Многоагентные системы полуавтоматического проектирования в машиностроении на базе механизма объект-функции

Г.Б. Евгеньев

Московский государственный технический университет им. Н.Э. Баумана, Москва, Россия

Аннотация

Предложена методология создания систем полуавтоматического проектирования изделий машиностроения с использованием технологий искусственного интеллекта. Методология основана на многоагентных методах создания баз знаний и пригодна для разработки систем проектирования и управления цифровых интеллектуальных производств. В качестве унифицированной модели агента предложена объект-функция с разбиением атрибутов на подмножество входных и выходных атрибутов метода агента. Объект-функция рассматривается как унифицированный модуль создания многоагентных систем. Предложены модели представления агента. Показано, что при таком представлении агент эквивалентен правилу-продукции, являющемуся модулем базы знаний. Наименования переменных модуля знаний должны выбираться из словаря, который может быть составлен с использованием различных естественных языков. Механизмы модулей знаний должны обеспечивать реализацию всех функций, которые могут потребоваться при формировании баз знаний. При использовании технологии экспертного программирования процесс формирования модулей знаний, их трансляции с получением на одном из традиционных языков объектных или исполняемых модулей и тестирования производится как одна операция. После получения необходимого набора модулей производится генерация метода, использующего подмножество модулей знаний. Показано, что кооперативное решение комплексных задач обеспечивается системой агентов.

Ключевые слова: проектирование, изделия машиностроения, интеллектуальные системы, многоагентные системы.

Цитирование: Многоагентные системы полуавтоматического проектирования в машиностроении на базе механизма объект-функции / Г.Б. Евгеньев // Онтология проектирования. – 2020. – Т.10, №1(35). – С.50-62. – DOI: 10.18287/2223-9537-2020-10-1-50-62.

Введение

В настоящее время многоагентные системы (МАС) и технологии (МАС-технологии) рассматриваются как наиболее перспективные информационные технологии ИТ-индустрии [1]. Однако внедрение МАС в практику осуществляется недостаточными темпами [2]. Концепция МАС предложена в середине 1980-годов и была высоко оценена как научными, так и индустриальными кругами. С самого начала она позиционировалась как парадигма создания сложных систем, построенная на биологических принципах, которая предлагает строить системы и решать задачи в том же стиле, в каком они решаются в живой природе и человеческом сообществе [3-5]. Основной принцип создания концептуальной модели МАС-приложений использует разбиение сложной задачи с множеством взаимодействующих сущностей на относительно простые законченные подзадачи, понятные специалисту. Решение этих задач поручается программным агентам, которые разрабатываются и программируются практически автономно, работают асинхронно и параллельно и взаимодействуют с помощью простой техники обмена сообщениями на языке, близком к естественному.

Одной из самых привлекательных сторон МАС-парадигмы является её способность естественно и эффективно решать трудную задачу разработки сложных программ, а именно программирование взаимодействий множества компонент программы. В концепции и техноло-

гии МАС, по существу, эта задача отделяется от программирования агентов и реализуется с помощью диалогов и протоколов их взаимодействия. Важно отметить, что в базовой формулировке парадигма МАС акцентирует внимание на этом факте: она формулируется как *парадигма вычислений на основе взаимодействий* (англ. *Computation as interactions*) [5-7]. Концепция обмена сообщениями с использованием диалогов и протоколов для реализации взаимодействий оказалась продуктивной на практике [8]. Вместе с тем, в работах [9, 10] обозначено отсутствие прогресса в широком промышленном применении МАС в настоящее время. *«В то время как имеются убедительные свидетельства важности МАС и технологий как исследовательской области, остаётся неясным, какой практический эффект от неё имеется к настоящему времени»* [10].

Агентом принято называть автономную компьютерную программу, которая способна к целенаправленному поведению в динамической, непредсказуемо изменяющейся внешней среде [2]. Автономность агента понимается как его способность функционировать в интересах достижения поставленной цели без вмешательства человека или других систем и при этом осуществлять самоконтроль над своими действиями и внутренним состоянием.

МАС определяется как *сеть слабо связанных решателей частных проблем* (агентов), которые существуют в общей среде и взаимодействуют между собой для достижения тех или иных целей системы. Взаимодействие может осуществляться агентами либо прямым образом – путём обмена сообщениями, либо *косвенным образом*, когда одни агенты воспринимают присутствие других агентов через изменения во внешней среде, с которой они взаимодействуют. МАС может содержать несколько однотипных или разнотипных агентов, которые могут иметь общие и/или различные цели, могут быть распределёнными по компьютерной сети, могут быть написаны на различных языках программирования и работать на различных операционных платформах [2].

Взаимодействие рассматривается в МАС как основной способ вычислений и координации поведения множества автономных программных или физических агентов. Агенты МАС могут взаимодействовать с целью *кооперативного решения* некоторой общей сложной задачи. В этом случае задача разбивается на более простые относительно автономные задачи, которые поручаются разным агентам. В таком варианте взаимодействие агентов имеет целью координацию локальных решений для достижения некоторого требуемого качества решения задачи в целом. Эта координация может достигаться либо в полностью распределённом варианте, либо с помощью управления с той или иной степенью централизации, реализуемого агентом, специально выделенным для этих целей. Качество решения исходной большой задачи обычно оценивается с помощью некоторой глобальной функции полезности, значение которой зависит от локальных решений агентов.

МАС, как одно из направлений в современных информационных технологиях, формирует *область исследований и разработок*, в которой принято различать три основных направления с общей теоретической основой, где агенты и МАС рассматриваются как:

- *метафора концептуального проектирования* (с акцентом на модульность);
- *источник технологий* (с акцентом на взаимодействие как принцип вычислений и принятия согласованных решений);
- *средство имитационного моделирования* (с акцентом на автономное поведение компонент, взаимодействующих на основе протоколов).

К настоящему времени предложены различные теории агентов и МАС, и каждая из них обладает своими выразительными возможностями, определяет сложность концептуальной модели агента и МАС.

1 Агент как объект-функция

Несмотря на более чем 20-летние усилия по разработке агентских технологий, они не достигли достаточной зрелости, например, зрелости объектно-ориентированного подхода (ООП) в программировании, который появился в конце 1960-х годов и в начале 90-х годов стал общепринятым [3].

Объект в программировании представляет собой некоторую сущность в виртуальном пространстве, обладающую определённым состоянием и поведением, имеющую заданные значения свойств (атрибутов) и операций над ними (методов). Значения атрибутов определяют состояние, а методы - поведение объекта.

Технический объект – это вещь, предмет искусственного (технического) происхождения. В процессе своего жизненного цикла объект первоначально возникает как виртуальная сущность, получаемая в процессе проектирования, а затем реализуется в результате применения тех или иных технологических процессов [11]. Т.е. объект соответствует требованию автономности, присущему агенту, и обладает методом, который определяет поведение объекта. Однако в ООП нет требования целесообразности такого поведения. В случае разработки технологии, при которой объект будет вести себя целенаправленно, он превратится в агента. Для решения этой задачи объект необходимо преобразовать в объект-функцию. Для этого атрибуты необходимо разбить на два подмножества: подмножество входных атрибутов метода и соответственно – выходных.

Целесообразность поведения – это свойство МАС, в которой агенты объединяются для решения комплексной проблемы, для чего должна быть разработана соответствующая технология, включающая обмен информацией между отдельными агентами. Для простоты желательно, чтобы агенты были бы однотипными. Следует отметить, что проблемы повышения уровня интеллектуальности интегрированных САПР применительно к проектированию лопаток осевых турбомашин изучались в работах Б.М. Аронова [12]. Применение описываемых в настоящей статье методов является дальнейшим развитием этого направления.

Концептуально искусственный агент представляет собой объект-функцию. Модель функционального блока, связанного с агентом, определена стандартом *IDEF0*. Согласно этой схеме атрибуты объекта разбиты на три категории: входные, выходные и управленческие. Метод объекта реализует механизм функционального блока.

Вместе с тем агент является объектом, в связи с чем его нотацию целесообразно максимально приблизить к принятому в ООП формату *UML* (*Unified Modeling Language*). Простым решением, учитывающим все упомянутые факторы, представляется форма, приведённая на рисунке 1. Такая форма содержит все компоненты представления класса, предусмотренные в *UML*: имя, атрибуты, операции и описание. Имя заносится в верхний раздел и состоит из идентификатора, отделённого двоеточием от наименования. Наименование строится по правилам *IDEF0*. Оно содержит описание действия (на рисунке 1 – Расчёт) предмета, на который направлено действие (на рисунке 1 – числа), и дополнительной информации (на рисунке 1 – передаточного). Далее следует описание атрибутов. В отличие от представления классов в *UML* это описание разделено на две части: входные и выходные. Такое разделение диктуется тем, что этот объект является функцией.

Наименования атрибутов, как это принято в интеллектуальных системах, должны выбираться из словаря системы, также как и их имена - идентификаторы, необходимые для написания формул. Условие, как это требует *IDEF0*, представляет собой ограничения, накладываемые на входные и управляющие параметры и определяющие область определения функции, реализуемой данным модулем действий. Условия отделяются от наименования атрибута вертикальной чертой, как это принято в математике. На рисунке 1 переменная «Вид СЕ» (сборочная единица) должна иметь значение «Редуктор», чтобы данный модуль выполнил своё

действие. Это – управляющая переменная. Другим условием выполнения модуля является то, что «Частота вращения на выходе, об/мин» должна быть больше нуля. Тогда выходная переменная «Передаточное число» может быть рассчитана по формуле $Ur_z = nh / nt$.

Под описанием атрибутов, как это принято в *UML* для представления классов, заносится операция, выполняемая данным модулем действий. В данном случае – это расчёт по формуле. С точки зрения *IDEF0* – это описание механизма выполнения данного функционального блока.

В нижнем разделе, как это допускает *UML* при моделировании классов, может записываться дополнительная информация. Она часто включает автора, разработавшего данный модуль, и источник информации, который был использован при разработке.

Рисунок 1 - Объект-функция в формате, приближенном к *UML*

При построении диаграмм состояний *UML* целесообразно использовать сокращённое представление, включающее только наименование.

Помимо схемного представления целесообразно иметь эквивалентное ему текстовое представление, которое автоматически генерируется документатором системы экспертного программирования [11]. В интеллектуальных системах проектирования подобные модули действий (с точки зрения *UML*) или функциональные блоки (с точки зрения *IDEF0*), или правила-продукции (с точки зрения искусственного интеллекта, ИИ) принято называть модулями знаний (МЗ).

Имеется возможность с помощью одного МЗ присваивать значения переменным и производить вычисления по набору взаимосвязанных формул. При этом предшествующие выходные переменные могут использоваться для определения последующих выходных переменных. С помощью МЗ типа формул можно формировать текстовые переменные, например, обозначения изделий, тексты содержания технологических операций и переходов и т.д. Пример формирования содержания перехода механической обработки в соответствии с Единой системой технологической документации (ЕСТД) приведён на рисунке 2.

При значениях входных переменных $Per = \text{”Точить”}$, $ElObr = \text{”канавку”}$, $NoEl = 1$, $DinPer2 = \text{”кольц.”}$, $DinPer4 = \text{”окончательно”}$ содержание перехода будет иметь такой вид: ”Точить кольц. канавку 1 окончательно”. Функция *STR* обеспечивает перевод данных из числовой формы в строковую.

Функциональные зависимости в инженерных документах часто имеют табличную форму представления. Для ввода таких зависимостей в базы знаний (БЗ) используются МЗ с механизмами в виде таблиц. Пример такого модуля для присвоения численных значений приведён на рисунке 3. Прилагаемая к модулю таблица может иметь шапку и боковик. В данном случае шапка содержит значения стандартных диаметров осей, а боковик — диапазоны исходных длин осей. На основе этих двух входов таблица позволяет определить значения вы-

ходной переменной — стандартной длины оси. Таблица может быть недоопределённой, т.е. содержать пустые клетки, как это имеет место в примере. При значениях входных переменных, соответствующих этим клеткам, модуль не даст решения. В таком случае проектировщик должен будет изменить входные данные, например исходную длину оси.

Модуль: ТКРЗ		
Разработчик: Евгеньев Г.Б.		
Наименование: Формирование содержания перехода		
Источник информации: ЕСТД		
Наименование	Имя	Ограничение
Переход обработки резанием	<i>Per</i>	[1,)
Элемент обрабатываемый	<i>ElObr</i>	
Номер элемента	<i>NoEl</i>	
Дополнит. информация перехода 2	<i>DinPer2</i>	
Дополнит. информация перехода 4	<i>DinPer4</i>	
Количество элементов	<i>KolEl</i>	$STR(NoEl:0)$ $Per+ "$ $+ DinPer2+ "$ $+ ElObr+ "$ $+ NoElStr+ "$ $+ DinPer4$
Номер элемента строковый	<i>NoElStr</i>	
Содержание перехода	<i>SodPer</i>	

Рисунок 2 - Пример формирования содержания перехода механической обработки в соответствии с ЕСТД

Разработчик: Евгеньев Г. Б.		
Наименование: Назначение стандартной длины		
Источник информации: Анурьев В.И. Справочник конструктора, т.2, стр.8		
Наименование	Имя	Ограничение
Диаметр оси стандартный, мм	<i>d</i>	(0, 22]
Длина оси исходная, мм	<i>Lo</i>	(25, 30]
Длина оси стандартная, мм	<i>L</i>	Таблица: TABL1

TABL1

Длина оси	Диаметр оси стандартный, мм							
исходная, мм	5	6	8	10	12	16, 18	20	22
(25, 28]	28	28	28	28	28	28		
(28, 30]	30	30	30	30	30	30	30	

Рисунок 3 - Пример модуля знаний, представляющего функциональную зависимость в табличной форме

Диаграммы компонентов - это один из видов диаграмм, применяемых при моделировании физических аспектов объектно-ориентированной системы. Они показывают организацию наборов компонентов и зависимости между ними. На рисунке 4 представлена диаграмма компонентов применительно к МЗ, приведённому на рисунке 1. Графически такая диаграмма представляется в виде графа с рёбрами и вершинами. Диаграммы компонентов обычно включают в себя: компоненты и интерфейсы, формирующие вершины графа, и отношения между ними, представляющие собой рёбра этого графа.

Рисунок 4 - Диаграмма компонентов UML применительно к модулю знаний расчёта числа передаточного

На рисунке 4 приведён компонент *PR3* и четыре интерфейса *VidSE*, *nt*, *nh* и *Urz*. Первые три интерфейса представляют собой управляющую (*VidSE*) и входные (*nt*, *nh*) переменные. Компонент *PR3* зависит от них, что отображается пунктирными стрелками. В свою очередь этот компонент реализует интерфейс (*Urz*), то есть обеспечивает вычисление этой переменной.

Неструктурированная совокупность МЗ в определённой прикладной области представляет собой БЗ этой области, аналогичную БЗ производственной системы.

Наименования и имена входных, управляющих и выходных переменных МЗ должны выбираться из словаря БЗ. Словарь может состояться с использованием различных естественных языков, что обеспечивает соответствие первому базовому требованию многоязычности систем класса *wiki*. Словарь в совокупности со значениями содержащихся в нём переменных выполняет в экспертном программировании функцию рабочей памяти производственной системы [11].

Механизмы МЗ должны обеспечивать реализацию всех функций, которые могут потребоваться при формировании БЗ. В число таких функций входят следующие основные:

- вычисление по формулам (в том числе присвоение значений переменным);
- определение значений по таблицам;
- выбор значений из баз данных;
- обновление значений в базах данных;
- занесение значений в базы данных;
- вычисление значений с использованием подпрограмм;
- вычисление значений с помощью методов, сгенерированных из модулей знаний;
- вычисление значений с помощью исполняемых *exe*-модулей или *dll*-библиотек, сгенерированных другими системами.

2 Механизм объект-функции как система агентов

МЗ, представляющие собой элементарные порождающие системы, объединяют в структурированные порождающие системы, носителем моделей которых они являются. Моделью структурированных порождающих систем с точки зрения ИИ являются семантические сети.

Модель структуры БЗ можно представлять как семантическую сеть взаимосвязанных модулей инженерных знаний. Поименованными узлами этой сети являются сами модули, а ориентированными рёбрами — входные и выходные переменные, наименования которых содержатся в словаре.

Семантическая сеть МЗ представляет собой ациклический ориентированный граф (рисунок 5). Ацикличность необходима для выполнения семантической сетью её функционального назначения — обеспечивать определение значений выходных переменных по заданным входным. Из-за наличия циклов в сети блокируется возможность нахождения значений всех или части искомых переменных.

В связи с ацикличностью всё множество МЗ может быть разбито на строго упорядоченное (ранжированное) множество подмножеств. Ранжирование производят, используя понятие протяжённости, которое существует для всех конечных связных графов.

Протяжённость $l(a, b)$ между вершинами a и b определяется как длина самой длинной связывающей их простой цепи.

Для определения протяжённости целесообразно построить фиктивный МЗ_00, не имеющий входных переменных. В число выходных переменных «нулевого» модуля включают входные переменные сети и выходные переменные модулей, не имеющих входов. Такие модули производят установку значений переменных по умолчанию; к их числу на рисунке 5

принадлежит M3_01. Всем M3 без входов присваивают нулевой ранг. M3_00 рассматривают как вершину, от которой рассчитывают протяжённости для всех остальных M3, не имеющих нулевого ранга.

Рисунок 5 - Диаграмма компонентов (а) метода (б)

В результате ранжирования с помощью протяжённости подмножество M3 ранга 1 составят модули, имеющие в качестве входных переменных только входные переменные метода и выходные M3 ранга 0. Подмножество ранга 2 образуют M3, использующие в качестве входных входные переменные метода и выходные переменные модулей рангов 0 и 1, и т.д.

Ранжирование M3 производится однократно при формировании сети. В дальнейшем ранги используют как метазнания для логического вывода на сети. Модули запускаются в порядке возрастания их рангов. M3 нулевого ранга запускаются без проверки предусловий. Далее производится проверка предусловий модулей ранга 1 и запускаются механизмы тех M3, предусловия которых удовлетворены.

Альтернативные решения могут возникать у модулей, связанных в узел типа ИЛИ. Такой узел формируют M3, принадлежащие к множеству одного ранга и имеющие одинаковые наборы входных и выходных переменных. На рисунке 3 это — M3_11 и M3_12.

Возникновение альтернативных решений зависит от того, является узел ИЛИ исключающим или неисключающим, т.е. пересекаются или не пересекаются области определения этих модулей, описанные в предусловиях.

Вариант, представленный на рисунке 6а, соответствует исключающему ИЛИ, так как области определения не пересекаются, а вариант на рисунке 6б — неисключающему, поскольку при значениях переменных предусловий, принадлежащих к общей области, могут быть применены оба M3. Последний вариант рассматривается как некорректное построение БЗ.

Модули рангов 2 и 3 на рисунке 5 формируют два узла типа И. МЗ, образующие узел типа И, должны принадлежать к подмножеству одного ранга и не иметь общих выходных переменных. Подмножество ранга 1 на рисунке 5 может быть представлено в виде логического выражения $(МЗ_{11} \vee МЗ_{12}) \wedge МЗ_{13}$. Узлы типа И не порождают альтернативных решений. Все модули, входящие в такие узлы, должны быть запущены на исполнение, поскольку они определяют разные переменные.

Подмножества различных рангов связываются логической связкой И, так как для решения задачи они все должны быть выполнены.

Различают два основных типа стратегий управления: безвозвратный и пробный. В безвозвратном режиме управления выбирается применимое правило и используется необратимо без возможности пересмотра в дальнейшем. В экспертном программировании используется такой режим [11].

При использовании технологии экспертного программирования процесс формирования МЗ, их трансляции с получением на одном из традиционных языков объектных или исполняемых модулей и тестирования производятся как одна операция. После получения необходимого набора МЗ производится генерация метода, использующего подмножество сгенерированных МЗ. Метод генерируется с использованием того же языка, что и МЗ, и представляет собой скомпилированную реализацию решателя для данного набора МЗ. В скомпилированном механизме выбора продукционных правил используется схема управления, подобная сети Петри. Процесс планирования и разрешения конфликтов использует условия запуска МЗ. Программный модуль, соответствующий МЗ, исполняется, если имеются значения всех входных и управляющих переменных и выполнено условие его применения.

Сгенерированный метод может быть использован в качестве механизма МЗ, что позволяет использовать иерархию правил.

Чтобы генерируемые методы согласно теореме о структурировании могли решать любые задачи программирования, в них должны быть реализованы три базовые структуры: следование, альтернатива и один из циклов. В экспертном программировании следование МЗ определяется последовательностью вычисления переменных, а альтернатива – условиями, заложенными в МЗ. Для формирования циклов введена выделенная переменная «конец цикла» (*Fincalc*), при появлении которой система генерирует циклический метод, выполняющийся до тех пор, пока *Fincalc*, которому предварительно присвоен 0, не примет значения 1. Один метод может содержать только один цикл. Тело цикла включает модули рангов 1 и более. Модули ранга 0 обеспечивают начальные установки значений переменных и в цикле не исполняются. Возможность использования методов в качестве механизмов МЗ позволяет получать как последовательности циклов, так и программы с вложенными циклами.

На рисунке 5а представлена диаграмма компонентов метода, а на рисунке 5б метод изображён в качестве компонента. Общая архитектура программ, генерируемых при экспертном программировании, соответствует принятой в модульном программировании. При этом модули, изображённые на этом рисунке, соответствуют методам, составленным из МЗ. Методы в качестве механизмов МЗ могут входить в вышерасположенную по иерархии программу.

МЗ, представляющие собой элементарные порождающие системы, объединяют в методы объектов. Из числа средств *UML* для представления методов, построенных на базе объектных функций, наиболее подходят диаграммы состояний.

Рисунок 6 - Области определения исключающих (а) и неисключающих (б) ИЛИ

3 Кооперативное решение сложных задач системой агентов

Парадигма МАС, реализуемая на основе взаимодействий, рассмотрена на примере проектирования червячных редукторов.

Имеются два основных способа использования технологий ИИ при автоматизации проектирования: встраивание и надстраивание. Первый подход используется в ряде CAD-систем [13]. Недостатком такого подхода является то, что инженеры, не занимающиеся конструированием, например технологи, не могут использовать интеллектуальные возможности таких систем. В статье предлагается использовать подход надстраивания, который позволяет применять методы ИИ на всех этапах жизненного цикла изделий.

МАС обладают статическими и динамическими свойствами. В UML для представления статического и динамического вида системы используются различные диаграммы. Основной статической диаграммой UML является диаграмма классов. Представление классов объектов в ней не обладает функциональностью, необходимой для описания МАС. Во-первых, не предусмотрено деление свойств на входные и выходные, что необходимо для представления объект-функций. Во-вторых, классы объектов не имеют предусловий, необходимых для управления процессом поведения системы. Эти проблемы можно решить, не выходя за рамки средств, принятых для представления диаграмм классов.

Например, в диаграмме классов червячного редуктора коренным классом является классификатор Единой системы конструкторской документации (ЕСКД). Перед началом проектирования целесообразно провести морфологический анализ проблемы. В цифровом машиностроении для проведения этого анализа следует использовать классификатор ЕСКД (рисунок 7). В МАС морфологический анализ производится в форме диалога с инженером. Отвечая на вопросы системы, специалист формирует классификационный код проектируемого изделия, по которому система переходит к БЗ многоагентного синтеза этого изделия. Таким образом, проектирование осуществляется последовательным использованием пары методов «Морфологический анализ» - «Многоагентный синтез».

В результате морфологического анализа сформирован код вида изделия, необходимо произвести многоагентный синтез червячного редуктора, для чего нужно задать параметры технического задания, отмеченные в качестве атрибутов этого изделия: мощность и частота вращения на выходном валу, модель электродвигателя и т.д. В результате может быть задействован многоагентный метод проектирования редуктора.

Редуктор червячный состоит из трёх основных узлов: колеса червячного, червяка и корпуса изделия. В состав узла колеса червячного входят: колесо червячное, вал колеса и подшипник колеса передний. В свою очередь имеются два варианта исполнения колеса: цельно-металлическое и сборное. В случае сборного колеса оно включает детали обода и ступицы. Все перечисленные объекты имеют соответствующие интеллектуальные методы конструирования с получением 3D моделей и чертёжной документации.

Аналогичным образом формируется МАС проектирования других узлов изделия.

Чтобы обеспечить возможность генерации 3D моделей с помощью БЗ необходимо создать МЗ, имеющий механизм, с помощью которого на основе параметризованной в CAD-системе модели детали и (или) сборочной единицы можно было бы сгенерировать 3D модель изделия с рассчитанными в других модулях величинами размеров [14].

В качестве CAD-системы в примере генерации 3D модели червячного колеса используется программа *Solid Works*. Диаграмма протокола работы МАС включает: выбор проектируемого узла, проектирование узла колеса червячного, проектирование колеса червячного, генерация 3D моделей колеса червячного (рисунок 8).

Рисунок 7 - Морфологический анализ проектируемого изделия

Рисунок 8 - Генерация 3D моделей колеса червячного

После построения 3D моделей всех узлов имеется возможность генерации модели изделия в целом (рисунок 9).

Рисунок 9 - Генерация 3D модели изделия

Заключение

В качестве унифицированной модели агента предложена объект-функция, которая рассматривается как унифицированный модуль создания МАС. При использовании технологии экспертного программирования процесс формирования МЗ, их трансляции с получением на одном из традиционных языков объектных или исполняемых модулей и тестирования производятся как одна операция. После получения необходимого набора модулей производится автоматическая генерация метода, использующего подмножество сгенерированных МЗ.

Предложенная методология позволяет создавать МАС непрограммирующим специалистам. В соответствии с этой методологией разработку таких систем осуществляют магистранты МГТУ им. Н.Э. Баумана.

Список источников

- [1] Gartner. Top Strategic Predictions for 2016 and Beyond: The Future Is a Digital Thing. <https://www.gartner.com>.
- [2] **Городецкий, В.И.** Современное состояние и перспективы промышленных применений многоагентных систем / В.И. Городецкий, О.Л. Бухвалов, П.О. Скобелев, И.В. Майоров // Управление большими системами: сборник трудов, выпуск 66. 2017. - С.94-157.
- [3] **Alanis, A.Y.** Bio-inspired Algorithms for Engineering / Alma Y. Alanis, Nancy Arana-Daniel and Carlos López-Franco. Elsevier Inc. 2018. - 152 p.
- [4] **Binitha, S.** A Survey of Bio inspired Optimization Algorithms / Binitha S, S Siva Sathya // International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-2, May 2012. P.137-151.
- [5] **M. Blowers and A. Sisti.** Evolutionary and Bio-inspired Computation: Theory and Applications, Proceedings of Spie (Book 6563), Society of Photo Optical (April 25, 2007). – 174 p.
- [6] **Avi Wigderson.** Mathematics and Computation: A Theory Revolutionizing Technology and Science PRINCETON 2019 – 440 p.
- [7] **Makhatadze, G.I.** Linking computation and experiments to study the role of charge-charge interactions in protein folding and stability. IOP Publishing Phys Biol. 2017 Feb 7. - DOI: 10.1088/1478-3975/14/1/013002.
- [8] **Luck M. et al.** Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing) // AgentLink. – 2005. – <http://www.agentlink.org/roadmap/>.

- [9] **Deloach, S.A.** Moving multi-agent systems from research to practice // Int. J. Agent-Oriented Software Engineering. – 2009. Vol.3, No.4. – P.378–382.
- [10] **Müller J., Fisher K.** Application Impact of Multiagent Systems and Technologies: A Survey // In «Agent-Oriented Software Engineering» book series. – Springer, 2013. – P. 1–26.
- [11] **Евгениев, Г.Б.** Основы автоматизации технологических процессов и производств. Т.1: Информационные модели. Т.2: Методы проектирования и управления - Москва, Изд-во МГТУ им. Н.Э. Баумана, 2015. - Т.1: - 441 с. - Т.2: - 479 с.
- [12] **Аронов, Б.М.** О путях повышения уровня интеллектуальности интегрированных САПР лопаток осевых турбомашин. Проблемы и перспективы развития двигателестроения / Самар. гос. аэрокосм. ун-т; [Гл. ред. Е. В. Шахматов]. – Самара: СГАУ, 1999. – Вып.3, ч. 2. – С.245-248.
- [13] **Thilmany J.** Putting Artificial Intelligence to Work in CAD Design – Engineers Rule. December 21, 2017. - <https://www.engineersrule.com/solidworks-puts-artificial-intelligence>.
- [14] СПРУТ-Технология. Автоматизация проектирования. - <https://sprut.ru/>.

Сведения об авторе

Евгениев Георгий Борисович, 1938 г. рождения. Окончил Московский авиационный технологический институт в 1960 г., к.т.н. (1964), д.т.н. (1975). Лауреат премии Совета Министров СССР (1991). Профессор кафедры «Компьютерные системы автоматизации производства» МГТУ им. Н.Э. Баумана. Член Российской ассоциации искусственного интеллекта. В списке научных трудов более 150 работ в области САПР и ИИ. AuthorID (РИНЦ): 17685. g.evgeniev@mail.ru.

Поступила в редакцию 30.01.2020, после рецензирования 17.02.2020. Принята к публикации 15.03.2020.

Multi-agent systems of semi-automatic design based on object-functions model in engineering

G.B. Evgeniev

Moscow State Technical University named after N.E. Bauman, Moscow, Russia

Abstract

A methodology for creating semi-automatic design systems for engineering products using artificial intelligence technologies is proposed. The methodology is based on multi-agent methods for creating knowledge bases and is suitable for the development of design and control systems for digital intelligent industries. As a unified agent model, an object-function mechanism is proposed with a partition of attributes into a subset of the input and output attributes of the agent method. Object-function is considered as a unified module for creating multi-agent systems. Graphic and textual models of agent representation are proposed. It is shown that with this representation, the agent is equivalent to the production rule, which is a knowledge base module. The names of the variables of the knowledge module should be selected from a dictionary that can be compiled using various natural languages. The mechanisms of knowledge modules should ensure the implementation of all the functions that may be required in the formation of knowledge bases. It is shown that the mechanism of an object-function mechanism can be a system of agents. When using expert programming technology, the process of generating knowledge modules, translating them into object or executable modules in one of the traditional languages, and testing is performed as a single operation. After obtaining the necessary set of modules, a method is generated that uses a subset of the generated knowledge modules. The generated method can be used as a mechanism of the knowledge module, which allows the use of a hierarchy of rules. It is shown that a cooperative solution to complex problems is provided by a system of agents.

Key words: engineering design, products, intelligent systems, multi-agent systems.

Citation: Evgenev GB. Multi-agent systems of semi-automatic design based on object-functions model in engineering [In Russian]. *Ontology of designing*. 2020; 10(1): 50-62. – DOI: 10.18287/2223-9537-2020-10-1-50-62.

List of figures

Figure 1 – Object-function in a format close to UML

Figure 2 – An example of the formation of the content of the transition of machining in accordance with the standard

Figure 3 – An example of a knowledge module representing a functional dependency in tabular form

Figure 4 – UML component diagram as applied to the gear ratio calculation knowledge module

Figure 5 – Diagrams of the components (a) of the method (b)

Figure 6 – Areas of definition of exclusive (a) and non-exclusive (b) OR

Figure 7 – Morphological analysis of the designed product

Figure 8 – Generation of 3D models of a worm wheel

Figure 9 – 3D product model generation

References

- [1] Gartner. Top Strategic Predictions for 2016 and Beyond: The Future Is a Digital Thing. <https://www.gartner.com>.
- [2] **Gorodetsky VI, Bukhvalov OL, Skobelev PO, Mayorov IV.** Current status and prospects of industrial applications of multi-agent systems [In Russian], UBS, 2017; 66: 94-157.
- [3] **Alma Y. Alanis, Nancy Arana-Daniel and Carlos López-Franco.** Bio-inspired Algorithms for Engineering. Elsevier Inc. 2018. 152 p.
- [4] **Binitha S, S Siva Sathya.** A Survey of Bio inspired Optimization Algorithms International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-2, May 2012. P.137-151.
- [5] **M. Blowers and A. Sisti,** *Evolutionary and Bio-inspired Computation: Theory and Applications*, SPIE Press, 2007.
- [6] **Avi Wigderson.** Mathematics and Computation: A Theory Revolutionizing Technology and Science PRINCETON 2019 440p.
- [7] **Makhataдзе GI.** Linking *computation* and experiments to study the role of charge-charge *interactions* in protein folding and stability. IOP Publishing Phys Biol. 2017 Feb 7. DOI: 10.1088/1478-3975/14/1/013002.
- [8] **Luck M. et al.** Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing) // AgentLink. 2005. <http://www.agentlink.org/roadmap/>.
- [9] **Deloach SA.** Moving multi-agent systems from research to practice // Int. J. Agent-Oriented Software Engineering. 2009. Vol.3, No.4. P.378–382.
- [10] **Müller J., Fisher K.** Application Impact of Multiagent Systems and Technologies: A Survey // In «Agent-Oriented Software Engineering» book series. Springer, 2013. P.1–26.
- [11] **Evgenev GB.** Fundamentals of automation of technological processes and production [In Russian]. V.1: Information models. V.2: Design and control methods - Moscow, Publishing house of MGTU named N.E. Bauman, 2015. V.1: 441 p. V.2: 479 p.
- [12] **Aronov BM.** About the ways to increase the level of intelligence of integrated CAD systems of axial turbomachine blades [In Russian]. Problems and prospects for the development of engine building. Samara state aerospace un-t; [Ch. ed. E.V. Shakhmatov]. Samara: SSAU, 1999. Vol. 3, part 2. P.245-248.
- [13] **Thilmany J.** Putting Artificial Intelligence to Work in CAD Design – Engineers Rule. December 21, 2017. <https://www.engineersrule.com/solidworks-puts-artificial-intelligence>.
- [14] SPRUT-Technology. Automation design [In Russian]. <https://sprut.ru/>.

About the author

Georgy Borisovich Evgenev (b. 1938) graduated from the Moscow aviation technological Institute in 1960, PhD (1964), Doctor of engineering (1975). He is the Winner of the USSR Council of Ministers Prize (1991). Professor at Moscow State Technical University named after N.E. Bauman. Member of the International Association for Ontology and its Applications. Co-author of about 190 scientific articles and abstracts in the field of CAD and AI. AuthorID (RCI): 17685. g.evgenev@mail.ru.

Received January 30, 2020. Revised February 17, 2020. Accepted March 15, 2020.