

ПРОГРАММНАЯ МОДЕЛЬ RnP-МАНИПУЛЯТОРА НА ОСНОВЕ СЕРВИС-ОРИЕНТИРОВАННОЙ АРХИТЕКТУРЫ

Предлагается программная сервис-ориентированная модель RnP-манипулятора, включающая как модель управления, так и модель оборудования, взаимодействующие друг с другом. Данная модель может быть использована для имитационного моделирования и прототипирования реальных систем, а также в учебном процессе.

Сервисно-ориентированная архитектура (*service-oriented architecture, SOA*) — это подход к разработке программного обеспечения, в основе которого лежат сервисы со стандартизированными интерфейсами [1]. В самом общем виде SOA предполагает наличие трех основных участников: поставщика сервиса, потребителя сервиса и реестра сервисов (рисунок 1). Взаимодействие в системе выглядит достаточно просто: поставщик сервиса регистрирует свои сервисы в реестре, а потребитель обращается к реестру с запросом. Основные принципы SOA: 1) слабое связывание; 2) повторное использование; 3) расширяемость; 4) распределенные вычисления. Это делает привлекательным использование данного подхода для создания программных имитационных моделей промышленных киберфизических систем (КФС), а также их прототипирования и реализации. Например, в работе [2] представлена сервис-ориентированная реализация систем распределенной автоматики, где оркестровка сервисов основывается на использовании функциональных блоков стандарта IEC 61499. В работе [3] была предложена программная сервис-ориентированная модель системы транспортировки багажа в аэропорту.

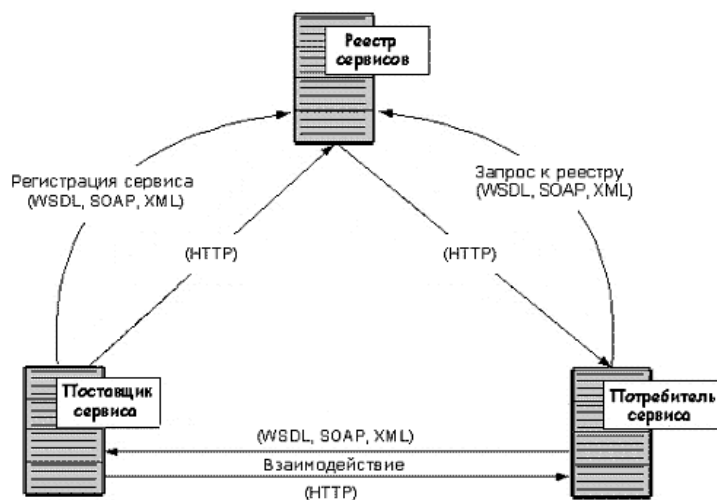


Рисунок 1 – Базовая архитектура SOA

В данной работе предлагается абстрактная программная сервис-ориентированная модель RnP-манипулятора. В отличие от работы [2],

нацеленной на сервис-ориентированную реализацию систем управления, в данной работе рассматривается замкнутая имитационная модель «управление-оборудование», которая включает, как сервисы, представляющие управление (контроллеры), так и сервисы, моделирующие оборудование, взаимодействующие друг с другом. Для начального формализованного описания контроллеров использовались конечные автоматы. Модель оборудования представлена в виде временных задержек. Данный подход является довольно абстрактным, тем не менее, он позволяет строить имитационные модели КФС любого уровня. При разработке композитных сервисов использовался язык BPEL (*Business Process Execution Language*), предназначенный для формального описания бизнес-процессов и протоколов их взаимодействия между собой [4]. Язык BPEL расширяет модель взаимодействия веб-сервисов и включает в эту модель поддержку транзакций.

PnP-манипулятор (*Pick-and-Place manipulator*) предназначен для отбора деталей с рабочей плоскости и их переноса в буферную зону [2]. Данный манипулятор состоит из двух цепочек цилиндров: цепочки горизонтальных цилиндров и цепочки вертикальных цилиндров. Цилиндр может «сжиматься» и «растягиваться» (за счет поршня). Всю цепочку цилиндров можно представить в виде телескопической антенны. На конце крайнего вертикального цилиндра расположена присоска, позволяющая захватить деталь, лежащую на плоскости. Присоска может притягивать деталь или отпускать её. Пример манипулятора с двумя горизонтальными и одним вертикальным цилиндром показан на рисунке 2. Данный манипулятор может подбирать одну из трёх деталей (pp1, pp2, pp3) и доставлять её в буфер для хранения деталей (pp0).

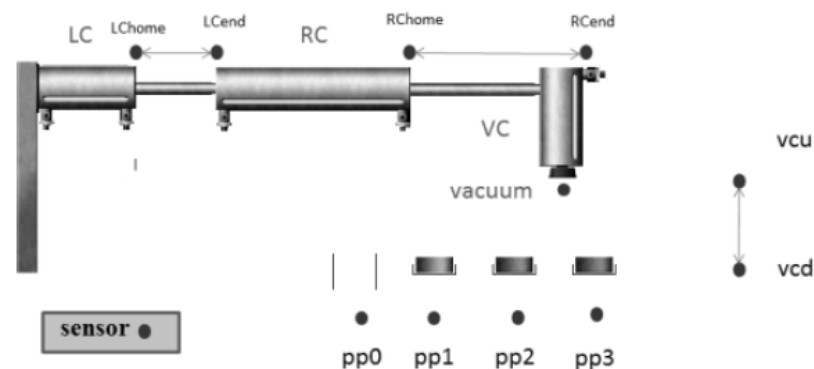


Рисунок 2 – Структура манипулятора "Pick and Place"

Работа PnP-манипулятора может быть описана набором временных диаграмм, одна из которых приведена на рисунке 3. На рисунках 2 и 3 LC и RC – левый и правый горизонтальные цилиндры, соответственно, VC – вертикальный цилиндр, VAC – вакуумная присоска, Ext и Ret – расширение и втягивание горизонтального цилиндра, соответственно, Down и Up – расширение и втягивание вертикального цилиндра, соответственно; Pick up – захват детали; Drop – отпускание детали.

Программная SOA-модель PnP-манипулятора включает следующие сервисы: сервис цилиндра (*Device*), сервис вакуумной присоски (*Vacuum*), сервис контроллера присоски (*VacuumController*), сервис контроллера горизонтального цилиндра (*CylinderController*), сервис контроллера вертикального цилиндра (*VerticalCylinderController*), сервис главного контроллера (*MasterController*), причем сервисы *Device*, *Vacuum* являются базовыми, а остальные сервисы - композитные.

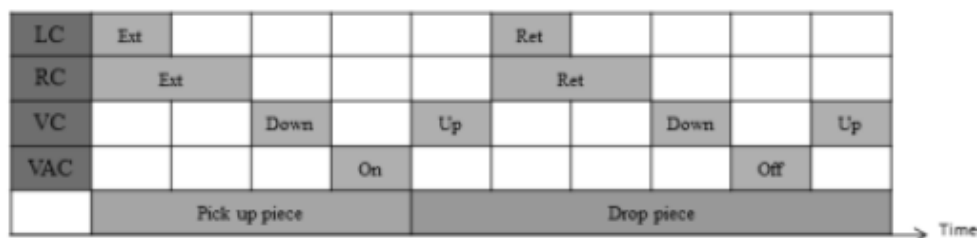


Рисунок 3 – Временная диаграмма работы манипулятора при взятии детали с места pp3

Работа цилиндра в сервисе *Device* реализована в виде временной задержки, равной *N* секундам. Входные данные сервиса: *cylinderName* (тип *string*) – имя цилиндра; *typeOfOperation* (тип *string*) – выполняемая операция (*forward* – выдвинуть, *backward* – задвинуть). Выходные данные: *result* (тип *string*) – результат работы, характеризующий состояние цилиндра.

Работа вакуумной присоски в сервисе *Vacuum* реализована в виде временной задержки, равной *K* секундам. Входные данные: *typeOfOperation* (тип *string*) – выполняемая операция (*fix* – притянуть деталь, *unfix* – отпустить деталь). Выходные данные: *result* (тип *string*) – результат работы.

Контроллер горизонтального цилиндра в сервисе *CylinderController* представлен в виде конечного автомата с состояниями: *init*, *cylAtHome*, *cylAtEnd*, *GoForward*, *GoBackward*. Состояния *GoForward* и *GoBackward* являются промежуточными. Данный сервис в процессе своего выполнения вызывает сервис *Device*, который представляет модель цилиндра. Входные данные сервиса: *currentState* (тип *string*) – текущее состояние контроллера; *command* (тип *string*) – выполняемая операция (*forward* – «выдвинуть», *backward* – «задвинуть»); *deviceName* (тип *string*) – имя цилиндра. Выходные данные сервиса: *newState* (тип *string*) – новое состояние, в которое перешел контроллер после выполнения очередной операции. Сервис получает на вход три переменные и в зависимости от текущего состояния процесс идёт по одной из трёх ветвей, каждая из которых соответствует своему состоянию. Далее, в зависимости от типа операции процесс подготавливает вызов сервиса цилиндра и передает ему на вход тип выполняемой операции и имя цилиндра. В лог будет выведено текущее, промежуточное и конечное состояния, а также показана работа устройства цилиндра в форме выдаваемых на печать сообщений. BPEL-диаграмма данного сервиса представлена на рисунке 4. В условных вершинах C1, C2, C3 с использованием переменной *currentState* анализируется состояние конечного автомата и в зависимости от ее значения будет выбрана одна из трех ветвей. В

условной вершине C4 с использованием переменной *command* определяется тип операции.

Как и в случае контроллера горизонтального цилиндра, контроллер вертикального цилиндра (сервис *VerticalCylinderController*) представлен в виде конечного автомата.

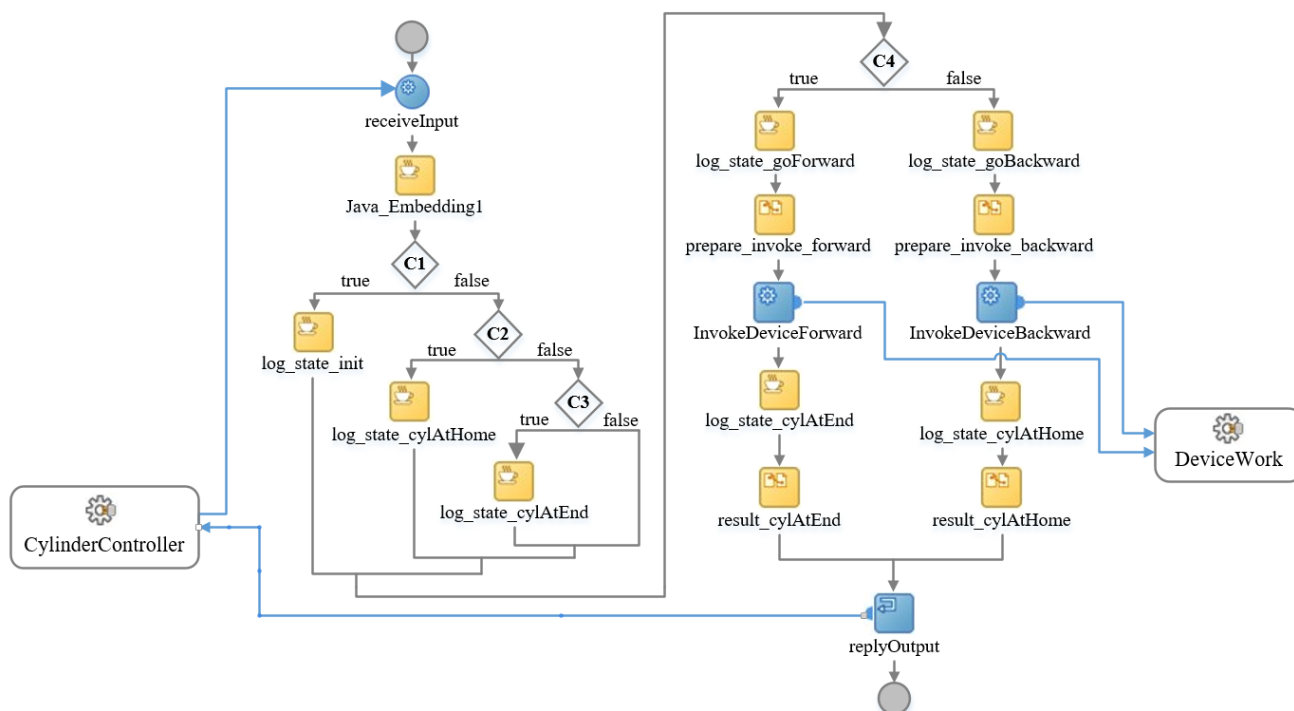


Рисунок 4 – BPEL-диаграмма для сервиса горизонтального цилиндра

Данный сервис в процессе своего выполнения вызывает сервис *Device* и сервис *VacuumController*. В отличие от контроллера горизонтального цилиндра данный контроллер является не только ведомым (*Slave*), но и ведущим (*Master*), так как он управляет работой контроллера вакуумной присоски. Входные данные: *currentState* (тип *string*) – текущее состояние контроллера; *currentVacuumState* (тип *string*) – текущее состояние контроллера вакуумной присоски; *command* (тип *string*) – выполняемая операция (*forward* – выдвинуть, *backward* – задвинуть); *deviceName* (тип *string*) – имя цилиндра. Выходные данные: *newState* (тип *string*), *newVacuumState* (тип *string*) – новые состояния, в которые перешли контролер цилиндра и контроллер присоски после выполнения очередной операции. Сервис получает на вход четыре переменные, и в зависимости от текущего состояния процесс идёт по одной из трёх ветвей. Далее, в зависимости от типа операции процесс подготавливает вызов сервиса цилиндра и передает ему на вход тип выполняемой операции и имя цилиндра. После этого, если была выполнена операция *forward*, то будет вызван сервис контроллера присоски, которому в зависимости от его текущего состояния будет передан тип операции. Если текущее состояние присоски *vacuumOn* то операция будет *unfix*, если *vacuumOff*, то операция *fix*. В лог будет выведено текущее состояние,

промежуточное и конечное контроллера цилиндра, текущее, промежуточное и конечное состояния контроллера присоски.

Работа главного контроллера в рамках сервиса *MasterController* была реализована в виде процесса, который управляет контроллерами горизонтального и вертикального цилиндров. Данный сервис в процессе своего выполнения вызывает сервисы *CylinderController* и *VerticalCylinderController*. Входные данные сервиса: *numberOfItem* (тип *int*) – номер детали, которую необходимо взять (от 1 до 3). Выходные данные: *endWork* (тип *string*) – сообщение об окончании выполнения работы. Данный сервис хранит состояния всех контроллеров, поэтому в него введены дополнительные внутренние переменные, значение которых он передает сервисам. Сервис получает на вход одну переменную, анализирует, какую именно деталь необходимо взять, и вызывает необходимые ведомые сервисы, передавая им на вход необходимые данные, в том числе их текущее состояние. Сначала сервис организует взятие детали, для этого он выдвигает необходимые горизонтальные цилиндры (деталь 1 – только первый цилиндр, деталь 2 – только второй, деталь 3 – первый и второй цилиндры), вертикальный цилиндр, который управляя присоской, притягивает деталь. Далее сервис организует доставку взятой детали к месту сброса, для чего задвигается вертикальный цилиндр и выдвинутые горизонтальные цилиндры, вертикальный цилиндр с деталью выдвигается, контроллер вертикального цилиндра отключает присоску и деталь сбрасывается, вертикальный цилиндр задвигается. После окончания работы сервис отправляет ответное сообщение с текстом, свидетельствующим об окончании работы.

Сервис-ориентированная модель PnP-манипулятора разработана и реализована на основе программного средства Oracle SOA Suite [5]. Направлением дальнейшей работы является реализация предложенной модели с использованием технологии облачных сервисов.

Литература

1. Erl T. Service-Oriented Architecture: Concepts, Technology and Design: 1st Edition. – Prentice Hall, 2016. – 792 P.
2. Demin E., Dubinin V., Patil S., Vyatkin V. Automation Services Orchestration with Function Blocks: Web-service Implementation and Performance Evaluation // Studies in Computational Intelligence, Vol. 640. – Springer, 2016. – P. 213-221.
3. Ручкин, М.А. Дубинин А.В., Дубинин В.Н. Сервис-ориентированная модель системы транспортировки багажа аэропорта // Сборник научных статей Международной научно-технической конференции "Современные информационные технологии", Пенза, 2020, вып. 31. – С. 6-13.
4. BPEL Designer and Service Engine User's Guide. – Sun Microsystems, Inc., 2009. – 180 P.
5. Reynolds A., Wright M. Oracle SOA Suite 11g R1 Developer's Guide – Packt Publishing, 2010. – 722 P.