УДК 518.682, 681.306

А. А. Горячкин, **В. Е. Зюбин**, **А. А. Лубков**

Новосибирский государственный университет
ул. Пирогова, 2, Новосибирск, 630090, Россия
Институт автоматики и электрометрии СО РАН
пр. Акад. Коптюга, 1, Новосибирск, 630090, Россия

E-mail: zyubin@iae.nsk.su

РАЗРАБОТКА ГРАФИЧЕСКОГО ФОРМАЛИЗМА ДЛЯ ОПИСАНИЯ АЛГОРИТМОВ В ПРОЦЕСС-ОРИЕНТИРОВАННОМ СТИЛЕ

Статья посвящена разработке графического процесс-ориентированного языка программирования HPD, предназначенного для спецификации управляющих алгоритмов в области промышленной автоматизации. Приводятся базовые идеи, лежащие в основе процесс-ориентированного программирования. Обсуждаются графические языки МЭК 61131-3 и графические спецификации, предназначенные для описания управляющих алгоритмов. Предлагается нотация языка HPD. Рассматривается пример его использования на тестовой задаче Шанмугьяма – Робертса.

Ключевые слова: процесс-ориентированное программирование, алгоритм управления, промышленная автоматизация, графический язык, hyper-process diagram.

Введение

В современных программно-аппаратных системах стоимость программного обеспечения сравнима, а в некоторых случаях даже превышает стоимость используемых аппаратных средств. При этом стоимость программной составляющей определяется не только стоимостью разработки и реализации, но и стоимостью сопровождения, которое предполагает как поиск и исправление ошибок, так и расширение функциональности [1]. Это приводит к созданию специализированных подходов, охватывающих автоматическую генерацию кодов программ по проектным спецификациям [2] и эволюционное расширение создаваемых программ [3].

Проблема снижения стоимости программного обеспечения актуальна и при создании систем управления технологическими процессами. Однако в силу специфики предметной области, в первую очередь связанной с *событийностью* алгоритма (необходимостью активно изменять реакцию системы в зависимости от событий на объекте управления), наличием *синхронизма* и *логического параллелизма* (необходимости обеспечения временной согласованности с физическими процессами, происходящими на объекте управления, и компактного описания их параллелизма), для создания ПО систем управления ТП используются специализированные языковые средства, которые слабо сочетаются с методиками управления проектами, разрабатываемыми в рамках компьютерного мэйнстрима.

Наиболее распространенные средства создания ПО в области промышленной автоматизации – пять языков МЭК-61131-3 [4], три из которых имеют графическую нотацию, что очень привлекательно с точки зрения простоты их изучения, и в общем обеспечивают удовлетворительное решение задачи создания ПО. В случае простых однородных алгоритмов управления (например, в задаче предполагается только дискретное управление при отсутствии контуров

Горячкин А. А., Зюбин В. Е., Лубков А. А. Разработка графического формализма для описания алгоритмов в процесс-ориентированном стиле // Вестн. Новосиб. гос. ун-та. Серия: Информационные технологии. 2013. Т. 11, вып. 2. С. 44–54.

аналогового регулирования) можно подобрать один из языков, не вызывающий проблем при использовании. Однако в случае автоматизации сложных объектов недостатки языков МЭК 61131-3 [5; 6] приводят к неоправданному росту кода с низким качеством, как в смысле наличия простых логических ошибок, так и в смысле трудоемкости его сопровождения. Это вынуждает разработчиков систем управления искать и применять альтернативные языковые средства. На практике встречаются примеры использования даже языков третьего поколения, таких как Си, Си++ или Паскаль [7].

В качестве языкового средства, обеспечивающего создание управляющих алгоритмов в области промышленной автоматизации, хорошо зарекомендовал себя язык процесс-ориентированного программирования Рефлекс [8]. Язык, известный также под именем «Си с процессами», предназначен для создания управляющих алгоритмов в области промышленной автоматизации и робототехнике: для систем, предполагающих активное взаимодействие с внешней средой, технологическим оборудованием, физическими процессами через датчики и органы управления. Базовые цели, которые ставились при разработке языка, – его адекватность специфике задач управления, простота изучения пользователем, комфортность программирования и сопровождения уже созданных программ. Язык по синтаксису напоминает язык Си, что делает его понятным для большинства практикующих программистов. Язык имеет англоязычный и русскоязычный синтаксис, а также допускает использование идентификаторов на русском языке. В отличие от языка Си, где программы строятся как иерархия функций, базовое понятие языка Рефлекс – процесс. Программа на языке Рефлекс – это множество параллельно исполняемых процессов, которые могут запускать друг друга, останавливать и контролировать текущее состояние. Для комфортного программирования систем промышленной автоматизации в языке предусмотрены операции с временными интервалами и средства описания связей с датчиками и управляющими органами.

На примере решения задачи автоматизации выращивания монокристаллического кремния методом вытягивания из расплава (метод Чохральского) [9] эмпирически доказано, что процесс-ориентированный подход позволяет создавать надежные и сопровождаемые алгоритмы сложных гетерогенных систем и программных комплексов в области промышленной автоматизации.

Показано, что процесс-ориентированный подход может обеспечить такие полезные возможности, как гибкая настройка времени реакции процессов на внешние события и статическая оценка ресурсоемкости алгоритма [10], а также использоваться при организации итерационной разработки управляющих алгоритмов в рамках концепции виртуальных объектов управления [11].

Показано, что на базе процесс-ориентированного подхода может быть проведено расширение языка ST из состава МЭК 61131-3 [12], которое радикально изменяет выразительные свойства ST и обеспечивает спецификацию на нем управляющих алгоритмов в широком диапазоне. Однако, несмотря на достигаемые высокие характеристики расширенной версии ST при кодировании, текстовая форма языка принципиально ограничивает его использование для спецификации алгоритма на начальных этапах разработки ПО, например, в силу того, что относительно высокий «порог вхождения» практически исключает его выбор в качестве языка коммуникации между разработчиком и заказчиком [2; 13]. Для коммуникативных целей крайне желателен графический процесс-ориентированный язык спецификации управляющего алгоритма.

Статья посвящена разработке графического языка программирования для описания управляющих алгоритмов в процесс-ориентированном стиле. Обсуждаются языки МЭК 61131-3 и альтернативные спецификации, предназначенные для графического описания управляющих алгоритмов. Предлагается нотация языка HPD. Рассматривается пример его использования на тестовой задаче автоматизации линии розлива бутылок.

Обзор существующих графических языков спецификации управляющих алгоритмов

Кроме требования отражать уже упоминавшуюся специфику (событийность, синхронизм и параллелизм), присущую предметной области, графические языки описания управляющих алгоритмов должны удовлетворять следующим требованиям.

1. Предоставлять возможность иерархической структуризации алгоритма в виде, адекватном технологическому описанию автоматизируемого процесса, специфицированному в техни-

ческом задании на разработку, в частности возможность гибкого управления параллельными процессами, например их выборочное прекращение в случае отказа оборудования.

2. Позволять генерацию исполняемого кода на основе графического описания.
3. Иметь низкий порог вхождения.
4. Обеспечивать кроссплатформенность и кроссбрендовость спецификации.
5. В случае наличия недостатков допускать их устранение, например за счет расширения свойств.

В качестве возможной базы для создаваемой графической нотации были рассмотрены три графических языка МЭК 61131-3 и известные альтернативные графические спецификации (табл. 1).

Таблица 1

Сравнительная таблица графических языков спецификации управляющих алгоритмов

Название	Тип*	Синхронизм	Параллелизм	Иерархия	Генерация кода	Простота изучения	Переносимость	Расширяемость
LD	DF	±	–	–	+	+	±	–
FBD	DF	+	±	±	+	+	±	–
SFC	CF	±	±	–	+	±	±	±
МЭК 61499	CF, ED	±	±	±	+	±	N/A	±
G	DF	–	±	±	±	±	±	–
HSM	CF, ED	–	–	±	+	±	+	±
Flowchart	CF	–	–	±	+	+	+	+

* Вычислительная модель программы: CF (control flow) потока управления; DF (data flow) – потока данных; ED (event-driven) – управления событиями.

Ladder Diagram (LD). Графический язык из состава МЭК 61131-3, стандартизованный вариант класса языков релейно-контактных схем. Тип – модель потока данных. Логические выражения на этом языке описываются с помощью метафоры реле, которые широко применялись в области автоматизации в 1960-х гг. Из-за своих ограниченных возможностей язык дополнен привнесенными средствами: таймерами, счетчиками и т. п. Благодаря простому представлению, LD легко транслируется в исполняемый код. Это же обуславливает простую и наглядную трассировку программ, как во время отладки, так и во время исполнения. Язык имеет низкий порог вхождения. Ориентирован на задачи логического управления. Средства распараллеливания участков кода и механизмы их последующей дивергенции отсутствуют. Параллелизм обеспечен только на уровне базовых инструкций. Структурирующими свойствами не обладает. Для организации синхронизма используются системные функции. Событийность инородна языку и может быть реализована только через большой объем рутинного кодирования. Кроссбрендовая переносимость спецификаций на LD не обеспечена. Устранение недостатков невозможно без изменения базовых свойств.

Function Block Diagram (FBD). Графический язык из состава МЭК 61131-3, реализующий вычислительную модель потока данных. Модель предполагает описания алгоритма на основе метафоры микросхем – в виде потоков данных, присоединяемых к функциональным блокам, производящим арифметические и логические преобразования. Язык имеет низкий порог вхождения. Транслируется в исполняемый код. Имеется возможность иерархического описания функциональных блоков. Концепция обеспечивает описание логического параллелизма. Для организации синхронизма используются системные функции. Возможность управления по событиям инородна базовой метафоре языка, предоставляемые механизмы ее реализации крайне ненадежны. Этот недостаток предопределяет специализацию FBD – задачи параметрического

регулирования. Кроссбрендовая переносимость спецификаций на FBD не обеспечена. Устранение недостатков невозможно без изменения базовых свойств.

Sequential Function Chart (SFC). Язык из состава МЭК 61131-3, предполагающий описание алгоритма в виде набора связанных пар: шаг (*step*) и переход (*transition*). Шаг представляет собой набор операций над переменными. Переход – набор логических условных выражений, определяющий передачу управления к следующей паре шаг-переход. По внешнему виду описание на языке SFC напоминает хорошо известные логические блок-схемы алгоритмов, хотя идеологически SFC близок к сетям Петри. Благодаря возможности указать для условного перехода более одного управляющего потока SFC позволяет реализовать параллелизм. В SFC присутствует возможность описания управления по событиям. К недостаткам языка традиционно относят его слабые структурирующие свойства [14] и отсутствие надежных механизмов конвергенции параллельных участков кода [6]. Для организации синхронизма используются системные функции. Для описания алгоритмического содержимого шагов и условий перехода применяются дополнительные языковые средства. Например, текстовые языки программирования Instruction List (IL) или Structured Text (ST) из стандарта МЭК-61131-3. В связке с языком ST обеспечивает возможность генерации исполняемого кода и расценивается как наиболее мощное средство описания алгоритмов управления. Кроссплатформенная переносимость затруднена. Кроссбрендовая переносимость спецификаций не обеспечена. Устранение недостатков невозможно без изменения базовых свойств.

Средство МЭК 61499. Графический язык МЭК-61499 [15] – одна из попыток преодолеть ограничения языков МЭК61131-3 и скомбинировать в одном языковом средстве и поддержку логического параллелизма, и поддержку событийности. Цель стандарта – предоставить методологию разработки сложных алгоритмов. Программные компоненты представлены функциональными блоками специального вида: кроме обычных для языка FBD входных и выходных данных, интерфейс функционального блока стандарта МЭК-61499 предполагает событийные входы / выходы. Несомненно, это нововведение частично решает проблему событийности для классических функциональных блоков. Концепция требует заметных затрат на изучение. Структуризация алгоритма в виде, адекватном заданному в техническом задании, затруднена. Подход обеспечивает генерацию исполняемого кода. Данные о кроссбрендовой переносимости спецификаций отсутствуют. Устранение недостатков затруднено.

Графический язык G. Графический язык программирования G, используемый в пакете программ LabVIEW, аналогично SFC основан на модели потоков данных. Последовательность выполнения операторов в языке определяется не порядком их следования (как в императивных языках программирования), а наличием данных на входах этих операторов. Операторы, не связанные по данным, выполняются параллельно в произвольном порядке. Позволяет существенно сократить трудоемкость создания пользовательского интерфейса, поэтому может быть использован в качестве альтернативы SCADA-пакету. Как и SFC, характеризуется проблемной реализацией событийности при описании взаимодействия с объектом управления. Для организации синхронизма используются системные функции. Известные подходы, обеспечивающие реализацию процесс-ориентированного стиля средствами языка G, связаны с существенными ограничениями [16]. Генерация исполняемого кода предполагает развертывание на целевой платформе дополнительного ПО – RunTime Engine. Кроссплатформенная переносимость обеспечивается единственным производителем (National Instruments). Кроссбрендовая переносимость отсутствует за неимением альтернативных производителей. Расширяемость практически невозможна.

Hierarchical State Machine (HSM). Язык HSM, базирующийся на расширенной модели конечного автомата, позволяет создавать вложенные состояния, описывая тем самым сложные иерархические структуры. Событие – принципиальное понятие в диаграмме состояний UML. Конечный автомат выполняет действия в ответ на произошедшее событие. Данное средство разработки позволяет описывать событийно-ориентированные системы, преобладающие в сфере АСУ. При этом генерация кода на основе UML диаграммы требует дополнительного описания событий и реакций на них, так называемый «action list» в HSM содержит лишь заголовки вызываемых методов. В силу существующей связи «событие – реакция» язык HSM не позволяет создавать параллельные алгоритмы. Порог вхождения – средний. Может быть обес-

печена кроссбрендовая и кроссплатформенная переносимость. Устранение недостатков затруднено.

Flowchart (логические блок-схемы). Язык отражает ход потока управления во время исполнения программы. Позволяет описывать алгоритм управления в виде последовательных шагов. Важным элементом блок-схемы является *условие*, с помощью которого происходит ветвление алгоритма. Язык имеет низкий порог вхождения. Событийность в блок-схемах может быть реализована соответствующими условиями, где реакция на событие – это переход по условию в нужное место алгоритма, однако описание сопряжено с выполнением рутинных операций по кодированию, аналогично случаю использования процедурных языков [6]. Средства организации синхронизма не предусмотрены. Язык допускает описание параллельных участков кода и модульную структуризацию, при этом механизмы гибкого управления параллельными участками кода не предусмотрены. Язык допускает генерацию исполняемого кода. Может быть обеспечена кроссбрендовая и кроссплатформенная переносимость. Устранение недостатков возможно.

Таким образом, можно констатировать, что, несмотря на серьезные недостатки в части возможности описания синхронизма и параллелизма, ориентации на вычислительные алгоритмы, при разработке графической нотации, ориентированной на описание управляющих алгоритмов широкого класса, может быть использован формализм Flowchart.

Графическая нотация Hyper-Process Diagram

В результате проведенного анализа существующих графических языков в качестве базы для разрабатываемой графической спецификации были выбраны средства Flowchart (проверка условия, процесс, стрелки потока управления). Предложенная графическая нотация языка, названного Hyper-Process Diagram (HPD), приведена в табл. 2. Графические средства Flowchart были расширены символами «процесс и его функции-состояния», «смена текущей функции-состояния», «нормальный останов текущего процесса», «останов текущего процесса по ошибке», «запуск группы процессов», «останов группы процессов по ошибке», «нормальный останов группы процессов», «терминатор состояния».

Таблица 2

Графическая нотация языка HPD

Наименование	Обозначение	Функция
Безусловное выражение	Файл expression.ai	Выполнение одной или нескольких операций, обработка данных любого вида (изменение значения данных, формы представления, расположения)
Условие	Файл condition.ai	Базовый элемент блок-схемы
Поток «да»	Файл flowyes.ai	Продолжение алгоритма при выполнении условия
Поток «нет»	Файл flowno.ai	Продолжение алгоритма в случае, если условие не выполнено
Дополнения	Файл addition.ai	Дополнения к описанию элементов диаграммы
СТАРТ процесса	Файл start.ai	Запуск процесса или группы процессов
СТОП процесса	Файл stop.ai	Нормальный останов процесса или группы процессов
ОШИБКА процесса	Файл error.ai	Останов процесса или группы процессов с ошибкой
Смена состояния	Файл changestate.ai	Смена текущей функции-состояния у процесса
Следующее	Файл next.ai	Смена текущей функции состояния на следующую
СТОП	Файл stop_current.ai	Нормальный останов текущего процесса
ОШИБКА	Файл error_current.ai	Останов текущего процесса с ошибкой
Таймаут	Файл timer.ai	Описание таймаута для функции состояния

Конец состояния	Файл end.ai	Выход из функции состояния

Пример использования HPD

В качестве тестовой задачи для демонстрации свойств HPD была выбрана задача Шанмугьяма – Робертса [17] по автоматизации линии розлива жидкости в бутылки (рис. 1).

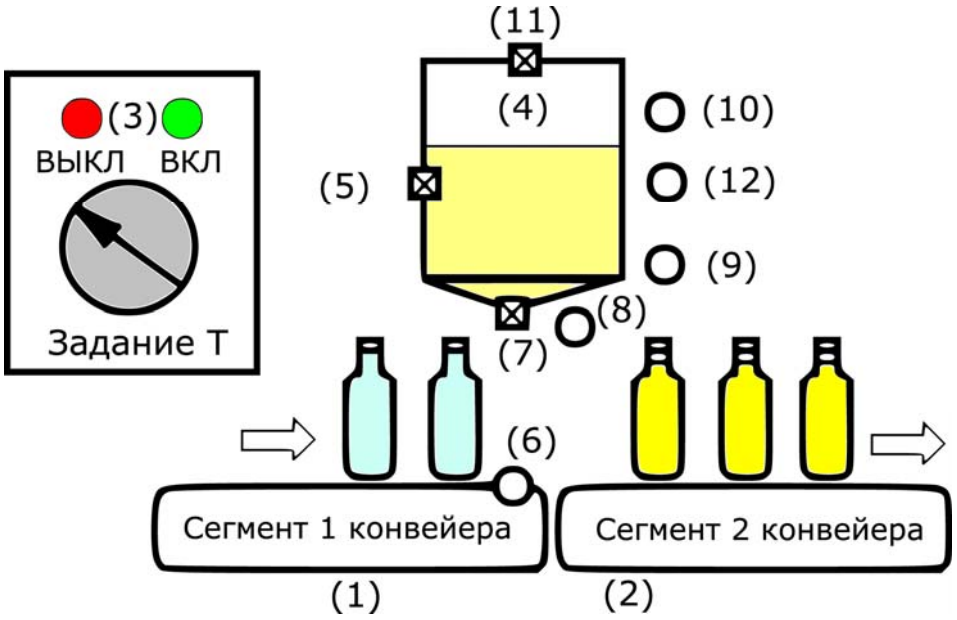

Рис. 1. Автоматическая линия розлива бутылок

Условие задачи формулируется следующим образом. Двухсегментный конвейер (*1*, *2*) используется для перемещения бутылок. Первый сегмент (*1*) включается и отключается в соответствии с алгоритмом, приведенным чуть позже. Второй сегмент (*2*) по включению системы постоянно находится в работе. Кнопка управления системой (*3*) включает и отключает систему. Резервуар (*4*) содержит разливаемую жидкость. По условию задачи разливаемая жидкость находится в нагретом состоянии, температура жидкости поддерживается на уровне 100 °C. Хотя температура может поддерживаться типовым ПИД-регулятором, по условию задачи она регулируется паровым клапаном (*5*). При падении температуры жидкости ниже 100 °C этот клапан открывается и пускает в систему перегретый пар, который разогревает резервуар. Клапан закрывается по достижении температуры 110 °C. Датчик положения бутылки (*6*) служит для обнаружения бутылки в заданном положении и срабатывает, когда бутылка находится под соплом резервуара. Когда бутылка обнаружена, первый сегмент конвейера останавливается. Клапан разлива в бутылку (*7*), расположенный на дне резервуара, открывается при совпадении следующих четырех условий: а) обнаружена бутылка; б) эта бутылка пуста; в) в резервуаре есть жидкость; г) температура жидкости больше или равна 100 °C. После его открытия жидкость через сопло заливается в бутылку. Количество жидкости в бутылке контролируется специальным фотодатчиком уровня жидкости (*8*) в бутылке, который срабатывает, когда бутылка заполнена. Уровень жидкости в резервуаре контролируется двумя датчиками: датчиком отсутствия жидкости (*9*) и датчиком полноты резервуара (*10*). Клапан пополнения резервуара жидкостью (*11*) открывается по срабатыванию датчика отсутствия жидкости и выключается при срабатывании датчика полноты резервуара. Операции разлива в бутылки (клапан разлива в бутылку) и разогрева (паровой клапан) запрещены во время пополнения резервуара жидкостью (клапан пополнения резервуара). Температура жидкости измеряется аналоговым датчиком (*12*).

Решение задачи средствами графического языка HPD, реализованного в виде набора инструментов в среде Microsoft Visio 2010, представлено на рис. 2–4.

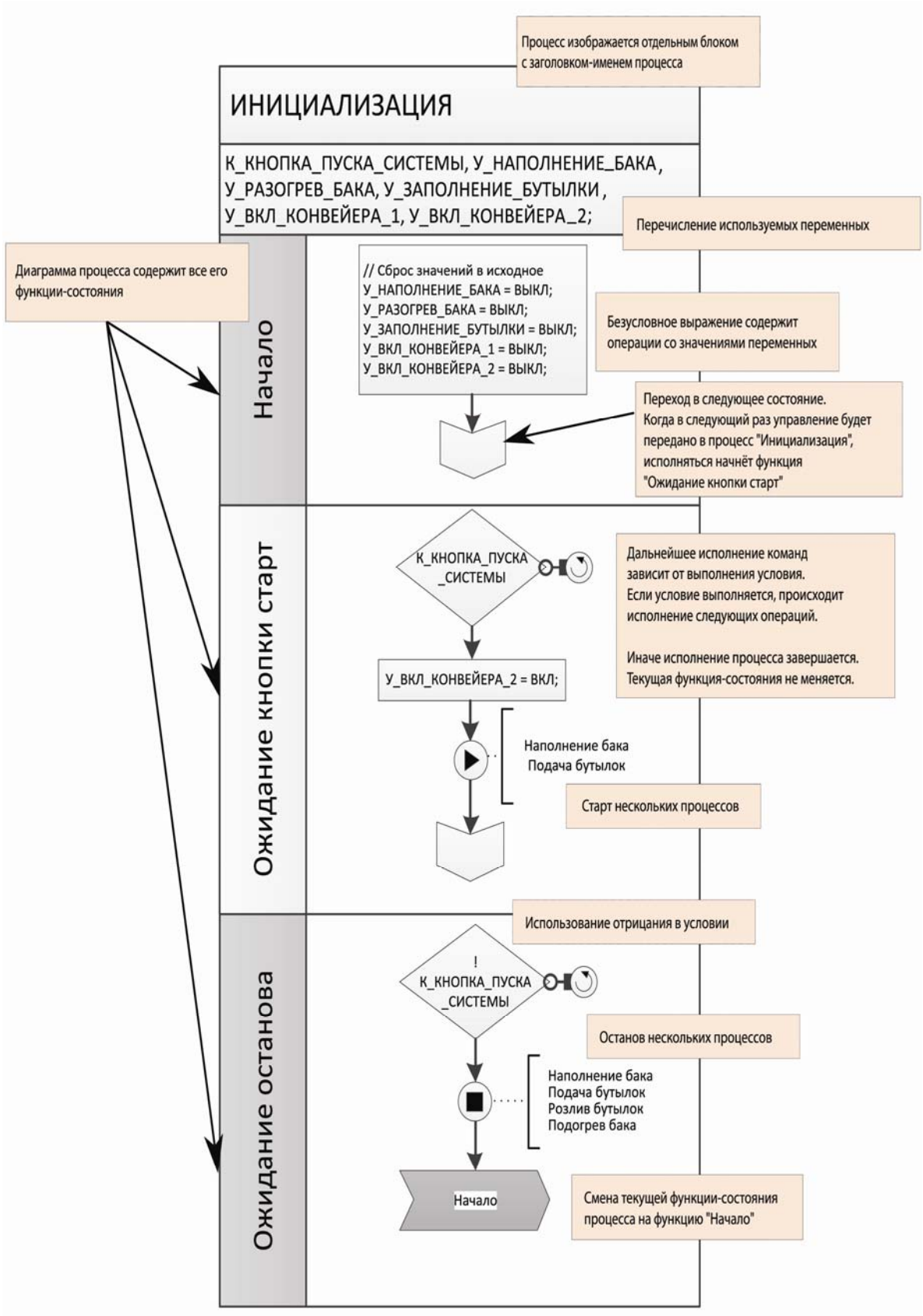

Рис. 2. Инициализация

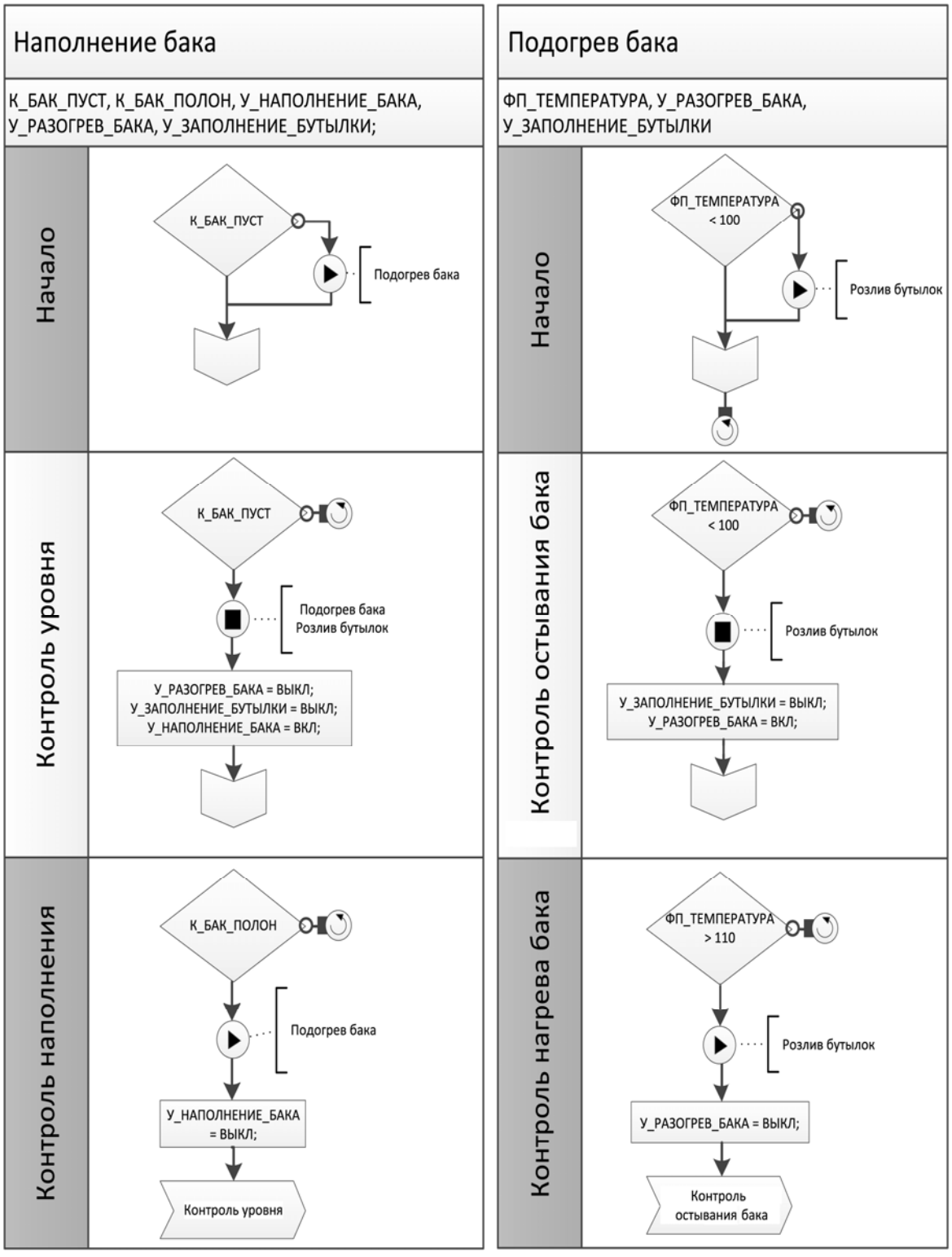

Рис. 3. Наполнение и подогрев бака

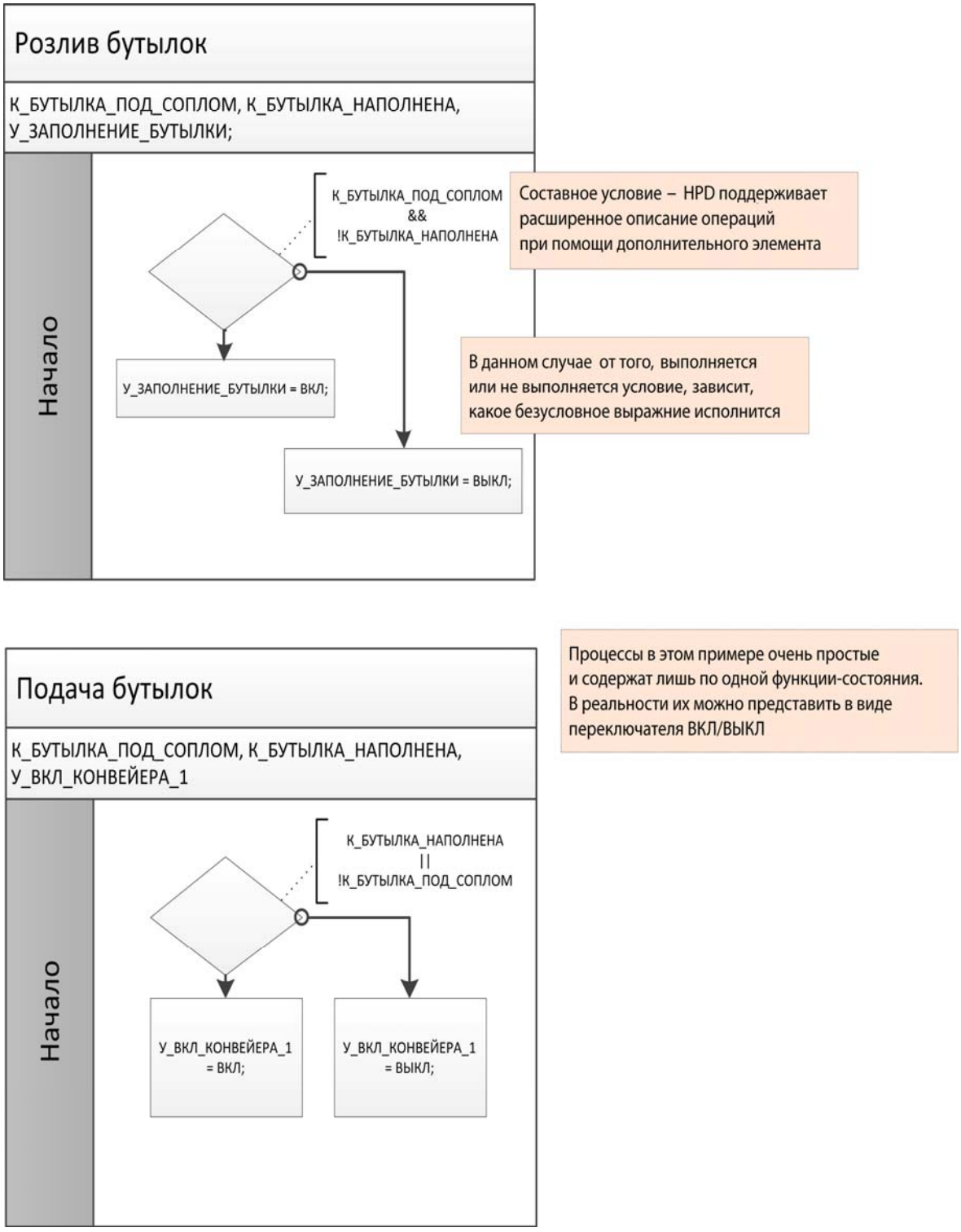

Рис. 4. Розлив и подача бутылок

В силу операциональной эквивалентности HPD и языка Рефлекс автоматически обеспечиваются функциональные требования, предъявляемые к формализму в части возможности управления по событиям, обеспечения синхронизма, параллелизма, формы структуризации алгоритма и механизмов гибкого управления параллельными участками кода. Это свойство HPD позволяет обеспечить и переносимость создаваемых спецификаций через существующие трансляторы языка Рефлекс.

Заключение

В работе предложен графический формализм HPD, расширяющий Flawchart выразительными средствами поддержки процесс-ориентированного стиля программирования. Формализм HPD позволяет создавать графические спецификации сложных гетерогенных управляющих алгоритмов и ориентирован на использование при проектировании ПО систем управления в качестве коммуникативного средства (облегчая согласование алгоритма с заказчиком и выявление внутренних несогласованностей в техническом задании, сокращая время выработки базовой стратегии управления и архитектуры разрабатываемых программ). Формализм может быть выбран в качестве удобного формата документирования запрограммированных алгоритмов, а также служить основой для создания интегрированной среды разработки алгоритмов управления с возможностями автоматической генерации переносимого исполняемого кода. Как вариант, формализм HPD может быть рекомендован в качестве средства, расширяющего возможности набора диаграмм UML.

Список литературы

1. *Galorath D.* Software Total Ownership Costs: Development Is Only Job One. Galorath Inc., 2012. URL: http://ebookbrowse.com/software-total-ownership-costs-job-one-pdf-d380720510.

2. *Котляров В. П., Дробинцев П. Д.* Автоматизация проектирования программного продукта с помощью формальных спецификаций // Вестн. Новосиб. гос. ун-та. Серия: Информационные технологии. 2011. Т. 9, вып. 4. С. 29–38.

3. *Легалов А. И., Солоха А. Ф.* Особенности языковой поддержки процедурно-параметрической парадигмы программирования // Вестн. Новосиб. гос. ун-та. Серия: Информационные технологии. 2011. Т. 9, вып. 3. С. 15–22.

4. IEC 61131-3. Programmable controllers. Part 3: Programming languages. 2nd ed. Int. Electrotechnic Commission, 1998.

5. *Crater K.* When Technology Standards Become Counterproductive. Control Technology Corporation, 1996.

6. *Зюбин В. Е.* Программирование ПЛК: языки МЭК 61131-3 и возможные альтернативы // Промышленные АСУ и контроллеры. 2005. № 11. С. 31–35.

7. *Вотинов М. В., Маслов А. А.* Автоматизация технологического процесса термической обработки сырья в пищевой промышленности на примере малогабаритной сушильной установки // Вестн. Новосиб. гос. ун-та. Серия: Информационные технологии. 2012. Т. 10, вып. 3. С. 15–25.

8. *Зюбин В. Е.* «Си с процессами»: язык программирования логических контроллеров // Мехатроника, автоматизация, управление. 2006. № 12. С. 31–35.

9. *Зюбин В. Е., Котов В. Н., Котов Н. В., Курочкин А. В., Лубков А. А., Лылов С. А., Окунишников С. В., Петухов А. Д.* Базовый модуль, управляющий установкой для выращивания монокристаллов кремния // Датчики и системы. 2004. № 12. С. 17–22.

10. *Зюбин В. Е.* Статическая балансировка вычислительной нагрузки в процесс-ориентированном программировании при многопоточной реализации // Вестн. Новосиб. гос. ун-та. Серия: Информационные технологии. 2012. Т. 10, вып. 2. С. 44–54.

11. *Зюбин В. Е.* Итерационная разработка управляющих алгоритмов на основе имитационного моделирования объекта управления // Автоматизация в промышленности. 2010. № 11. С. 43–48.

12. *Зюбин В. Е.* Пути расширения языка ST из состава МЭК 61131-3 для задач промышленной автоматизации // Приборы и системы. Управление, контроль, диагностика. 2009. № 3. С. 16–19.

13. *Зюбин В. Е.* Графика и текст: какой язык выбрать программисту // Открытые системы. СУБД. 2004. № 1. С. 54–58.

14. *Анисимов Н. А., Голенков Е. А., Харитонов Д. И.* Композициональный подход к разработке параллельных и распределенных систем на основе сетей Петри // Программирование. 2001. № 6. С. 30–43.

15. *Christensen J.* IEC 61499. Function Blocks for industrial-process measurement and control systems. FAQ. 1999.

16. *Зюбин В. Е.* LabVIEW: создание управляющих алгоритмов в процесс-ориентированном стиле // Промышленные АСУ и контроллеры. 2011. № 1. С. 39–45.

17. *Shanmugham S. G., Roberts C. A.* Application of Graphical Specification Methodologies to Manufacturing Control Logic Development: A Classification and Comparison // Int. J. Computer Integrated Manufacturing. 1998. Vol. 11, № 2. P. 142–152.

Материал поступил в редколлегию 24.02.2012

A. A. Goryachkin, V. E. Zyubin, A. A. Lubkov

PROCESS-ORIENTED GRAPHICAL SPECIFICATION FOR CONTROL ALGORITHMS

This paper presents a graphic process-oriented language called hyper-process diagram (HPD) for specification of control algorithms in industrial automation. The paper describes conceptual means of process-oriented programming, and evaluates IEC 61131-3 and alternative graphic formalisms. HPD notation is presented and used to develop control algorithms for the Shanmugham-Roberts discrete-event control system.

Keywords: process-oriented programming, control algorithm, industrial automation, graphic language.