подчинённых классов объектов.

Структура метода показана на рис. 2.

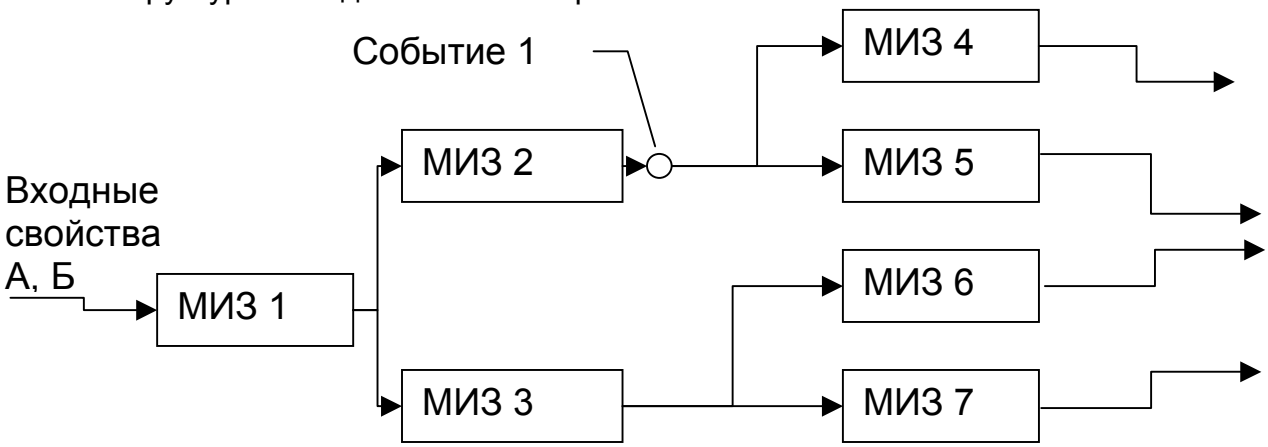

Рис. 2. Логическая структура метода базы знаний

База знаний (БЗ) представляет собой набор методов, между которыми могут существовать связи, например, один метод может вызываться из другого, а также метод может использоваться в качестве реакции на какое-нибудь событие.

Структура базы знаний изображена на рис. 3.

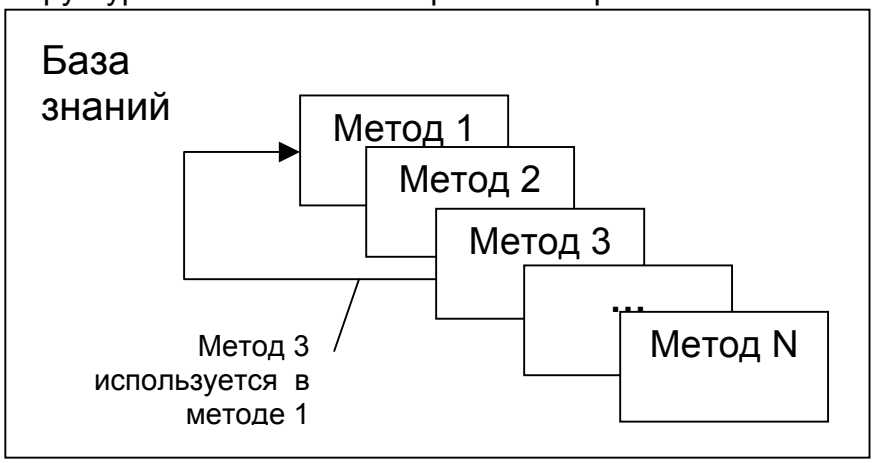

Рис 3. Структура базы знаний

Основное назначение модуля генерации баз знаний – поддержка разработки методов проектирования непрограммирующими специалистами, но если по каким-либо причинам более удобно традиционное написание исходного кода методов, то такая возможность также предусмотрена. В этом случае на основе информационной модели генерируется структура САПР и код служебных методов, необходимых любой САПР объектов машиностроения (например, метод, запрашивающий у пользователя свойства, которые не были определены в процессе расчета). Сгенерированную структуру можно наполнить кодом на языке C# в любом редакторе с использованием стандартных средств отладки.

Создание метода. Для каждого класса информационной модели может быть создано произвольное количество методов, но некоторые из них, являющиеся служебными, как отмечалось ранее, создаются в автоматическом режиме. Каждый метод использует внутренний набор свойств, множество таких свойств представлено в словаре метода. Таким образом, метод оперирует исключительно своим внутренним словарём, а взаимодействие с внешней средой, т.е. с классами определенного объекта машиностроения, которым данный метод принадлежит, осуществляется путём передачи свойств класса в качестве параметров.

Для разработки метода базы знаний пользователем инструментальной среды должны быть выполнены следующие действия:

1) создание словаря метода;
2) генерация модулей инженерных знаний;
3) определение входов и выходов метода;
4) автоматическая сборка метода, т.е. объединение МИЗов в семантическую сеть;
5) проверка корректности сборки метода, ручная доработка последовательности МИЗов;
6) добавление событий;
7) верификация корректности счёта метода, доработка метода;
8) подключение метода к классу информационной модели;
9) генерация кода метода.

Структура и логика работы метода рассмотрены далее в соответствии с этой последовательностью действий.

Словарь метода содержит перечень свойств, которыми может оперировать метод. Существует разделение свойств метода на входные, выходные и локальные (или внутренние). Входные и выходные свойства метода должны быть связаны со свойствами класса, к которому подключается метод. Типы подключаемых свойств метода и класса должны совпадать. Для каждого свойства задаются краткое имя, полное имя и тип свойства: целочисленный, вещественный, строковый, символьный, а также любые пользовательские типы. Для свойств строкового типа обязательно задание списка строк – значений, принимаемых этими свойствами. Такие списки допустимых значений могут быть заданы и для числовых свойств.

Виды модулей инженерных знаний. Чтобы максимально облегчить непрограммирующему специалисту разработку методов проектирования, необходимо предусмотреть достаточное разнообразие элементарных расчетных единиц. Ниже описаны виды модулей знаний, реализованные в системе.

МИЗ типа *Формула*. Модуль знаний этого типа позволяет получить значение свойства как результат вычисления некоторого математического выражения либо как результат применения пользовательских функций и операций над нечисловыми классами. Для проверки корректности записи формул используют синтаксический анализатор, который оценивает правильность формул с точки зрения синтаксиса.

В качестве операций и функций, применимых к свойствам, могут быть использованы:

- стандартные операции и функции – операции над стандартными типами;
- функции, определённые в данном классе; под функцией понимают метод, возвращающий один параметр;
- функции других классов; для использования этих функций необходимо подключить соответствующие библиотеки классов;
- произвольные функции из базы знаний; для этого требуется предварительно импортировать функцию в текущую базу знаний.

МИЗ типа *Формула* позволяет записать не одну формулу, а некоторую последовательность взаимосвязанных формул. Это упрощает представление метода, делая его менее громоздким и более наглядным.

МИЗ типа *Закон*. Во избежание многократного повторения набора формул, представляющих различные формы одних и тех же физических законов, удобно иметь библиотеку таких законов. Тогда в качестве элементарной расчётной единицы будет выступать закон из библиотеки с указанием свойства, которое требуется получить в качестве результата, и соответствия имён словарных свойств и переменных закона.

Реализация МИЗа этого вида потребовала создания библиотеки физических законов, которые хранятся в стандартном инфиксном виде. Чтобы при использовании закона можно было получить любое свойство, реализован механизм получения обратных функций для выражений произвольной сложности. В основе этого механизма лежит необходимость определения для всех операций и функций, применяемых в физическом законе, обратных им операций и функций.

Причём для операций и функций, имеющих более одного операнда, необходимо определять обратные операции и функции для каждого операнда.

Приведём пример обратной функции. Функция *sin(x)* имеет один операнд, соответственно и обратная функция будет одна, для этого операнда:

x = arcsin (#0), где #0 – результат вычисления прямой функции.

Рассмотрим пример обратной операции. Для операции «+», имеющей 2 операнда, необходимо определить две обратные операции для каждого операнда:

#1 = #0 - #2; #2 = #0 - #1, где #0 – результат вычисления прямой операции, #1 – первый операнд, #2 – второй операнд.

МИЗ типа *Таблица*. Проектирование часто сопровождается выборками из таблиц, причём в качестве таких таблиц могут выступать как нормативные справочники, уже существующие в электронном виде в некотором формате (например, Excel-таблицы либо таблицы базы данных), так и простые зависимости, которые удобно записывать в табличной форме. Таблицы первого вида называются импортируемыми таблицами, о них будет рассказано ниже. Таблицы второго вида, которые создаются внутри метода и впоследствии сохраняются в методе, называются *локальными таблицами* или просто *таблицами*.

Структура МИЗа типа *Таблица* такова: таблицы имеют два измерения, первое – это словарные свойства, которые могут быть входными или выходными параметрами модуля знаний, а второе измерение – это конкретные значения, принимаемые этими свойствами (в том числе диапазоны и формулы).

Недостатками использования локальных таблиц являются увеличение объема метода, а также невозможность использования таблицы другими методами.

МИЗ типа *Импортируемая таблица*. Информация нормативно-справочных баз данных также представляется в виде таблиц, ранее созданных в некотором формате, например, в формате электронных таблиц Excel либо в форматах баз данных (Access, MySQL и др.). Создание локальных копий таких таблиц нерационально, удобнее использовать ссылки на существующие внешние таблицы. МИЗ этого типа должен содержать путь к файлу с внешней таблицей, тип внешнего файла (Excel, Access и др.), имя таблицы во внешнем файле и соответствие между именами полей внешней таблицы и именами локальных свойств. Типы словарных свойств и полей внешней таблицы должны быть совместимыми.

МИЗ типа *Цикл*. В методах проектирования часто используют приближенные методы расчета, которые могут описываться с помощью различного вида циклов. Модули знаний типа *Цикл* реализуют принцип инкапсуляции и позволяют в теле цикла использовать как операции, так и созданные ранее модули инженерных знаний.

Самым простым видом цикла является *цикл с заданным числом итераций*. В качестве параметров для цикла такого типа указываются начальное и конечное значения итератора и шаг итерации. Кроме того, необходимо указать операцию либо МИЗ, используемые в теле цикла (может быть использована только одна операция либо один МИЗ). Реализовать последовательность операций можно путем использования модуля знаний типа Формула или типа Объединение (последний тип будет рассмотрен ниже).

Другой тип цикла – *цикл с предусловием или постусловием*. Такого рода циклы выполняются до тех пор, пока результатом выполнения условия является значение «истина». Различие состоит в том, что для циклов с предусловием результат выполнения условия проверяется до начала каждой итерации, а для циклов с постусловием – после. Для реализации такого цикла должны быть заданы инициализирующая операция либо МИЗ, выполняемые один раз перед началом отработки цикла; условие, проверяемое на каждом шаге цикла; операция либо МИЗ, выполняемые на каждой итерации цикла. Условие может быть сложным, т.е. состоять из нескольких подусловий, объединённых логическими операциями «И» или «ИЛИ». Условия оперируют свойствами из словаря и могут, как и ограничения, оперировать диапазонами (i=(,0], [1,10]) и операциями сравнения (>, <, >=, <=, =).

МИЗ типа *Объединение*. Одним из способов упрощения записи метода и повышения наглядности является использование специального механизма – МИЗа типа *Объединение*, который позволяет объединить в одну логическую единицу набор модулей знаний. Однако объединению следует подвергать сопряженные МИЗы, последовательность выполнения которых определена ранее. Таким образом, в момент создания МИЗа типа *Объединение* необходимо построить семантическую сеть из сопрягаемых МИЗов. Входами и выходами МИЗа типа *Объединение* становятся входы и выходы семантической сети, определённые автоматически во время построения и затем доработанные вручную.

МИЗ типа *Вложенный метод*. Другой способ структурирования метода можно реализовать с помощью МИЗа типа *Вложенный метод*. Такой МИЗ позволяет некоторый фрагмент расчётов вынести в отдельный метод, собрать и отладить его отдельно, а затем интегрировать в текущий метод. Особенностью такого подхода, в отличие от МИЗа типа *Объединение*, является использование автономных словарей свойств каждым из методов. При создании МИЗа этого типа требуется указать интегрируемый метод, а также установить соответствие между словарными свойствами вложенного и текущего методов.

Определение входов метода выполняется автоматически на основании анализа входов и выходов МИЗов. Входами метода считаются свойства, которые во всех МИЗах являются исключительно входными, но не выходными. Допускается ситуация, когда свойство переопределяется в каких-либо МИЗах. Возможны условия, при которых такая ситуация может вызвать некорректное определение входов – переопределяемое свойство может быть не идентифицировано как входное. Заметим, что сама структура автоматически собираемой сети не предполагает многократное переопределение свойств, так как автоматическая сборка может пройти некорректно. Однако в некоторых случаях переопределение упрощает работу. Для этого предусмотрена возможность интерактивной («ручной») коррекции автоматически определённых входов.

Определение выходов метода выполняется аналогичным образом. Выходы – это те свойства, которые, являясь выходными для МИЗов, не используются в качестве входов. Здесь также предусмотрена возможность интерактивного изменения перечня выходов метода, так как список свойств, необходимых пользователю на выходе, может быть шире списка свойств,

определенного автоматически. Для вычисления значений входных свойств (запроса у пользователя на этапе верификации работы метода) необходим служебный МИЗ, который позволит провести инициализацию входов. Этот МИЗ носит название инициализирующего МИЗа или *Ini-МИЗа*. Ещё один служебный МИЗ, позволяющий завершить работу метода, называется *заключающим* МИЗом, или *Out-МИЗом*.

Автоматическая сборка метода представляет собой объединение отдельных модулей знаний в семантическую сеть, которое осуществляется на основе анализа входных и выходных свойств МИЗов.

Построение метода проводится с помощью механизма разделения всех МИЗов на ранги. Под рангом понимается уровень очерёдности исполнения МИЗа. Для *Ini-МИЗа* ранг равен единице, для *Out-МИЗа* ранг максимален.

МИЗы могут образовать цикл, когда входы одного из них служат выходами другого. Такой метод не может быть выполнен, он требует доработки и исправления. В этом случае сборка будет остановлена на том этапе, до которого зацикливание не происходило.

Возможен также случай некритического зацикливания, простейшим примером которого является переопределение свойства.

Результатом сборки является сеть модулей, изображенная на рис. 4.

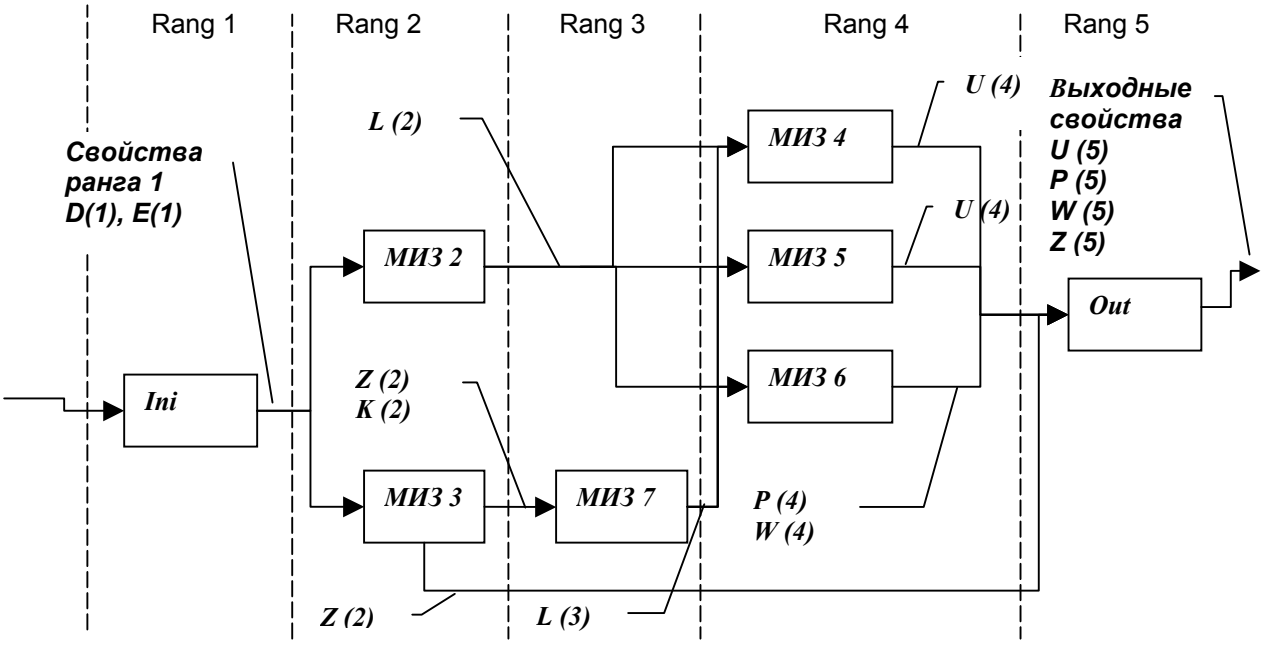

Рис. 4. Сеть модулей знаний как результат автоматической сборки метода

Доработка метода. Из сказанного выше следует, что существует возможность некорректной сборки метода в случае, когда свойства переопределяются. Поэтому пользователю предоставлена возможность интерактивной доработки структуры метода путём перестановки МИЗов. Для того, чтобы МИЗ нельзя было расположить в ранге, где ещё не определены его входные свойства, введены процедуры контроля.

Добавление событий. Событие можно включить в произвольную точку собранного метода: перед выполнением некоторого МИЗа или после него (если

сработают ограничения и МИЗ выполняться не будет, то и событие не возникнет); перед началом работы или после окончания выполнения всего ранга. Генерируемое событие может иметь параметры, в качестве которых используют свойства метода. Как и в случае ручной перестановки МИЗов, для событий необходимо отслеживать, определены ли на момент возникновения события свойства, используемые в событии.

Верификация корректности работы метода. На этом этапе проверяется корректность работы метода. Разработка метода завершается генерацией кода на языке C#, а значит, отработку методов в готовой САПР обеспечит компилятор Visual Studio. На этапе же верификации код ещё не сгенерирован, но требуется осуществить проверку корректности метода. Для этого разработаны специальные методы, названные решателями. Отдельные решатели требуются как для метода в целом, так и для каждого вида модулей знаний.

Решатель позволяет запустить на выполнение созданный метод, используя переданные ему значения входных параметров, и определить значения выходных параметров метода. Используя семантическую сеть модулей, решатель метода последовательно вызывает решатели для каждого МИЗа, а также управляет общими данными (словарём). Запуск отдельных модулей осуществляется порангово, перед запуском каждого из них осуществляется проверка выполнения наложенных на него ограничений. Кроме того, перед запуском каждого МИЗа решатель метода проверяет, определены ли все входные свойства, необходимые данному МИЗу. Если свойства не определены, то метод некорректен и его выполнение невозможно. Такая ситуация может возникнуть, если МИЗ, определяющий свойства, которые служат входными для другого МИЗа, не был выполнен из-за наложенных на него ограничений.

Доработка метода. Если результаты верификации неудовлетворительны, то требуется изменение метода. Предусмотрены такие уровни изменений:

- корректировка «тела» МИЗа без изменения его входных и выходных свойств, не влекущая за собой изменений входов и выходов всего метода и семантической сети;
- изменение перечня свойств метода, требующее повторного автоматического определения входных и выходных свойств;
- добавление или удаление МИЗов, требующее повторного ранжирования, «сжатия» сети для исключения из неё удалённых МИЗов и последующей вставки новых МИЗов.

Подключение метода к модели классов. Когда метод базы знаний создан и отлажен, его необходимо подключить к модели классов. Для выбранного класса объекта машиностроения следует указать имя подключаемого метода, а затем установить соответствие входных и выходных свойств метода базы знаний и свойств класса. Тем самым будут определены фактические параметры метода. При подключении осуществляется проверка типов сопрягаемых свойств.

Кодогенерация. В процессе генерации кода САПР модуль генерации баз знаний на основании семантической сети расчётных единиц генерирует код метода. Подход к кодогенерации аналогичен подходу к разработке решателей: разработаны генераторы кода для каждого вида модулей знаний и генератор кода метода в целом, задача которого – отслеживать последовательность отработки кодогенераторов МИЗов, а также генерировать код, позволяющий проверить

корректность свойств метода (все ли входные свойства определены и соответствуют ли их значения наложенным ограничениям). Если входы некоторого МИЗа не определены, генерируется соответствующая исключительная ситуация.

Таким образом, на основании структур, разработанных в модуле создания информационной модели, и методов расчета, код которых был сгенерирован в этом модуле, с использованием методов взаимодействия с CAD-системой получаем отдельную, независимую от среды разработки систему автоматизированного проектирования.

При выполнении настоящей работы получены следующие основные результаты:

1. Проанализированы возможности и ограничения средств инженерии знаний современных САПР.

2. На основе анализа содержания типовых проектных операций и процедур управления процессом проектирования определен перечень элементов базы знаний (модулей знаний), необходимых для их реализации.

3. Разработаны алгоритмы генерации модулей знаний выбранных типов и методов базы знаний, предусматривающие различные сочетания интерактивных и автоматических процедур, что обеспечивает необходимую гибкость инструментария.

4. Программно реализован и проходит тестирование прототип модуля генерации баз инженерных знаний в составе инструментального комплекса создания специализированных САПР, позволяющий создавать иерархические базы знаний непрограммирующему специалисту.

Список литературы

1. Направления интеллектуализации САПР в машиностроении / Н.Э. Тернюк, В.Ю. Гранин, А.В. Булыгин и др. // Открытые информационные и компьютерные интегрированные технологии: сб. науч. тр. Нац. аэрокосм. ун-та им. Н. Е. Жуковского “ХАИ”. – Вып. 39. – Х.,2008. – С. 14-27.
2. Суханов Ю. О «новом взгляде» на классификацию САПР. Ч. 2 / Ю. Суханов, О. Ефанов, Ю.Береза // CAD/CAM/CAE Observer; – 2007. – № 6 (36). – С. 15-23.
3. Перспективы развития вычислительной техники: в 11 кн. / Е. Кузин, А. Ройтман, И. Фоминых и др.; под ред. Ю. Смирнова. – М., 1989. – Кн.2. Интеллектуализация ЭВМ.
4. Вороненко Д. Будущее CASE-средств / Вороненко Д. // Компьютеры + Программы. – 1996. – № 7. – С. 28-31.
5. Евгенев Г.Б. Системология инженерных знаний / Г. Б. Евгенев. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 376 с.

Рецензент: д-р техн. наук, проф. В.Г. Сухоребрый, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ»

Поступила в редакцию 25.02.2009