УДК 621.9.06-52

О.И. Лисовиченко, А.А. Лавров

ПОДХОД К СОЗДАНИЮ ОБЪЕКТНО-ОРИЕНТИРОВАННОЙ СРЕДЫ МОДЕЛИРОВАНИЯ ПРОИЗВОДСТВЕННЫХ СИСТЕМ

Введение

Проектирование современных гибких компьютеризированных систем (ГКС) осуществляется, в основном, исходя из планируемой номенклатуры объектов производства. На практике, однако, не всегда возможна достаточно точная оценка и прогноз структурных и параметрических вариаций этих объектов. В результате, созданная ГКС может оказаться либо не в состоянии выполнить поставленные задачи, либо будет выполнять их с неприемлемо высокими затратами.

В связи с этим необходимо уже на этапе проектирования ГКС заложить такую степень ее технологической гибкости, которая обеспечивала бы возможность изменять номенклатуру объектов производства без существенного переоборудования самой системы.

Проектирование и исследование систем такого типа осложняется их разнородным построением (прежде всего, интеграцией существенно отличающихся по физическим и функциональным свойствам компонент), а также особенностями синхронизации и координации. В частности, для описания и исследования логики процессов системы на верхних уровнях можно использовать аппарат сетей Петри (СП). Системы дифференциальных уравнений используются для исследования динамики функционирования, а средства и методы процессных алгебр применимы при оценке и разрешении конфликтных и тупиковых ситуаций и др. Эти методы позволяют эффективно организовывать процесс разработки путем их взаимного семантического согласования и в наибольшей степени соответствующие корректным подзадачам. Кроме того, ситуация усугубляется отсутствием ориентированных на пользователя унифицированных средств визуальной поддержки принятия решений проектировщика или исследователя.

Подход к решению задач объединения разных формализмов описания модулей системы рассмотренны в [6, 7, 11]. В данной статье пойдет речь о решении проблемы средств визуальной поддержки: структуры и методов описания систем.

Постановка задачи

Цель данной статьи – разработка концепции автоматизированного рабочего места (АРМ) синтеза/анализа ГКС на основе открытой объектно-ориентированной среды моделирования.

 ISSN 1562-9945

Изначально, АРМ должен удовлетворять следующим требованиям:

- “интеллектуальный” выбор и использование семантически согласованных формализмов для описания процессов в ГКС, описанных в БД АРМ;
- возможность подключения дополнительных модулей для решения специализированных задач. Например, модуля решения проблемы управления ограниченными пропускными возможностями составляющих производственных систем с *периодическим характером операций* [1], модуля *распределенного* управления материальными потоками;
- интеграция дополнительных формализмов для более эффективного и полного описания процессов системы (объектные СП, разностные уравнения, методы маршрутизации транспортных потоков [2, 18] и т.д.);
- наличие функций определения и визуализации переменных пространственных положений как при абстрагированном описании системы на верхнем уровне иерархии, так и отдельных модулей системы (ГПМ, ПР, АТМ и т.д.) на более низких уровнях описания системы;
- наличие функции подготовки выходных данных для дистанционного управления в произвольном рабочем пространстве [8] и т.д.).

1 Концепция объектно-ориентированной среды моделирования

Одним из возможных путей для решения поставленной задачи могло бы стать использование уже разработанных специализированных пакетов, в частности, позволяющих пользователю самостоятельно конструировать и добавлять новые элементарные блоки, используя входной язык пакета - Model Vision Studium, Anylogic, Dymola, Sim Station, либо обращаться к низкоуровневым процедурным языкам - Simulink, Mat Lab и т.п. В отличие от традиционных систем автоматизированного проектирования (CAD-систем), такие пакеты позволяют строить модель из функционально-ориентированных блоков, исследовать геометрию, кинематику, динамику движений. Однако, сложности системного характера (высокие требования к программной платформе), совместимости (проблемы информационного обмена с другими компонентами), финансовые (высокая стоимость полнофункциональных коммерческих версий пакетов) ограничивают применение такого подхода при построении АРМ. К тому же, модели таких сложных типов не реализованы в полном объеме ни в одном из известных пакетов. Речь идет о событийно - управляемых многокомпонентных иерархических моделях переменной структуры, построенных из блоков с переменными типа “вход”, “выход”, “состояние”, “контакт” и т.п. Поэтому необходимо выполнение новой разработки программного обеспечения, реализующего отмеченные задачи, как на функциональном, так и на алгоритмическом уровне (например, рассмотреть

возможность информационного обмена с CAD, GPSS - модулями). Один из таких подходов интеграции предложен в работе [10].

В свою очередь, алгоритмы описания структурных моделей и поточных отношений в ГКС должны соответствовать принципам структурного [5] и объектно-ориентированого [3] проектирования.

Структурные модели и определения типов (т. е. специализаций) узлов (которые в свою очередь отвечают отдельным физическим или виртуальным модулям), а также определение поточных отношений (которые отображают взаимную зависимость компонентов и содержание материальных и информационных потоков между ними) описаны в [9].

В предлагаемом решении объектная ориентированность рассматривается в широком смысле: не только в контексте программирования, но и в отношении системного моделирования и проектирования в целом. Возможность интеграции методов структурного и объектно-ориентированного проектирования достигается на базе применения Унифицированного Языка Моделирования (Unified Modeling Language — UML) [4,14]. Рассмотрим некоторые базовые элементы и определения такого подхода.

Определение 1. Класс — это группа сущностей (объектов), обладающих сходным набором: данные и правила поведения с другими объектами (интерфейс, утилита) [3,14].

UML предлагает несколько разновидностей класса, в зависимости от свойств определения и прав доступа к набору описанных в этом классе операций.

Определение 2. Интерфейс — это класс, в котором определены только операции.

Определение 3. Утилита — это класс, все атрибуты и операции которого общедоступны (public).

Графическое представление класса на рис. 1.

Рис. 1 – Графическое обозначение класса

Атрибуты хранят инкапсулированные данные класса, а методы описывают поведение объектов класса. Отображение любой из секций или указателя на отсутствие элементов не является обязательным;

Для описания иерархии на уровне системы схемы классов необходимо определить отношения между классами, включая возможность рефлексивного отношения класса с самим собой (см. таблицу 1).

Пример схемы классов для некоторых компонент ГКС приведен на рис. 2.

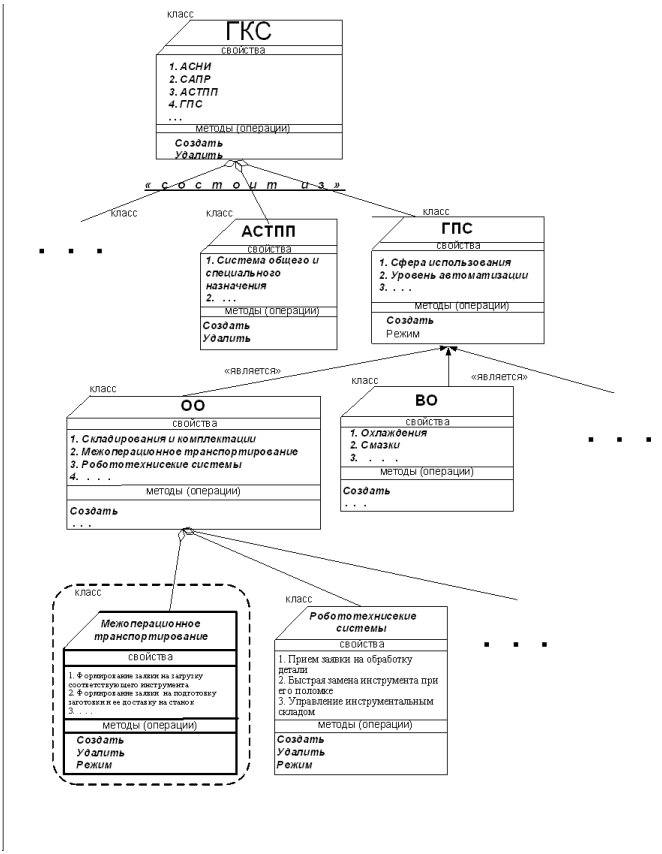
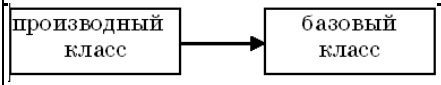
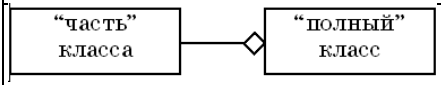
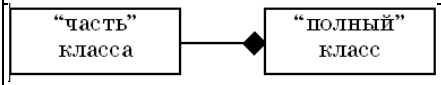
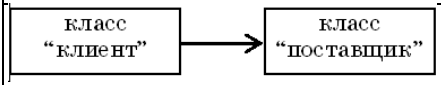
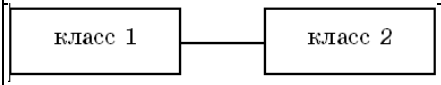
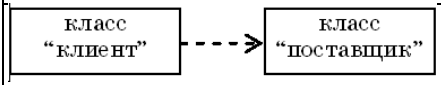


Рис. 2 – Схема некоторых классов для предметной области ГКС

Таблица 1.

Отношения между классами

Изображение	Отношение
	Наследование (Inheritance)
	Сборка (Aggregation)
	Композиция (двуместная связь) (Composition)
	Однонаправленная ассоциация (Unidirectional Association)
	Двунаправленная ассоциация (Bidirectional Association)
	Зависимость (Dependency)

В дальнейшем отдельнго представителя некоторого класса будем называть объектом класса или просто *объектом*.

Определение 4. Объект – некоторая сущность, которая инкапсулирует в себе данные (атрибуты) и методы как единое целое и взаимодействует с внешним окружением через определенный интерфейс. В нашем случае предлагается использование интерфейса на базе XML-документов [12, 13]. Каждый объект всегда является экземпляром определенного класса.

Объект обладает набором состояний, в которых он может находиться, строго определенным поведением и уникальным идентификатором; структура и поведение схожих объектов определяется в их общем классе.

Объекты могут исполнять конкретные роли. Роль определяет отношение между классом и его экземплярами [3], выделяя некоторое их подмножество. Считается, что все эти объекты похожи по своему поведению и состояниям, которые они могут принимать. Например, в системе может существовать много объектов класса “Основное оборудование”, но часть из них в данный момент играет роль вызываемых (например, модуль обработки), а часть — вызывающих (например, модуль межоперационного транспортирования), в то время как остальные (например, модуль сборки) остаются свободными.

Каждому объекту на схемах соответствует графический элемент, показанный на рис. 3.

Объект может находиться в определенных состояниях, а его описание

 ISSN 1562-9945

Рис. 3 – Графическое обозначение объекта

имеет следующий синтаксис:

<имя объекта> : <имя класса> ‘[‘<список состояний>’]’, где <имя объекта> - полное название объекта; <имя класса> — полное имя класса, представителем которого является объект; <список состояний> — набор состояний объекта (рис. 4), в которых он может находиться во время своей “жизни” в системе. Возможные состояния объекта определяются на этапе анализа проектируемой системы, когда. выделяются основные фазы, в которых объект может находиться. В последующем, в процессе проектирования системы, эти состояния могут корректироваться.

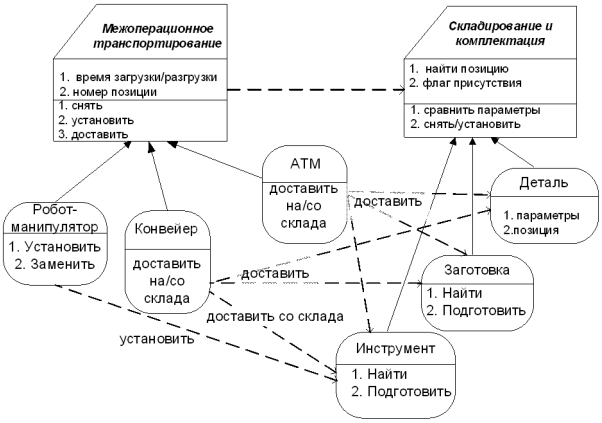


Рис. 4 – Графическое отображение отношений между классами и объектами

Все объекты одного и того же класса, как отмечалось выше, характеризуются одинаковыми общими наборами атрибутов, которые наследуются от этих классов. Однако объединение объектов в классы определяется не наборами атрибутов, а семантикой. Так, например, объекты ГПМ и АТМ могут иметь одинаковые атрибуты: тип, масса, элементы ориентирования и т.д. При этом они могут относиться к одному классу, если рассматриваются в задаче просто как технологический модуль, либо к разным классам, если рассматриваются с точки зрения выполняемых операций (что в практических задачах является более естественным).

Все эти свойства нужно учитывать при более детальном построении

диаграммы классов системы и взаимных отношений как между классами, так и между классом и объектом. На рис. 5 представлена логическая структура функций в виде подсистем и отношений между подсистемами системы синтеза/анализа ГКС. Все подсистемы связываются через базовую подсистему “Экспертная система”, которая, используя интерфейс пользователя и базу знаний, формирует запросы в формате XML-документа необходимым для решения поставленной задачи компонентам. При получении ответа на запрос, подготавливает и предоставляет результаты пользователю.

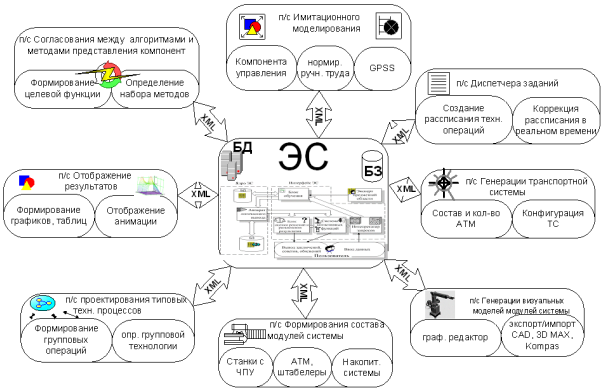


Рис. 5 – Структурная схема системы моделирования ГКС

Например, нам требуется выбрать ПР с определенными параметрами из базы данных (БД). Через интерфейс подсистемы “Экспертная система” формулируем задачу, определяя необходимые нам параметры ПР. Подсистема “Формирование состава модулей ГКС” выступает в качестве клиента, которая принимает запрос в виде XML-документа, формирует ответ (список) и выдает запрос в Базу Знаний (БЗ). WEB-сервер, который обслуживает БЗ подсистемы “Экспертная система” транслирует запрос в БЗ и далее передает SQL-запрос непосредственно в БД для формирования списка требуемых ПР в XML формате. Когда список ПР создан идет формирование ответа и выдача результата (рис. 6). Другими словами данная последовательность (пользователь→WEB-сервер→БЗ→ЕВ-сервер →компонента →ЕВ-сервер →З →ЕВ-сервер →Д →ЕВ-сервер →З →ЕВ-сервер→ пользователь) показывает очередность работы БЗ, БД, WEB-сервера при работе с компонентами. Результат может формироваться как с использование когнитивных функций самой подсистемы “Экспертная система”, так и с использованием подсистемы “Отображение результатов”

Каждый XML-документ, подобно HTML-документу, состоит из *заголовка* сообщения и *тела* сообщения (содержание запроса, обрамленное тегами запроса). Вид типового документа представлен ниже (описание

 ISSN 1562-9945

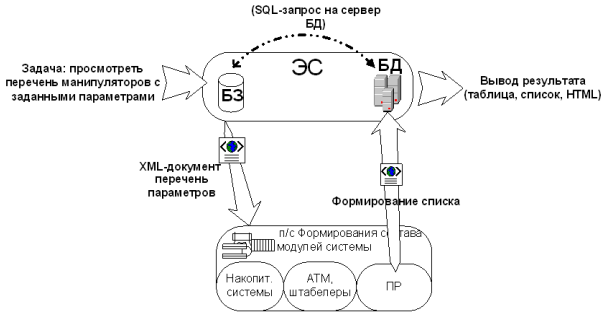


Рис. 6 – Алгоритм формирования запроса в формате XML-документа

параметров манипулятора):

```
<?xml version="1.0"?>
<xmlDocument>
<Header>
описание тела заголовка
</Header>
<Request>
описание тела запроса
</Request>
</xmlDocument>
```

типовая структура

```
<?xml version="1.0"?>
<xmlDoc1>
<Header>
<From>TExpertSystem</From>
<To>TModulesList</To>
</Header>
<Request>
<Sender>Manipulator</Sender>
<TimeOperation=0.35/>
<TypeOperation=transport/>
<TypeOrientation=pallete/>
<NumberCapture=3/>
<TypeCapture= sponges />
</Request>
</xmlDoc1>
```

пример запроса

Рис. 7 – Вид типового XML-документа для информационного обмена между подсистемами АРМ

Приемлемым кандидатом реализации объекта в объектно-ориентированном программировании является компонента.

Определение 5. Компонента — совокупность переменных и правил поведения, которая взаимодействует с внешним миром исключительно через внешние переменные. Компонента всегда — явно или неявно — является экземпляром некоторого класса. Например, когда при синтезе системы на структурной схеме размещается новый блок (например, ГПМ), тем самым неявно порождается новый экземпляр выбранного класса из библиотеки (БД основного оборудования). В общем случае модель может содержать много экземпляров одного и того же класса. Использование классов позволяет моделировать многокомпонентные системы, имеющие регулярную или переменную структуру.

Определение подсистем как наборов объектов

В предлагаемом подходе моделирующая система рассматривается как набор подсистем, каждая из которых, в свою очередь, представляет собой совокупность компонент определенных классов. На рис. 8 приведен пример одной из подсистем - “Согласование между алгоритмами и методами представления компонент”. Задача подсистемы заключается в динамическом построении из множества модулей (экземпляров), конфигурации, которая наилучшим образом описывает текущее состояние системы с максимально возможной степенью адекватности поставленной задаче.

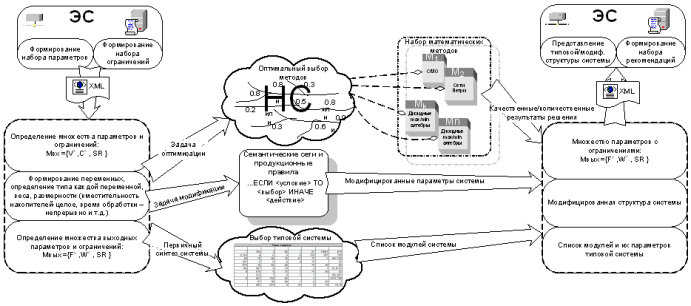


Рис. 8 – Структура подсистемы “Согласование между алгоритмами и методами представления компонент”

Смешанное использование и рациональный выбор методов на базе технологии нейросетей с нечеткой логикой выбора [17,16]. Модульная структура построения и выбора алгоритмов использует как продукционные правила, так табличный выбор типовых конфигураций, удовлетворяющего параметрам модуля (рис. 8).

Задание параметров формируется на базе следующего набора множеств:

$V^x = \{V(X_1), V(X_2), \ldots, V(X_k)\}$ – множество входных переменных;

$C^x = \{c_1^x, c_2^x, \ldots, c_m^x\}$ – множество степеней зависимости;

$SR = \{R_1, R_2, \ldots, R_n\}$ – множество правил, объединенных общими семантическими свойствами или предназначенных для решения общей задачи;

$F^y = \{F_1^y, F_2^y, \ldots, F_z^y\}$ – множество “четких” фактов, определяющих вес каждой выходной переменной;

$W^x = \{W(Y_1), W(Y_2), \ldots, W(Y_j)\}$ – множество выходных переменных.

При необходимости решения определенной задачи создается подкласс со своей структурой независимых модулей. Функция степени зависимостей c_i^x определяет комбинированную стратегию использования определенного модуля, при условии циклической проверки текущих параметров и ограничений.

 ISSN 1562-9945

Структура модуля в виде агрегата

Рассмотрим формирование модулей на примере экземпляра "Логическое управление" подсистемы "Имитационное моделирование.

Пусть модуль "Логическое управление" удовлетворяет некоторым требованиям:

1. Алгоритмы управления и контроля отдельными объектами оформлены в виде автономных подклассов (экземпляров основного класса);
2. Подклассы изолированы друг от друга в силу инкапсуляции и могут быть независимо спроектированы и отлажены;
3. Объекты подклассов реализуются строго последовательно, для упрощения алгоритмизации и программирования.

Представим модуль контроля и управления агрегатом общего вида (рис. 9). При этом будем полагать, что созданный класс модуля размещается на том же иерархическом уровне, что и модули управляемой системы (например, управление потоками межоперационного транспортирования). На нижнем уровне по отношению к классу находится уровень ресурсов управляемого (контролируемого) агрегата, верхним уровнем по отношению к классу служит модуль основного оборудования (или подсистема верхнего уровня).

Определим типовой класс как множество:
$T = \{A, B, H, X, Y, O_{вх}, O_{вых}, C, S, Z\}$, где:

1. A – множество настроек объекта класса T, перечень данных и функций, которые должны идентифицировать и инициализировать конкретные объекты класса T. Данный перечень включается в качестве набора параметров в открытом методе, называемом конструктором класса и имеющем то же имя, что и класс T. В теле конструктора выполняется присваивание закрытым данным – членам класса значений параметров из множества A.;
2. B – множество состояний $B = \{b_0, b_1, \ldots, b_N\}$ объекта класса T; оно состоит из перечня наименований состояний алгоритмов методов $z_0, \ldots, z_N$. Таким образом, объект класса T характеризуется вектором значений переменных состояния всех алгоритмов - методов класса. Переменные состояний должны иметь префикс - букву B – в имени метода. Например, Bget, Bput, Bzon и так далее. Переменные состояний принадлежат к закрытым данным – членам класса T. Эти переменные недоступны другим алгоритмам вне объекта класса T, но являются доступными всем методам данного класса. При необходимости, чтобы иметь информацию о состоянии алгоритма того или иного метода класса, необходимо предусмотреть специальный открытый метод, возвращающий при его вызове значение переменной состояния;
3. H –представляет собой набор закрытых данных членов класса T, используемых в функциях – членах класса. Эти данные запоминаются, модифицируются и сохраняются;

4. X – представляет собой подмножество разрядов вектора всех входных сигналов, получаемых методом опроса в каждом программном цикле. Номера разрядов перечисляются в множестве A параметров настройки. (Для класса измерений или вычислений множество X есть либо дискретное значение аналогового параметра, либо множество последовательных отсчетов, например, синусоидального сигнала.);
5. Y – это значения подмножества разрядов вектора всех выходных дискретных сигналов контроллера. Номера разрядов этого подмножества задаются во множестве A параметров настройки. Множество Y формируется методом ***put*** на основе результатов вычислений в остальных методах класса. Предложенная схема типового класса контроля и управления агрегатом используется в дальнейшем как база для синтеза конкретных классов контроля, управления и измерений;
6. $O_{вх}$– множество открытых данных – членов класса T, доступных для модификации из объектов других классов;
7. $O_{вых}$ – это открытые данные – члены класса T, доступные для чтения другими объектами других классов;
8. C – это набор кодов команд, приходящих из подсистемы (п/с), на которые должны реагировать конкретные объекты данного класса. Эти коды задаются и запоминаются конструктором класса. (Например, код команды, получаемой из подсистемы обозначим “***command***”). Закрытые переменные - конкретные коды команд начинаются с буквы C, например: ***Con***, ***Coff*** (включить и отключить, соответственно);
9. S – сигналы, передаваемые в подсистему. Эти сигналы могут быть двух типов. Первый тип предполагает передачу в подсистему пары: номер сообщения и обобщенный код состояния контролируемого агрегата. Номер сообщения задается одним из параметров настройки A. Сигналы первого типа передаются в подсистему в произвольном порядке от различных объектов различных классов. Второй тип сигналов предполагает их упаковку в стандартное сообщение, где код состояния конкретного агрегата записывается в фиксированную позицию, номер которой определяется одним из параметров настройки A;
10. Z – множество открытых и закрытых методов класса T.

Графическая структура класса показана на рис. 8.

Открытые и закрытые методы из множества $Z = \{z_0, z_1, \ldots, z_N\}$ (являющиеся функциями-членами класса T), могут быть переопределяемыми (виртуальными). Обязательными методами являются:

1. ***run*** – основной открытый метод (z_0), в теле которого последовательно вызываются остальные N методов класса T. Если T – базовый класс, то метод ***run*** может быть либо виртуальным, либо избыточным (вызывающим пустые функции-члены);

 ISSN 1562-9945

2. ***get*** – закрытый метод (z_1), который осуществляет вызов и запоминание входной информации (X) нижнего уровня, а также осуществляющий необходимые преобразования информации в формат для последующей обработки. Метод может быть виртуальным;
3. ***put*** – закрытый метод, осуществляющий вывод результатов обработки как на нижний уровень (Y), так и на верхний уровень (S). Метод может быть виртуальным.

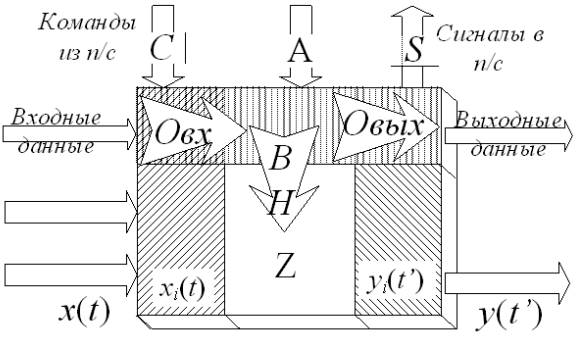


Рис. 9 – Структура класса логического управления в виде агрегата

Не обязательными, но наиболее часто используемыми являются закрытые, в том числе виртуальные, методы обработки команд (C) или условий автоматического включения - отключения ($Овх$) управляемого агрегата.

Кроме перечисленных классов, система может содержать и другие закрытые и открытые методы (подмножества множества Z), определяемые необходимостью в моделируемой системе.

Предложенный подход позволяет, работая с системой как с моделирующей программой для синтеза/анализа ГКС, осуществлять:

1. экспертный отбор технологической конфигурации системы;
2. семантическое согласование разнородных методов решения поставленных задач;
3. формирование задач-запросов в аналитическом виде (построение функционала параметров, требований и ограничений);
4. формирование задач-запросов в диалоговом режиме (пошаговое решение запросов);
5. мониторинг, как результатов, так и промежуточных данных на любом этапе выполнения поставленных задач;
6. повышение эффективности решения задач за счет дополнения или редактирования компонент, входящих в подсистемы (компоненты) моделирующей программы;

7. иерархическую абстракцию или уточнение модели для более полного изучения и решения поставленной задачи.

Выводы

1. Предложен подход к созданию автоматизированного рабочего места пользователя (АРМ) для решения задач синтеза/анализа ГКС на базе открытой объектно-ориентированной среды моделирования, позволяющей подключение специализированных модулей для решения различных задач.

2. Предложен XML-интерфейс для информационного обмена компонент объектно-ориентированной среды моделирования.

3. Дана иллюстрация объектно-ориентированного описания многокомпонентной подсистемы, на примере подсистемы “Согласования между алгоритмами и методами представления компонент”.

4. Рассмотрен объектно-ориентированный подход к формированию модулей системы на примере модуля “Логическое управление” как агрегат общего вида.

Литература

1. Банашак З., Лісовиченко О.І., Гергі Д.І., Ямпольський Л.С. Моделювання розподіленого управління перепускними спроможностями циклічних виробничих систем //Труды филиала МГТУ им. Н.Э. Баумана в г. Калуге. Специальный выпуск: Материалы международн. Науч.-техн. Конф. “Приборостроение-2001”, п. Симеиз, 17-21 сентября 2001г.-С. 158-163.

2. Банашак З.А., Лісовиченко О.І., Ткач Г.М., Ямпольський Л.С. Реалізація концепції розподіленого керування з самосинхронізацією потоків транспортних засобів ГВС// Міжвідомчий науково-технічний збірник “Адаптивні системи автоматичного управління”.- Дніпропетровськ: ДНВП Системні технології, 2001-Вип. 4’(24).- С.88-108.

3. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++. – М.: “Бином”, СПб.: “Невский диалект”. 1998. – 560 с.

4. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя//М., изд-во ДМК, 2000г.

5. Йордан Э. Структурное проектирование и конструирование программ. - М.:Мир. 1979. - 416 с.

6. Лисовиченко О.И. , Костюк В.И., Лавров А.А., Ямпольский Л.С. Интегрированная семантически согласованная среда гиперпространственного моделирования гибких компьютеризированных производственных // Автоматизація виробничих процесів.-2001р.- 2(13).- С.86-100.

 ISSN 1562-9945

7. Лисовиченко О.И., Ямпольский Л.С., Герги Д.И., Остапченко К.Б., Ткач М.М. Гиперпространственная семантика процессов в сложных агрегатированных системах // Труды филиала МГТУ им. Н.Э. Баумана в г. Калуге. Специальный выпуск: Материалы международн. Науч.-техн. Конф. “Приборостроение-2000”, п. Симеиз, 18-23 сентября 2000 г.-С. 411-416.
8. Лісовиченко О.І. , Данішевський Б.М. , Птічнікова А.С. , Ланкін Ю.М. , Ямпольський Л.С. Візуалізація моделювання гіперпросторово-розподілених гнучких комп’ютерно-інтегрованих виробничих систем Міжвідомчий науково-технічний збірник “Адаптивні системи автоматичного управління”.-Дніпропетровськ: ДНВП Системні технології, 2004-Вип. 7’(27).
9. Сігал О. Л., Лісовиченко О.І., Ямпольський Л.С. Функціонально-еквівалентні структурні перетворення складних систем // Труды филиала МГТУ им. Н.Э. Баумана в г. Калуге. Специальный выпуск: Материалы международн. Науч.-техн. Конф. “Приборостроение-99”, г.Ялта, 20-25 сентября 1999 г.-С. 159-163.
10. Швачко В.В., Поліщук М.М., Ямпольський Л.С. Розробка програмного забезпечення для визначення та вызуалізації положень ланок маніпулятора // Адаптивні системи автоматичного управління.- 1999.- Вип. 2 (22).- С. 74-81.
11. Ямпольский Л.С., Лавров А.А., Лисовиченко О.И., Сигал А.Л. Проблема автоматизации моделирования и управления в гибких гетерогенных распределенных сборочных системах // Міжвідомчий науково-технічний збірник “Адаптивні системи автоматичного управління”.-Дніпропетровськ: ДНВП Системні технології, 1999.- Вип. 2 (22).- С. 53-73.
12. "XML Schema Part 1: Structures."W3C Recommendation, 2 May 2001.
13. "XML Schema Part 2: Datatypes."W3C Recommendation, 2 May 2001.
14. Booch G., Rumbaugh J. UML 1.1 Semantics. http://www.rational.com/uml/ 1997.
15. Booch G., Rumbaugh J., Jacobson I.: Unified Modeling Language User Guide. Addison-Wesley Professional, 2005.
16. Feijs, L.M.G., Jonkers, H.B.M.: Formal Specification and Design. Cambridge University Press, 2005.
17. Michels K., Klawonn F., Kruse R., Nürnberger A.: Fuzzy Control: Fundamentals, Stability and Design of Fuzzy Controllers. Springer, 2006.
18. Yampolsky L.S., Banashak Zb., Lisovichenko O.I., Ostapchenko K.B. Towards distributed control of manufacturing systems // Вестник СевГТУ. Вып. 27: Автоматизация процессов и управление: Сб. науч. тр. / Редкол.: В.Н. Торлин (отв. ред.) и др.; Севастоп. гос. техн. ун-т.- Севастополь, 2000.-С. 80-87.