Г.М. МАРТИНОВ, д-р техн. наук,
Р.Л. ПУШКОВ, аспирант

Построение инструментария отладки управляющих программ систем ЧПУ на языках высокого уровня

Предложен метод создания инструментария отладки управляющих программ систем ЧПУ на языках высокого уровня. Проанализированы основные возможности языков высокого уровня, сформированы требования к инструментарию отладки, выделен набор визуальных компонентов для его разработки.

The article suggests a method of debugging tools' designing for CNC part programs in numerical control systems implemented by high-level language. It analyzes main options of high level languages, formulates the requirements for debugging toolkit and defines the designing set of visual components.

Введение

В современных системах ЧПУ языки высокого уровня используются для написания стандартных и собственных циклов управляющих программ, создания подпрограмм и разработки групповых технологий обработки.

Языки высокого уровня для разработки управляющих программ обладают всеми возможностями языков четвертого поколения для ПК, а значит сложность процесса разработки и отладки управляющих программ для систем ЧПУ сопоставима со сложностью аналогичного процесса для ПК. При разработке программ для ПК используется хорошо развитый инструментарий отладки, в который входят, например, средства для контроля хода выполнения программы и средства отслеживания переменных. В отличие от ПК, в системах ЧПУ средства отладки практически отсутствуют (за редкими исключениями, например, системы *Typ3osa* фирмы *Bosch*). Максимум, что может предоставить интерфейс оператора системы ЧПУ – это предварительное моделирование обработки в простейшем симуляторе или отслеживание текущих координат при выполнении управляющей программы с отключенными приводами.

Представление о языках для систем ЧПУ

Стандарт *ISO-7bit* определяет формат построения управляющих программ для систем ЧПУ. Управляющая программа содержит команды перемещения, установки и смещения системы координат, коррекции на радиус и длину инструмента, установки подачи, выбора инструмента и команды управления электроавтоматикой.

Управляющие программы, хотя и подчиняются единому стандарту, практически не переносимы с одного станка на другой. Это объясняется тем, что производители систем ЧПУ адаптируют стандарт для своих нужд. К тому же, трудоемкость создания программ для деталей со сложным контуром, а также повторяющимися элементами контура, остается очень большой.

Языки высокого уровня следует рассматривать как средство повышения открытости систем ЧПУ. С помощью этого механизма, не изменяя ядра, станкостроители встраивают собственные станочные и измерительные циклы, а конечные пользователи разрабатывают групповые технологии обработки с помощью подпрограмм.

Выявлены признаки языка управляющих программ, позволяющие отнести его к языкам высокого уровня:

1) использование переменных, имеющих тип и размерность, в т. ч. и их массивов;

2) проведение математических и строковых операций с переменными, а также реализация функции трансформации системы координат;

3) вызов подпрограмм и возможность оформления фрагмента кода в виде подпрограммы с параметрами вызова и возвращением результата;

4) организация циклов (многократно повторяющихся участков управляющей программы), условных и безусловных переходов;

5) подключение дополнительных функциональных библиотек для проведения сложных вычислений и расширения алгоритмов интерполяции;

6) поддержка файловых операций.

Отсутствие единого стандарта обусловливает разные степени реализации языков высокого уровня со стороны производителей систем управления.

Процесс отладки и поиска ошибок чрезвычайно сложен, что объясняется разветвленностью алгоритма управляющей программы, наличием специфичных языковых конструкций и использованием переменных. При анализе возможностей языков управляющих программ различных систем ЧПУ были взяты системы ЧПУ известных производителей *Siemens*, *GE Fanuc*, *Andronic*, *BoschRexroth*. При выборе систем ЧПУ для анализа учитывалось, что *Siemens* [1] и *Fanuc* [2] являются наиболее распространенными системами ЧПУ, *Andronic* [3] предназначена для выполнения специализированных задач высокоскоростной обработки скульптурных поверхностей, а системы *BoschRexroth* [4] построены по двухкомпьютерной схеме и являются классическим примером в области систем ЧПУ с открытой архитектурой. В таблице систематизированы возможности языков высокого уровня, реализованных в системах ЧПУ.

Язык *Anlog-C* предоставляет практически неограниченные возможности для создания управляющих программ в системе *Andronic*, но сложность этого языка и предъявляемые завышенные требования к квалификации разработчиков делают его непопулярным среди специалистов.

BoschRexroth предлагает приложение отладки *CPL-Debugger* (*Custom Program Language Debugger* – отладчик языка программирования пользователя), но он ориентирован только на работу с языком *CPL*.

Стандарт *STEP-NC ISO 14649* является альтернативой *ISO-7bit*, призванной исправить его основные недостатки. В основе *STEP-NC* была заложена идея прямого обмена информацией между системами *CAD*/*CAM*/*NC*. Стандарт *STEP-NC ISO 14649* определяет специальную структуру управляющей программы ЧПУ, которую используют для построения логических блоков в рамках структурного программирования обработки. Структура управляющей программы не является списком типовых обрабатываемых форм; она определяет план операций, который представляет собой последовательность

Ключевые возможности языков высокого уровня

Возможности	Системы ЧПУ			
	Siemens	*GE Fanuc*	*Bosch Rexroth*	*Andronic*
Типы переменных	*INTEGER*, *REAL*, *BOOL*, *CHAR*, *STRING*, *AXIS*, *FRAME*	*REAL*	*INTEGER*, *REAL*, *DOUBLE*, *BOOLEAN*, *CHARACTER*	*INTEGER*, *FLOAT*, *CHAR*, *POINTER*
Работа с массивами	1-, 2-мерные	—	1-, 2-мерные	1-, 2-, многомерные
Уровень вложенности подпрограмм	11	4	8	В *Anlog-C* неограничен
Модальный вызов подпрограмм	+	+	+	—
Вызов подпрограмм, не загруженных в память системы ЧПУ	+	—	+	—
Работа с макросами	+	—	—	частично
Реализуемые виды циклов	*LOOP*, *FOR*, *WHILE*, *REPEAT*	*WHILE*	*FOR*, *WHILE*, *REPEAT*	*WHILE*
Поддержка ветвлений (условных переходов)	*IF*, CASE	*IF*	*IF*, *CASE*	По стандарту *ANSI-C*. В управляющей программе не поддерживаются
Допустимая вложенность циклов и переходов	8 (в пределах каждой подпрограммы)	5	10 для *CASE*	В *ANSI-C* неограничено.
Арифметические и логические операции	+	+	+	+
Операции со строками	+	—	+	+
Расширенная адресация	+	—	+	+
Трансформация систем координат	+	—	+	+
Файловые операции	+	—	+	Только в *Anlog-C*
Многоканальное программирование и синхронизация	+	—	+	Только в *Anlog-C*
Средства отладки программ	*Dry-Run* режим. Моделирование в встроенном графическом эмуляторе	*Dry-Run* режим.	*Dry-Run* режим, *CPL-Debugger*	*Dry-Run* режим. Отладка программы на *Anlog-C* в специальной среде программирования

исполняемых объектов. Кроме того, возможны: свободная организация процесса обработки, параллельные структуры, контуры, условные переходы и др. Исполняемые объекты в составе плана операции инициирует активность станка. Существуют три типа исполняемых объектов: собственно план операции; функция ЧПУ; шаг операции. Шаг операции описывает процессы, в которые вовлечены интерполируемые координатные оси. В отличие от этого, функции ЧПУ сопоставлены единичным событиям и с интерполяцией не связаны.

Изделие получают из заготовки путем: удаления типовых форм (*features*); условного или безусловного выполнения ассоциированных с типовыми формами шагов операции (*working steps*) в потоке управления, задаваемом исполняемыми объектами (*executables*); с необходимыми допусками; использования инструмента, отвечающего всем необходимым требованиям. Предполагается, что система управления способна интерпретировать подобную информацию и генерировать необходимые перемещения и циклы.

Язык *STEP-NC* применяется только в экспериментальных системах и его будущее ставится под сомнение большинством крупных производителей [5]. *STEP-NC* нельзя рассматривать в традиционном представлении языков высокого уровня, его надо воспринимать как потенциальную альтернативу сегодняшних решений.

Базовые понятия процесса отладки

Рассмотрим основные термины и определения, характеризующие процесс отладки управляющих программ. Большинство из этих понятий имеют аналоги в области программирования [6].

Отладка (debugging) – этап разработки управляющей программы, на котором обнаруживают, локализуют и устраняют ошибки. В качестве основного средства отладки используется программный инструментарий – отладчик *(debugger)*, который включает в себя пользовательский интерфейс для пошагового выполнения программы: строка за строкой, команда за командой, с остановками на некоторых строках программного кода

или при достижении определённого условия. Он также дает возможность отображать значения переменных, устанавливать точки или условия останова. Отслеживать правильный ход выполнения программы можно путем установки в критических её частях отладочных операторов вывода трассировочной информации о процессе.

Функциональность отслеживания значений переменных (*watching*) предоставляет возможность просматривания значений переменных на любом этапе выполнения программы и их изменения в момент достижения точки останова.

Стек вызовов (*callstack*) – *LIFO*-стек (*Last In First Out*) хранит информацию об активных процедурах, т.е. процедурах, которые были вызваны, но не завершились возвратом в вызвавшую процедуру. Стек вызовов позволяет отслеживать место, куда каждая из активных процедур должна вернуть управление после своего завершения.

Точка останова (*breakpoint*) – осуществляет преднамеренное прерывание выполнения программы, при котором вызывается отладчик. В отладчике программист может исследовать текущее состояние программы (например, значения переменных, стек вызова). После анализа программа может быть завершена либо продолжена с места останова.

Условные точки останова определяют одно или несколько условий, при которых происходит прерывание программы. Наиболее часто практикуется условие останова при вызове указанной инструкции программы (*instruction breakpoint*). Другой пример – останов при операции чтения/записи переменной или выход значения за пределы определенного диапазона (*data breakpoint* или *watchpoint*).

Точка трассировки (*tracepoint*) – схожа с точкой останова, но в отличие от нее не прерывает выполнение управляющей программы, а передает управление специальной процедуре трассировки, после выполнения которой продолжается исполнение управляющей программы со следующего кадра. Процедура трассировки может выводить информацию в окно сообщений, вести журнал или осуществлять подсчет количества проходов. Целесообразно использование точки трассировки, когда в прерывании работы управляющей программы нет необходимости или прерывание нарушает состояние процесса отработки программы. Например, выполнение каких-то кадров может зависеть от времени, прошедшего с определенного момента.

Трассировка (*trace*) – пошаговое выполнение программы с остановками на каждой команде. Этот режим выполнения позволяет просматривать состояние программы и значения переменных после каждого кадра программы.

Архитектура инструментария отладки управляющих программ для систем ЧПУ

Открытая архитектура систем управления характеризуется наличием коммуникационной среды и документированных *API*-функций (*Application Programming Interface*), позволяющих взаимодействовать с ядром и реализовать, в т. ч. функциональность отладки [7]. Продвинутые системы ЧПУ поддерживают на уровне ядра установку точек останова и отслеживание переменных. В остальных ЧПУ подобные механизмы можно реализовать при помощи режимов пошагового выполнения программы, выполнения блока программы и чтения данных системы, в т. ч. текущих значений переменных и параметров системы.

Построение универсального инструментария отладки управляющих программ возможно при использовании принципа модульности и открытости, что позволит адаптировать его для конкретных прикладных задач.

Выбор набора компонентов определяется функциональностью системы. С одной стороны, компоненты привязаны к прикладным функциям, которые реализуют, с другой стороны, должны быть достаточно независимыми, чтобы можно было свободно компоновать приложения с минимумом изменений в коде. Это означает, что компоненты должны иметь общие механизмы для осуществления связи между ними и обработку ошибок и исключений.

Базовая архитектура инструментария отладки управляющих программ представлена на рис. 1.

Выделен *сервер языка управляющей программы*, который маскирует особенности конкретной системы ЧПУ и реализует специфицированный набор интерфейсов для

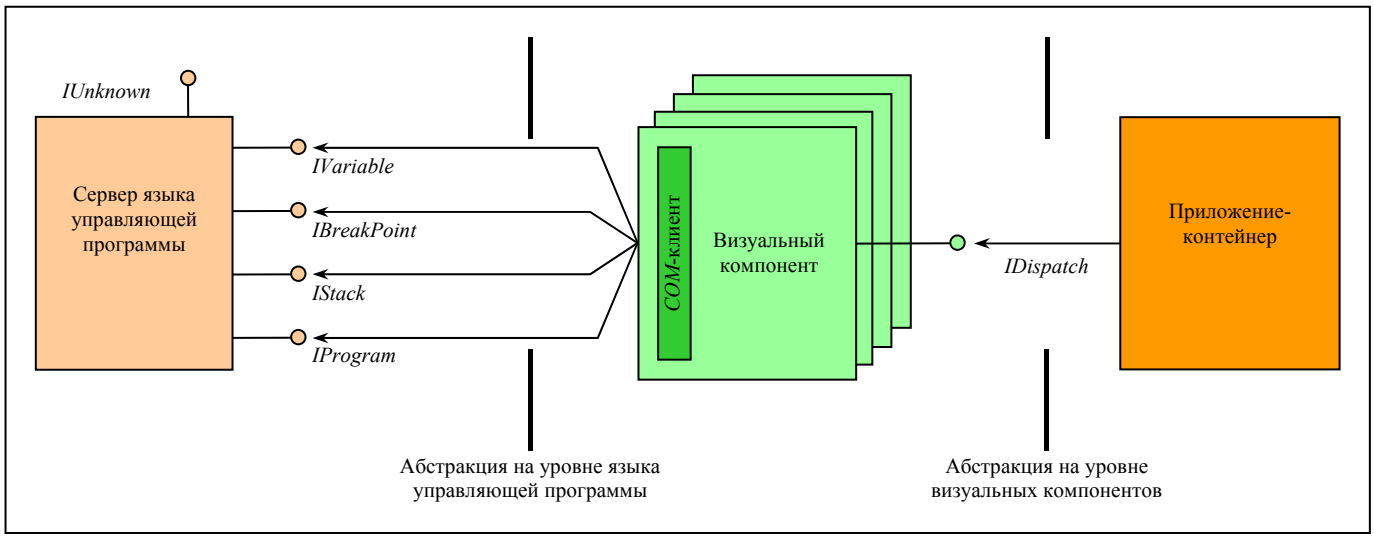

Рис. 1. Базовая архитектура инструментария отладки

интеграции в системе. Сервер языка управляющей программы разрабатывается для каждой системы ЧПУ, используя ее возможности и специфики, при необходимости дополняя систему ЧПУ нужными для отладки функциональностями, которые могут не поддерживаться на уровне ядра. Реализация стандартизированного внешнего интерфейса позволяет заменить серверную часть одной системы ЧПУ на другую, без затрагивания остальных компонент инструментария отладки.

Контейнер определяет, будет ли инструментарий отладки встроен в интерфейс оператора или будет реализован как отдельное приложение.

Визуальные компоненты формируют пользовательский интерфейс инструментария и предоставляют функциональные возможности отладки.

На рис. 2 показано, как производится выделение набора компонентов при помощи *Use-Case* диаграммы прецедентов инструментария отладки [8].

Группирование прецедентов по их функциональному назначению позволяет выделить компоненты приложения. Таким образом, навигацию, работу с переменными и точками останова при чтении/записи переменных осуществляет *компонент дерева навигации*. Отслеживание текущего значения переменной и его изменение закреплено за *компонентами синхронного и асинхронного отслеживания значений переменных*. Вывод сообщений для оператора, а также протокол работы и возникновения ошибок ведется в *окне вывода*. Отображение, редактирование управляющей программы, а также работа с точками останова на строках программы осуществляется *редактором управляющей программы*. *Компонент стека* служит для просмотра стека вызовов и навигации по уровням управляющей программы. *Компонент файловой системы ядра* позволяет работать с файлами ядра системы ЧПУ.

Внешний вид интерфейса компонентов, их разложение в инструментарии отладки и ключевые функциональности приведены на рис. 3.

Предложенный список компонентов реализует ключевые функциональности инструментария отладки. Список может быть изменен или расширен без затрагивания существующих компонентов.

Реализация программного обеспечения

Учитывая специфику систем управления и современные тенденции разработки ПО, необходимо находить компромисс при смешивании программных технологий и применении готовых решений.

Разработка сервера языка в виде *COM*-компонента (*Component Object Model* – объектная компонентная модель) позволяет абстрагироваться от *API*-функций (*Application Programming Interface* – интерфейс программирования приложений), поставляемых разработчиками систем ЧПУ.

Реализация *COM*-сервера на управляемом или неуправляемом коде определяется, исходя из целесообразности, для каждого конкретного решения.

Реализация визуальных компонентов на *C#* позволяет, во-первых, быть независимым от платформы исполнения, что предлагает .NET, во-вторых, использовать

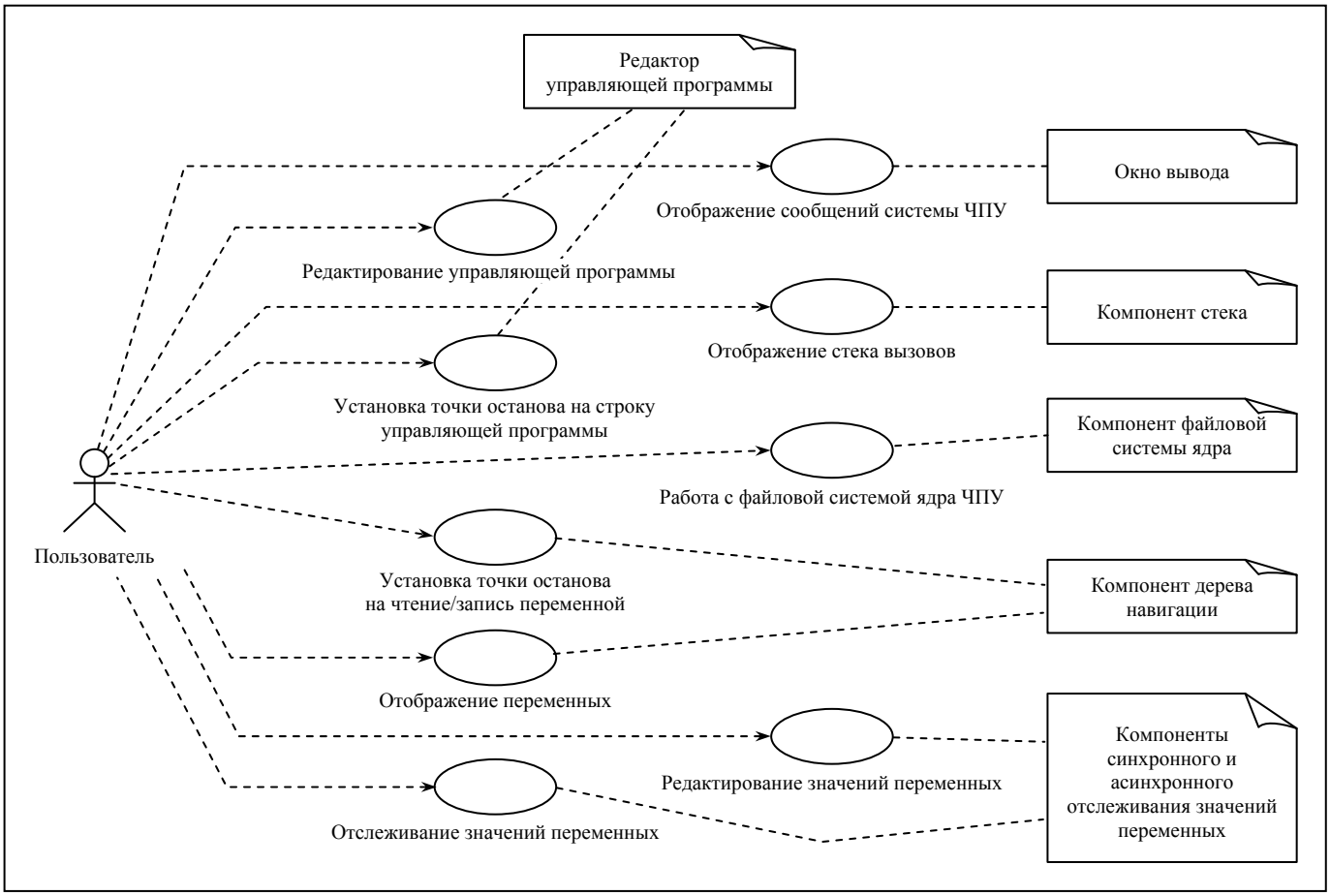

Рис. 2 Определение компонентов по Use-Case диаграмме функциональностей

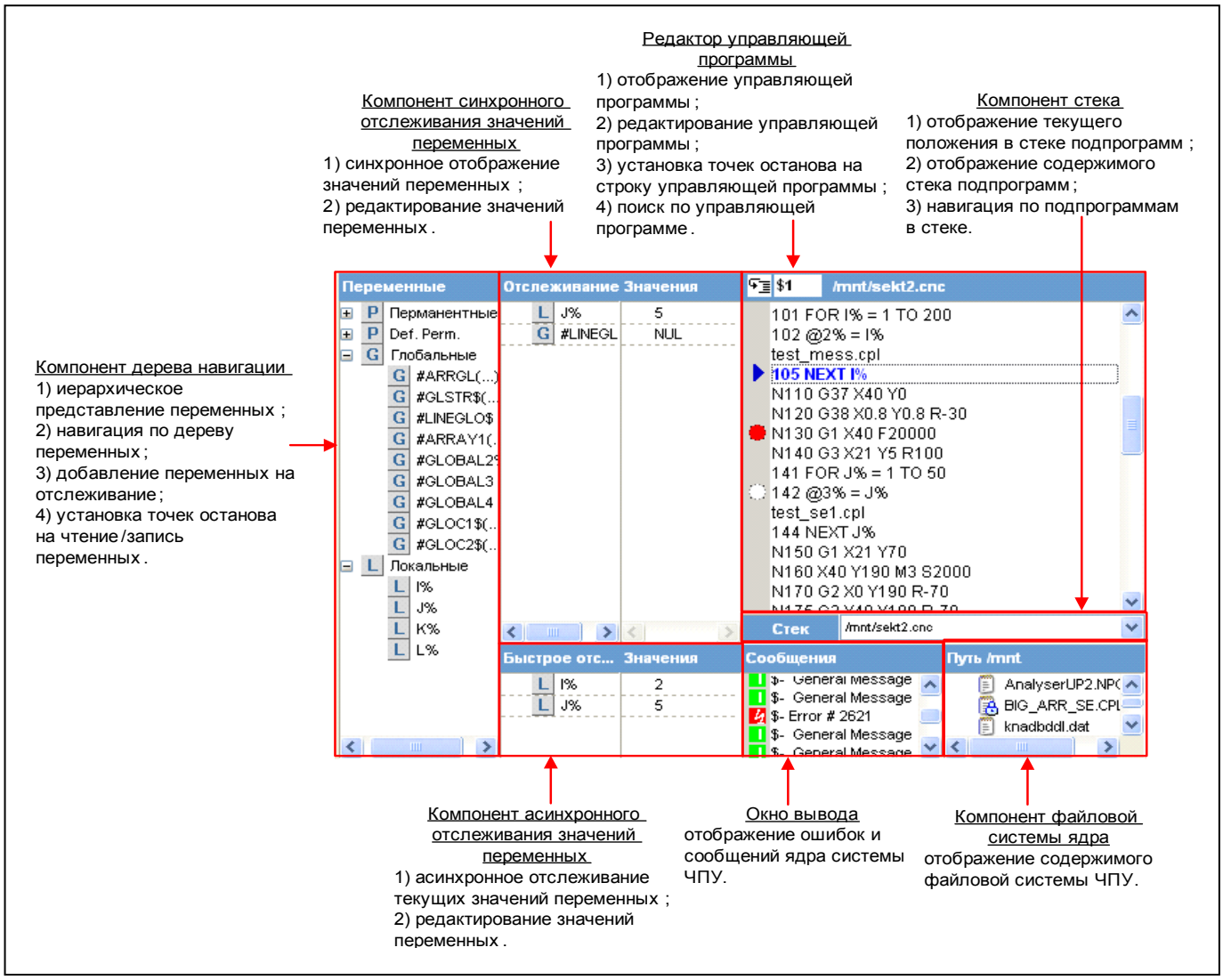

Рис. 3. Внешний вид интерфейса компонентов инструментария отладки

большое количество коммерческих решений для этой платформы, что в итоге сокращает время разработки.

Наши исследования показали, что в качестве базового компонента разработки для всех визуализируемых компонентов инструментария целесообразно использовать *ComponentOne FlexGrid.NET* [9]. Этот компонент обладает хорошим быстродействием и широкими возможностями настройки. С помощью *FlexGrid.NET* строится иерархическое представление переменных в виде дерева, реализуются таблицы для отслеживания и редактирования значений переменных, создаются многострочные редакторы для отслеживания управляющей программы и расстановки точек останова, а также формируется стек вызовов подпрограмм.

Предложенное решение построения инструментария отладки обеспечивает унифицированный вид и функциональность всех визуальных компонентов экрана, поскольку они все построены на базе *FlexGrid.NET*. Адаптация инструментария для системы ЧПУ *WinPCNC* [10] представлена на рис. 4. Суть адаптации заключается в разработке и подключении *COM*-сервера языка управляющих программ для системы *WinPCNC*. Все визуальные компоненты остаются без изменений. Без изменений остается и контейнер, реализованный в виде

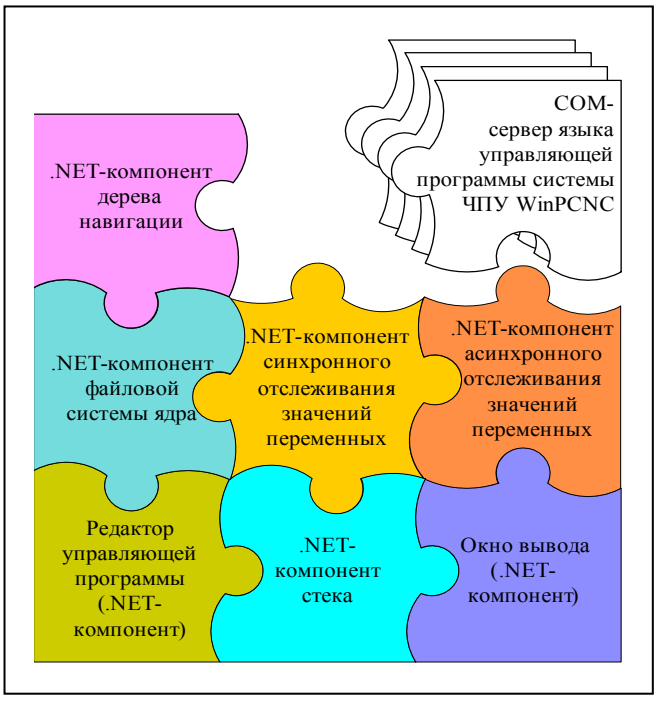

Рис. 4. Адаптация инструментария отладки к системе ЧПУ WinPCNC

независимого приложения. В случае встраивания инструментария отладки в интерфейс оператора *WinPCNC* необходимо производить интеграцию визуальных компонентов.

Выводы

Разработка современных управляющих программ и стандартных циклов на языках высокого уровня невозможна без специального инструментария отладки.

Применение компонентной архитектуры при построении отладчика управляющей программы позволит адаптировать его с минимальными затратами практически на любую версию системы ЧПУ.

Использование отладчика управляющих программ с хорошо продуманной и структурированной архитектурой позволяет сократить время разработки и отладки управляющих программ.

Работа выполнена в МГТУ "СТАНКИН".

Контактный телефон (499) 972-94-40.
E-mail:book@ncsystems.ru

Список литературы

1. *Siemens* SINUMERIK 840D/840Di/810D Advanced Programming Guide, *http://www.automation.siemens.com/doconweb/*
2. *GE Fanuc* Automation Series 16i/18i/160i/180i – Model A. Operation and Maintance Handbook. GE Fanuc Automation North America, Inc., 1997, *http://www.gefanuc.com*
3. *Andron* Andronic 400 Reference Manual. Andron GmbH, 1995, *http://andronic.com*
4. *CPL* Programming Manual. Robert Bosch GmbH, Erbach/Germany, 2000.
5. *http://ingener.info/index.php*?name=News&op=article&sid=15
6. *Microsoft* Software Developer Network, *http://msdn.microsoft.com*
7. *Сосонкин В.Л., Мартинов Г.М.* Системы числового программного управления: Учебное пособие. М.: Логос, 2005.
8. *Грейди Буч, Джеймс Рамбо, Айвар Джекобсон*. Язык UML. Руководство пользователя / Пер. с англ. М.: ДМК Пресс, 2001.
9. *http://www.componentone.com*/Products/NET/FlexGrid.aspx.
10. *Мартинов Г. М.* Академическая версия системы ЧПУ WinPCNC // Инструмент, технология, оборудование. 2007. № 8.

С.А. ВИНОКУРОВ, канд. техн. наук, доцент

Анализ и синтез математической модели электромеханической системы с бесконтактным двигателем постоянного тока при динамическом управлении

В статье на основе сравнительного анализа методов управления электромеханическими системами (ЭМС) с бесконтактным двигателем постоянного тока (БДПТ) осуществлен синтез математической модели системы в динамическом режиме. Предлагаются и исследуются управляющие функции по напряжению питания двигателя.

The article analyses and syntheses the mathematical models for electromechanical systems including brushless direct current motor under dynamic control in respect to comparative analysis of control modes. It suggests feed voltage control functions for the motor and cites the functions' investigation results.

Введение

В последнее время ЭМС с БДПТ находят все более широкое распространение в современных робототехнических устройствах и комплексах, задачах автоматизации различных технологических и производственных операций, медицинском оборудовании, аэрокосмическом приборостроении, в аппаратуре военного и специального назначения. Данное обстоятельство обусловлено очевидными преимуществами БДПТ, к числу которых, помимо отсутствия механического коммутатора, относят их хорошие регулировочные способности и широкий диапазон регулирования, относительно высокий КПД и оптимальные энергетические показатели [1…3]. ЭМС с БДПТ, обладая положительными свойствами двигателей постоянного тока коллекторного исполнения по регулировочным и механическим характеристикам, практически не нуждаются в эксплуатационном обслуживании и имеют высокие показатели надежности, что позволяет использовать их в сложных условиях внешней среды [4…6]. В этой связи разработка современных методов, моделей и алгоритмов управления ЭМС с БДПТ вызывает несомненный интерес у отечественных и зарубежных исследователей.

Сравнительный анализ методов управления ЭМС с БДПТ

Анализируя современные подходы к формированию управления ЭМС с различными типами исполнительных двигателей, можно обобщенно рассмотреть следующие основные этапы [7]:

- получение информации о цели и задачах управления;
- получение информации о состоянии объекта управления;
- анализ информации, выработка на ее основе решения об управлении;
- осуществление управляющих воздействий.

При разработке ЭМС с БДПТ необходимо с требуемой степенью качества реализовать процесс управления, учитывая также, что каждая фаза (стадия) процесса управления протекает при воздействии различных помех от внешней среды, часто полифакторных и недетерминированных. При выборе метода управления и последующем синтезе соответствующих устройств системы управления (СУ) обеспечивают достижение одной из следующих основных задач: стабилизации; выполнения программы; слежения; оптимизации.

Задачей стабилизации системы является поддержание ее выходных величин вблизи некоторых неизменных заданных и требуемых значений, несмотря на действие помех и внешних возмущающих воздействий. Параметры, стабилизируемые в процессе управления, выбираются в зависимости от условий и области применения конкретной системы, а также требований ее качества функционирования. Задача выполнения программы