# Мультиагентная модель открытой системы ЧПУ типа PCNC

В.Л. Сосонкин, Г.М. Мартинов (МГТУ "СТАНКИН")

*Показано, что математическое обеспечение современной системы ЧПУ носит характер распределенного комплекса программно-реализованных модулей, приспособленных к автономной работе и взаимодействию между собой. Моделью математического обеспечения такой системы может послужить мультиагентная структура, в которой коммуникационные команды оказываются событиями в межагентном интерфейсе. При реализации математического обеспечения ЧПУ на основе такой модели можно получить преимущества, которые вытекают из повышенного быстродействия распределенных потоковых систем.*

## Введение

В числе важнейших требований к современным системам ЧПУ типа PCNC (Personal Computer Numerical Control) можно отметить открытый характер архитектуры, возрастающую роль периферии, четкую спецификацию внутренних и внешних коммуникационных интерфейсов, возможность параллельного многоканального воспроизведения разных управляющих программ при работе с несколькими объектами [1]. При этом математическое обеспечение системы ЧПУ все более приобретает черты распределенной системы программно-реализованных модулей (или функциональных блоков), способных работать автономно и независимо, а также и взаимодействовать между собой. В международных проектах OMAC (http://www.arcweb.com/omac/), OSACA (http://www.osaca.org), JOP (http://www.mstc.or.jp/jop) были предприняты попытки разработать модульную архитектуру и предложить компонентный подход к реализации отдельных модулей. Так, на рис.1 показана модульная структура математического обеспечения системы ЧПУ согласно концепции OMAC.

Модульная структура системы ЧПУ в проекте OSACA представлена на рис. 2. Архитектурные модули распределенной системы дополнительно декомпозированы на функциональные блоки.

Межмодульное взаимодействие осуществляется соответственно клиент-серверной технологии (master-slave) или по типу "каждый с каждым" (peer-to-peer). Работа системы в целом может протекать путем последовательной активизации функций (method-driven) аналогично традиционным объектно-ориентированным моделям; или путем активизации событий (event-driven), как в мультиагентных (multi-agent) или холонических (holonic) распределенных системах (www.agentbuilder.com; domino.iec.ch; http://www.holobloc.co; www.agentec.de; www.ifm.eng.cam.ac.uk; www.functionblocks.org; www.opencybele.org) [2]. Далее рассмотрим возможность использования мультиагентного подхода при построении моделей открытых систем ЧПУ. Заметим, что этот подход вполне совместим с объектно-ориентированным программированием.

## Открытые системы ЧПУ

Основными признаками открытой системы ЧПУ являются (http://www.mdsi2.com/):

- интеграция SoftCNC (ядра системы ЧПУ), SoftPLC (встроенного программно-реализованного контроллера электроавтоматики [3]), интерфейса оператора и БД в единой системе;
- многоканальное управление, использующее копии основного математического обеспечения; публикация API-функций интерфейса пользователя и подсистемы РВ;
- поддержка SERCOS-интерфейса (http://www.sercos.com), Profibus, DeviceNet и др.;

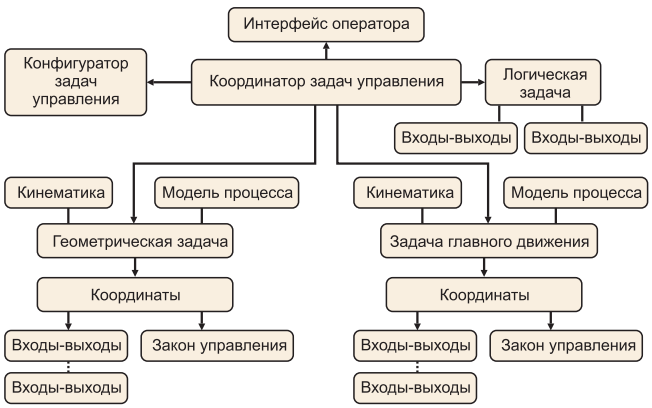



*Рис. 1. Модульная структура системы ЧПУ согласно концепции OMAC*

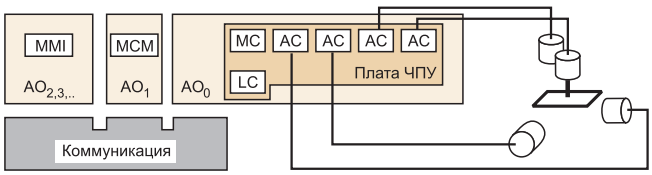



*Рис. 2. Декомпозиция системы ЧПУ на функциональные блоки в проекте OSACA,*
*где MMI (Man-Machine-Interface) – интерфейс оператора; MCM (Motion Control Manager) – диспетчер каналов; MC (Motion Controls) – ядро ЧПУ (интерпретатор-интерполятор); AC (Axis Controls) – контроллер осевого перемещения; SC (Spindle Controls) – контроллер шпинделя; LC (Logic Controller) – контроллер автоматики; AO (Architectural Object) – модуль системы*

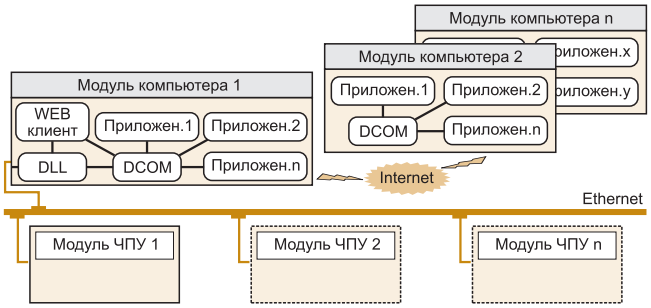



*Рис. 3. Базовая архитектура математического обеспечения ЧПУ*

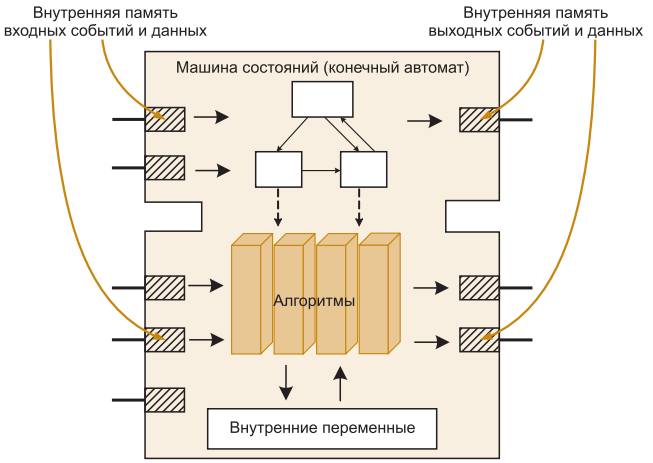



*Рис. 4. Структура агента в стандарте IEC-61499*

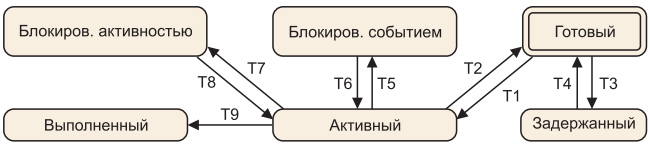



*Рис. 5. Конечно-автоматная модель состояний агента*

• единая ОС и Intel-архитектура процессора, стандартная платформа ПК;

• поддержка языков электроавтоматики согласно стандарту IEC-61131-3 (http://www.PLCopen.org).

Интерфейсные функции API (Application Programming Interface) открывают доступ к библиотекам и инструментальным средствам, которые позволяют расширить существующую систему. Существует возможность интегрировать в систему пакеты MES (Manufacturing Execution System), строить собственный интерфейс пользователя и разрабатывать свои приложения в составе математического обеспечения. Открытая БД РВ показывает текущее состояние планировщика программ и значения системных переменных. Интерфейс программ позволяет читать, использовать, менять переменные, которые могут быть адресованы со стороны SoftPLC и процессов OpenCNC.

Все подсистемы открытой системы ЧПУ (интерфейс оператора, ядро, контроллер SoftPLC) реализованы программно на базе компьютерной Windows-платформы с расширением РВ. Преимущества программной реализации подсистемы формообразования объединены с достоинствами цифровых приводов подачи. Ядро допускает расширение, поддержанное мощной инструментальной системой, позволяющей создавать как новые функции ядра, так и интерфейс оператора.

Анализ архитектурных вариантов систем ЧПУ типа PCNC [4] показывает, что выделение автономного терминального модуля MMI превалирует вне зависимости от построения программно-аппаратной платформы. Это связано с проблемами РВ и особенностями ОС для терминальной задачи и ядра ЧПУ. Таким образом, базовой архитектурой является та, которая показана на рис. 3. Моделью математического обеспечения такой системы может послужить распределенная мультиагентная структура, которая станет основой для новых и более прогрессивных реализаций.

## Агентные модели распределенных систем в стандарте IEC-61499

Понятие агента определено через событийно-управляемую (event-driven) сущность, способную к автономной работе и коммуникации. Агенты в стандарте IEC-61499 наследуют свойства функциональных блоков стандарта IEC-61133-3, но обладают дополнительными свойствами (рис. 4).

Агент инкапсулирует данные и методы. Подавляющее большинство методов относится к типу private и не могут быть вызваны извне. При этом нет ограничений на способ реализации таких методов. Остальные методы типа public могут быть инициированы только событийными объектами. Другими словами, методы типа public предлагают другим агентам канал для доступа команд-событий.

Именно свойство автономности агента запрещает обращение к нему с вызовом метода. Агент интерпретирует событие и действует соответственно результатам интерпретации, а также может посылать события самому себе.

Характер активности агента определяется его текущим состоянием, в которое он переходит по событию. Можно представить, например, шесть обобщенных состояний агента (рис. 5):

1. готовый (runnable) – начальное состояние, в котором агент ожидает прихода событий;

2. активный (active) – активное состояние, в котором метод агента работает с данными;

3. задержанный (hold) – агент может быть задержан другим агентом, после чего может вернуться только в начальное состояние;

4. блокированный событием (event-blocked) – поток, в котором работает агент, может быть заблокирован, если агент посылает синхронное событие другому объекту и ожидает подтверждения;

5. блокированный активностью (activity-blocked) – агент "а" ожидает завершения работы агента "б";

6. выполненный (done) – работа агента завершена, если он не ожидает никаких других событий.

Агент может инкапсулировать в себе другие агентные структуры. Таким образом, существует возможность де-

композиции сложных модулей на более простые (функциональные блоки являются реализациями типов "функциональные блоки", специфицированных в стандарте IEC 61499-1). Все это позволяет выстроить сложную и прозрачную программно-реализованную структуру.

Модель приложения, соответствующая стандарту IEC 61499, может представлять собой мультиагентную сеть функциональных блоков, объединенных потоками событий и данных (рис. 6).

Современные технологии предполагают использование одноагентных и мультиагентных систем. В одноагентных системах агент выполняет свою задачу от имени пользователя или другого процесса, взаимодействуя при этом с пользователем или локальными и удаленными ресурсами, но не с другими агентами. В мультиагентных системах агенты активно взаимодействуют между собой, решая те проблемы, которые находятся за пределами их собственных возможностей. Распределенная система представляет собой, таким образом, сообщество агентов.

Мультиагентные реализации сложных систем доказали свои преимущества и получают все большее распространение в распределенных системах самого разнообразного назначения, в том числе в реконфигурируемых системах и системах с элементами искусственного интеллекта. Базовым свойством таких систем называют agile, что означает одновременно высшую эффективность и гибкость.

### Мультиагентная модель системы ЧПУ

Для построения мультиагентной модели открытой системы выделим в составе математического обеспечения ЧПУ базовые модули по терминологии OSACA: интерфейс оператора MMI, ядро ЧПУ MC, контроллер автоматики LC (рис. 7 а). При этом ядро имеет сложную структуру и инкапсулирует архитектурные объекты второго порядка (рис. 7, б).

В соответствии с архитектурой OSACA в межмодульном пространстве размещены интерфейсные коммуникационные объекты API трех категорий: объекты-переменные, поддерживающие доступ к данным; объекты-процессы для управления переходами в конечном автомате модуля; объекты-события, подтверждающие завершение работы объектов-переменных и объектов-процессов. Все эти объекты в архитектуре OSACA специфицированы и предложены в качестве международного стандарта.

В интерфейсе MMI – LC поддерживается доступ к таймерам, маркерам, счетчикам, блокам данных.

Движение информации через границу MMI – MC выглядит следующим образом. Любое приложение, входящее в состав MMI, может запрашивать данные ядра ЧПУ для чтения через сервер OCI (Open Control Interface). Запросы могут быть ручными или автоматическими. Автоматически запрашиваемые данные вносятся в процессе управления в "список наблюдения" (watchlist). Запрашиваемые вручную данные вносятся в тот же список на время получения данных. Любое приложение, входящее в состав MMI, может

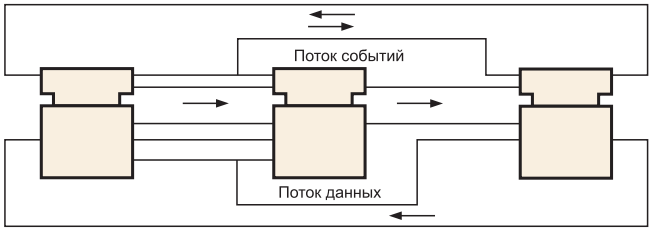



*Рис. 6. Модель приложения соответственно стандарту IEC 61499*

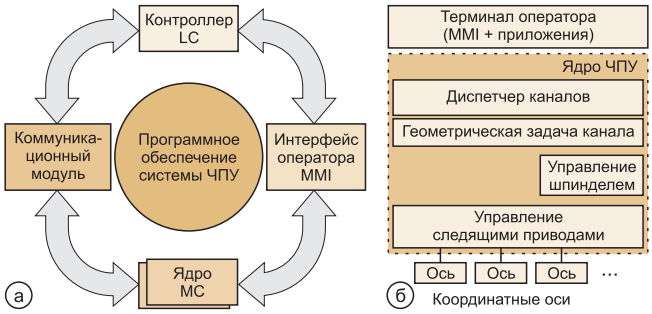


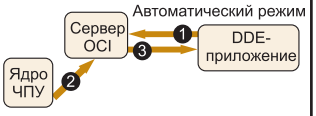
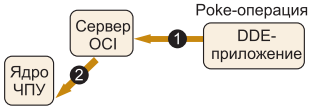

*Рис. 7. Базовые модули ПО системы ЧПУ: а – общая структура; б – структура ядра*

запрашивать данные ядра ЧПУ для записи. Информация записывается в ядре ЧПУ с помощью Poke-операции (сохранение в памяти). Если приложение MMI запрашивает Poke-операцию, то OCI-сервер обновляет соответствующие данные в ядре. Наконец, приложение может обратиться к ядру с запросом на выполнение команды (например: загрузить управляющую программу, удалить управляющую программу из соответствующей директории).

Сервер OCI повторяет запросы, если процессор работает, но перегружен. Если сервер обнаруживает критическую ошибку (например, нет ответа процессора), то коммуникация приостанавливается на время задержки. По истечении этого времени сервер вновь попытается связаться с процессором. Если и эта попытка окажется неудачной, то сервер предпримет новую попытку после устранения отказа.

Структура транзакций, поддерживаемых сервером OCI, показана в таблице.

*Таблица. Структура транзакций, поддерживаемых сервером OCI*

| Запрос данных для чтения | Запрос данных для записи |
|---|---|
|   |   |
| 1. DDE-совместимое приложение запрашивает данные у OCI-сервера. Сервер, в свою очередь, обращается к ядру для внесения данных в "список наблюдения" watchlist.<br>2. Ядро ЧПУ вносит данные в список наблюдения и посылает текущие их значения OCI-серверу.<br>3. Ядро посылает запрошенные данные до тех пор, пока их значения будут меняться, и данные будут оставаться в "списке наблюдения". OCI-сервер передает данные приложению MMI. | 1. DDE-совместимое приложение посылает значения структуры данных в OCI-сервер.<br>2. Сервер передает эти значения структуре данных в ядре. |

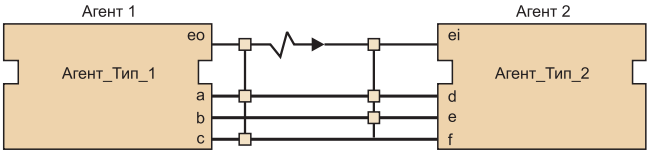



*Рис. 8. Структура двухагентной системы MMI – MC*

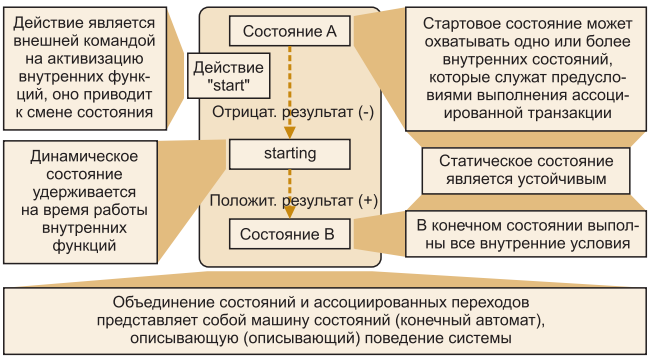



*Рис. 9. Формальное представление конечного автомата агента*

В формате данных при их запросе имеются поля: тип данных, указатель возможности чтения-записи, индекс массива (если данные представлены массивом).

В мультиагентной архитектуре коммуникационные структуры данных и команд приобретают иной смысл. Все они сохраняют свои наименования и смысловое наполнение (семантику), но оказываются событиями в межагентном интерфейсе. Их назначением становится управление переходами конечных автоматов агентов. В свою очередь, состояния конечных автоматов порождают работу агентных методов и требуемые (запрашиваемые событиями) потоки данных.

Структура двухагентной системы MMI – MC показана на рис. 8, а структура трехагентной системы MMI – MC – LC соответствует рис. 6. При этом каждый агент может принадлежать уровню, допускающему иерархические агентные вложения.

Важным компонентом каждого агента в системе ЧПУ служит конечный автомат (state machine), изменяющий свои состояния под влиянием событий. Формальное представление конечного автомата агента приведено на рис. 9. Состояния автомата инициируют работу методов агента, которые, в свою очередь, в процессе работы порождают внутренние данные и внешние межагентные потоки данных. Между любыми двумя устойчивыми статическими состояниями конечного автомата можно зафиксировать динамическое состояние, в котором оценивается положительный или отрицательный эффект работы события.

В архитектуре OSACA всякому архитектурному объекту сопоставлен универсальный конечный автомат, (рис. 10). Подобная же структура может быть принята за основу при построении конечного автомата агента.

В реальных агентах математического обеспечения ЧПУ конечные автоматы обретают разные структуры. Так, на рис. 11 показаны конечные автоматы агентов MMI и MC. Двойной контур в состояниях автомата агента MMI указывает на иерархическое вложение подрежимных конечных автоматов [1]. Состояние "selected" конечного автомата агента MC представляет собой текущее состояние конечного автомата агента MMI. Таким образом, агент MC реагирует только на те события, которые и соответствует текущему состоянию конечного автомата агента MMI.

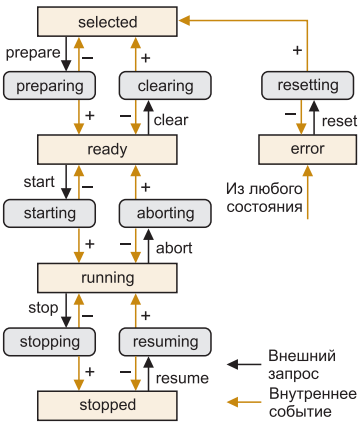



*Рис. 10. Обобщенная структура конечного автомата архитектурного объекта OSACA*

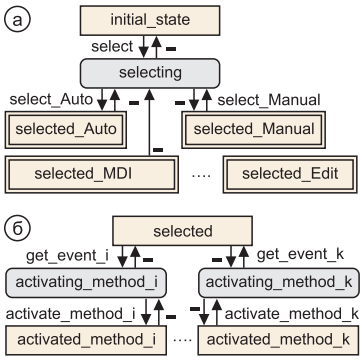



*Рис. 11. Конечные автоматы агентов MMI (а) и MC (б)*

## Заключение

Достоинства мультиагентных структур в распределенных реконфигурируемых системах получили достаточно убедительные доказательства. Для систем ЧПУ особенно важны такие свойства подобных структур, как прозрачность и масштабируемость, которые наилучшим образом соответствуют требованиям открытой архитектуры. Мультиагентные системы наследуют те решения, которые были сформулированы в проекте OSACA. При этом следует принять во внимание дополнительные преимущества, которые вытекают из повышенного быстродействия распределенных потоковых систем.

## Список литературы

1. *Сосонкин В.Л., Мартинов Г.М.* Системы числового программного управления. М.: Логос. 2005.
2. *Елькин И.В., Кустарев П.В.* Модель абстрактных функциональных блоков // Научно-технический вестник СПб ГИТМО (ТУ). Выпуск 10. Информация и управление в технических системах. – СПб.: СПб ГИТМО (ТУ), 2003.
3. *Сосонкин В.Л., Мартинов Г.М., Перепелкина М.М.* Концепция управления электроавтоматикой станков с ЧПУ по типу виртуальных контроллеров SoftPLC // Приборы и системы. Управление, диагностика, контроль. 2003. №7.
4. *Сосонкин В.Л., Мартинов Г.М.* Новейшие тенденции в области архитектурных решений систем ЧПУ // Автоматизация в промышленности. 2005. №4.

**Сосонкин Владимир Лазаревич** – д-р техн. наук, профессор, **Мартинов Георгий Мартинович** – д-р техн. наук, доцент Московского государственного технологического университета "СТАНКИН". Контактный телефон (499) 972-94-40. E-mail: book@ncsystems.ru Http: www.ncsystems.ru