УДК 004.75 : 519.876.5 DOI 10.21685/2227-8486-2020-4-10

ПРОГРАММНЫЕ МОДЕЛИ МЕХАТРОННЫХ СИСТЕМ НА ОСНОВЕ СЕРВИС-ОРИЕНТИРОВАННОЙ АРХИТЕКТУРЫ

В. Н. Дубинин, А. В. Дубинин, М. А. Ручкин

SOFTWARE MODELS OF MECHATRONIC SYSTEMS BASED ON SERVICE-ORIENTED ARCHITECTURE

V. N. Dubinin, A. V. Dubinin, M. A. Ruchkin

Аннотация. *Предмет и цель работы*. Принятие концепции «Индустрия 4.0» в качестве одного из основных направлений развития промышленного производства предполагает использование новых подходов к моделированию, проектированию и реализации мехатронных систем, переходящих в своем развитии в разряд киберфизичеких систем. Это определяет актуальность использования новых информационных технологий, в числе которых сервис-ориентированные архитектуры и технологии семантического веб, для решения данной проблемы. *Методы*. Предложена инженерная методика построения реконфигурируемых программных моделей мехатронных систем, рассматриваемых как дискретные событийные системы. Программная модель строится по иерархическому модульному принципу на основе сервис-ориентированной архитектуры. Для построения модели отдельного устройства как объекта автоматизации, входящего в систему, используется паттерн проектирования Model/View/Controller. Каждая из компонент этого паттерна может быть отображена на отдельный сервис. *Результаты и выводы*. На основе предложенной методики разработаны программные модели PnP-манипулятора и системы транспортировки багажа в аэропорту, пригодные для их прототипирования. Для построения данных моделей использовались программные средства Oracle SOA Suite, OpenESB и Netbeans IDE. Благодаря гибкости, реконфигурируемости и интероперабельности возможен быстрый переход от программных моделей к реальным системам.

Ключевые слова: модель, сервис-ориентированная архитектура, веб-сервис, PnP-манипулятор, система транспортировки багажа, BPEL.

Abstract. *Subject and goals*. The adoption of the concept Industry 4.0 as one of the main directions of development of manufacturing intends to use new approaches to the modeling, design and implementation of mechatronic systems, which are growing to cyber-physical systems. This determines the relevance of using new information technologies including the service-oriented architectures and semantic web technologies to solve this problem. *Methods*. An engineering technique for constructing reconfigurable software models of mechatronic systems considered as discrete event systems is proposed. The software models building follows a hierarchical modular principle and is based on SOA. To build a model of an individual device as an automation object included in the system, the MVC design pattern is used. Each of the components of this pattern can be mapped to a separate service. *Results and conclusions*. Based on the proposed technique, software models of a PnP manipulator and a baggage handling system at the airport have been developed, suitable for their prototyping. To build these models, the software tools Oracle SOA Suite, OpenESB and Netbeans IDE were used. Owing to flexibility, reconfigurability and interoperability, the rapid transition from software models to real systems is possible.

Keywords: model, service-oriented architecture, web service, PnP manipulator, baggage handling system, BPEL.

Введение

Принятие концепции «Индустрия 4.0» в качестве одного из основных направлений развития промышленного производства будущего предполагает использование новых подходов к моделированию, проектированию и реализации мехатронных систем, лежащих в основе многих производственных процессов, например процессов сборочных и обрабатывающих производств, транспортных и логистических систем и т.д. Перспективным направлением является внедрение новых информационных технологий, в числе которых сервис-ориентированные архитектуры и технологии семантического веб, в проектирование подобного рода систем.

Сервис-ориентированная архитектура (СОА) – это модульный подход к разработке программного обеспечения, основанный на использовании распределенных, слабо связанных заменяемых компонентов, оснащенных стандартизированными интерфейсами для взаимодействия по стандартизированным протоколам [1]. Программные комплексы, разработанные в соответствии с СОА, обычно реализуются как набор веб-сервисов, взаимодействующих по протоколу *SOAP* (существуют и другие реализации). Основные принципы СОА: 1) слабое связывание; 2) повторное использование; 3) расширяемость.

Краеугольный камень архитектуры СОА – *сервис* (или *веб*-сервис при соответствующей реализации). Сервисы инкапсулируют детали реализации (операционную систему, платформу, язык программирования) от остальных компонентов, таким образом обеспечивая их комбинирование и многократное использование для построения сложных распределенных программных комплексов, а также независимость от применяемых платформ и инструментов разработки, способствуя масштабируемости и управляемости создаваемых систем. Сервисы взаимодействуют друг с другом путем обмена сообщениями, обеспечивая таким образом функциональность всей системы.

Базовая СОА состоит из поставщика (провайдера) сервисов, их потребителя и сервис-ориентированной инфраструктуры, которая в свою очередь включает репозиторий (каталог) сервисов и сервисную шину (рис. 1). Взаимодействие в системе выглядит довольно просто: поставщик регистрирует свои сервисы в реестре, а потребитель обращается к этому реестру с запросом. В случае веб-сервисов в системе используются так называемые *WSDL*-документы, являющиеся контрактами по активизации и хранящиеся в каталоге сервисов, из которого эти сервисы могут быть запрошены и извлечены посредством механизма *UDDI*.

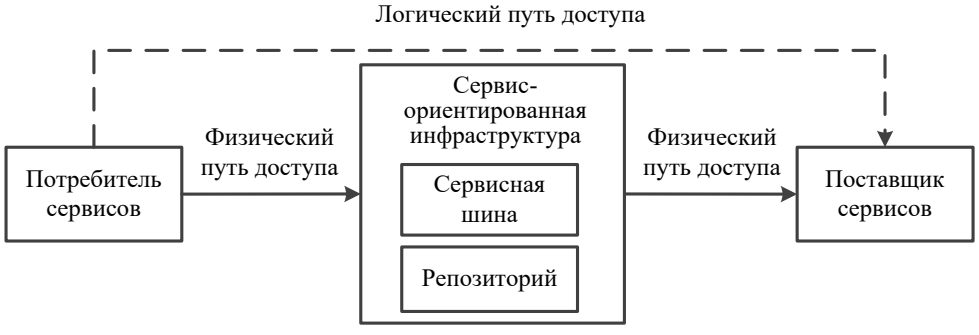

Рис. 1. Базовая сервис-ориентированная архитектура

Важную роль в СОА играет корпоративная сервисная шина (*Enterprise Service Bus*, *ESB*) [2]. По сути, она предоставляет магистральную сеть и инфраструктуру для соединения провайдеров и потребителей сервисов. Шина *ESB* позволяет взаимодействовать клиентам и поставщикам сервисов на основе *BPEL*, *Java*, *.NET* через протоколы *SOAP*, *REST*, *JNI*, *RNI*.

Достоинства СОА-подхода делают его довольно привлекательным для построения программных (имитационных) моделей мехатронных систем (МС), а также киберфизических систем (КФС) в целом. Обширной сферой, где используются МС, является область промышленной автоматизации.

Преимущества СОА были распознаны в сообществе по моделированию и симуляции, что нашло свое отражение в создании ряда сервис-ориентированных сред моделирования, в числе которых DUNIP, SOAD, D-CD++, DDSOS, SOHLA, FEDEP, XMSF, OGSA, Si3 [3]. В работе [4] дается обзор использования СОА в промышленных системах. Отмечается их широкое применение в информационных системах промышленных предприятий, при этом выявляются проблемы использования СОА в системах управления нижнего уровня, среди которых время ответа, поддержка событийного асинхронного параллельного выполнения, надежность и др. Разработке подходов, методов и инструментов проектирования на основе СОА в сфере интегрированной автоматизации посвящен европейский проект Arrowhead [5]. В работе [6] предлагается использовать семантические web-сервисы на языке OWL-S для управления промышленными процессами в реальном времени. Семантическая модель системы автоматически изменяется на основе механизма нотификации событий. В работе [7] представлена сервис-ориентированная реализация систем распределенной автоматики, где оркестровка сервисов основывается на использовании функциональных блоков стандарта IEC 61499. Данный подход был рассмотрен в работе [8] для построения системы транспортировки багажа в аэропорту.

Несмотря на значительный прогресс в области моделирования систем промышленной автоматики на основе принципов СОА, в настоящее время отсутствует простая инженерная методика построения программных моделей мехатронных систем на основе доступных инструментов данной архитектуры.

Материал и методика

Кратко изложим инженерную методику разработки программных моделей МС, рассматриваемых как дискретные событийные системы. Программная модель МС строится по иерархическому модульному принципу на основе СОА. Для создания модели отдельного устройства как объекта автоматизации, входящего в МС, используется паттерн проектирования Model/View/Controller (MVC) [9]. Каждая из компонент этого паттерна может быть отображена на отдельный сервис.

Методика построения программной модели устройства МС будет следующей.

1. Разработка формальной модели управления. В качестве таких моделей могут использоваться модели переходов состояний (конечные автоматы, сети Петри, формальные грамматики и т.д.), а также онтологические модели, позволяющие представить семантику. Разрабатывается структура состояния

модели, решается вопрос о хранении состояния во внешней памяти, поскольку одним из принципов СОА является отсутствие возможности хранения данных в сервисе (*statelessness*). В случае использования онтологий для хранения состояний может применяться конечная точка SPARQL. Разрабатывается интерпретатор формальной модели (явно или неявно), который программируется на языке реализации сервиса. Выделяются входные и выходные данные модели, которые отображаются на входы и выходы сервиса.

2. Разработка формальной модели физической части устройства сводится в простом случае к назначению временных задержек для каждой из операций, совершаемых в исполнительном механизме. В случае механизма со сложной логикой дополнительно могут быть использованы модели переходов состояний.

3. Разработка человеко-машинного интерфейса (ЧМИ) и визуального представления устройства. ЧМИ будет, по сути, представлять некий «пульт оператора», на который выводится информация о происходящих событиях в системе и о состоянии устройства и с которого могут быть посланы управляющие сигналы. В простом случае ЧМИ может не разрабатываться, а в продвинутых вариантах – приближаться к возможностям SCADA-систем.

4. Организация взаимодействия сервисов. При этом может использоваться как обычное клиент-серверное взаимодействие, так и взаимодействие с применением механизма нотификации (SOA 2.0). Для описания технологических процессов и взаимодействия сервисов может служить язык BPEL (*Business Process Execution Language*) [10]. Данный язык расширяет модель взаимодействия веб-сервисов и включает в нее поддержку транзакций. Структура взаимодействия сервисов может быть произвольной.

Результаты

В статье представлены две программные модели МС, а именно: модель PnP-манипулятора и модель системы транспортировки багажа (СТБ) в аэропорту, построенные в соответствии с приведенной методикой.

PnP-манипулятор (*Pick-and-Place manipulator*) предназначен для отбора деталей с рабочей плоскости и их переноса в буферную зону [11]. 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 ее. Пример манипулятора с двумя горизонтальными и одним вертикальным цилиндром показан на рис. 2. Данный манипулятор может подбирать одну из трех деталей (pp1, pp2, pp3) и доставлять ее в буфер для их хранения (pp0).

Работа PnP-манипулятора может быть описана набором временных диаграмм, одна из которых, связанная с переносом детали из места pp3, приведена на рис. 3. На рисунках 2 и 3 приняты следующие обозначения: 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.

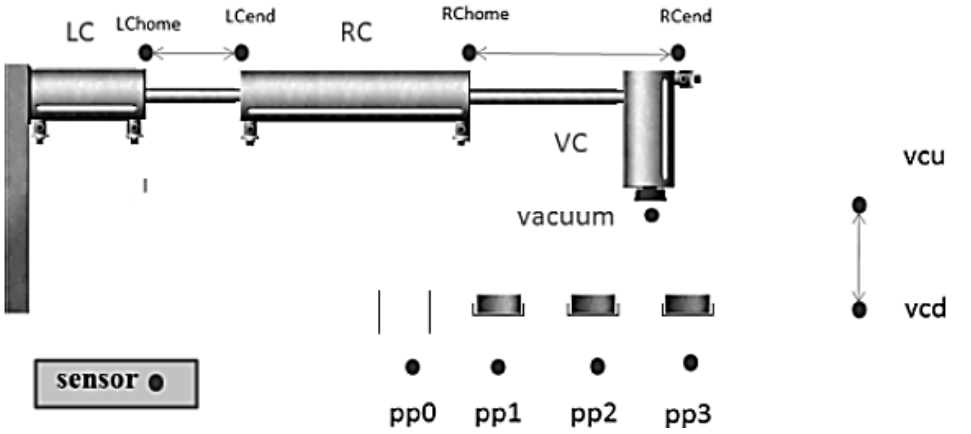

Рис. 2. Структура манипулятора «Pick and Place»

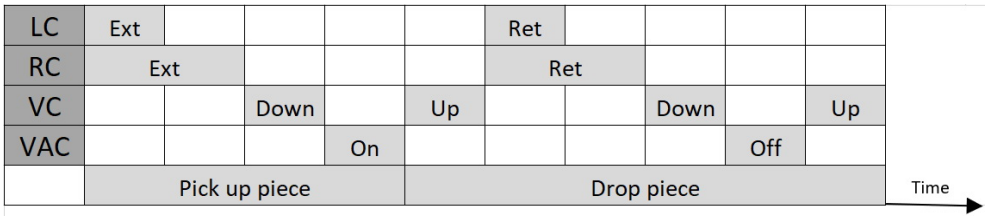

Рис. 3. Временная диаграмма работы PnP-манипулятора

В работе [11] была предложена сервис-ориентированная реализация системы управления PnP-манипулятором, основанной на функциональных блоках стандарта IEC 61499. В отличие от данной работы в статье предлагается 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 использовалась модель конечных автоматов. Модель оборудования представлена в виде временных задержек. При разработке композитных сервисов применялся язык BPEL.

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 – композитными.

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.

Контроллер горизонтального цилиндра в сервисе *CylinderController* представлен в виде конечного автомата со следующими состояниями: *init, cylAtHome*, *cylAtEnd*, *GoForward*, *GoBackward*. 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 идет по одной из трех 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 рис. 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.

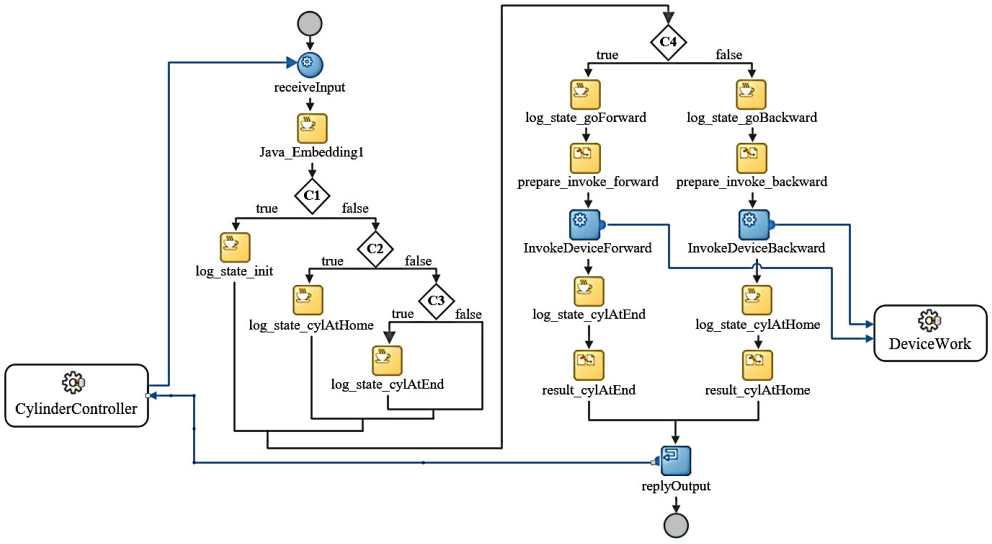

Рис. 4. BPEL-диаграмма для сервиса горизонтального цилиндра

Как и в случае контроллера горизонтального цилиндра, контроллер вертикального цилиндра (сервис *VerticalCylinderController*) представлен в виде конечного автомата. Данный сервис в процессе своего выполнения вызывает сервисы *Device* и *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.

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. Следует отметить, что сервис-ориентированная модель PnP-манипулятора разработана и реализована на основе программного средства Oracle SOA Suite [12].

На рисунке 5 представлена система транспортировки багажа (СТБ) аэропорта. Описание ее работы можно найти, например, в работе [13].

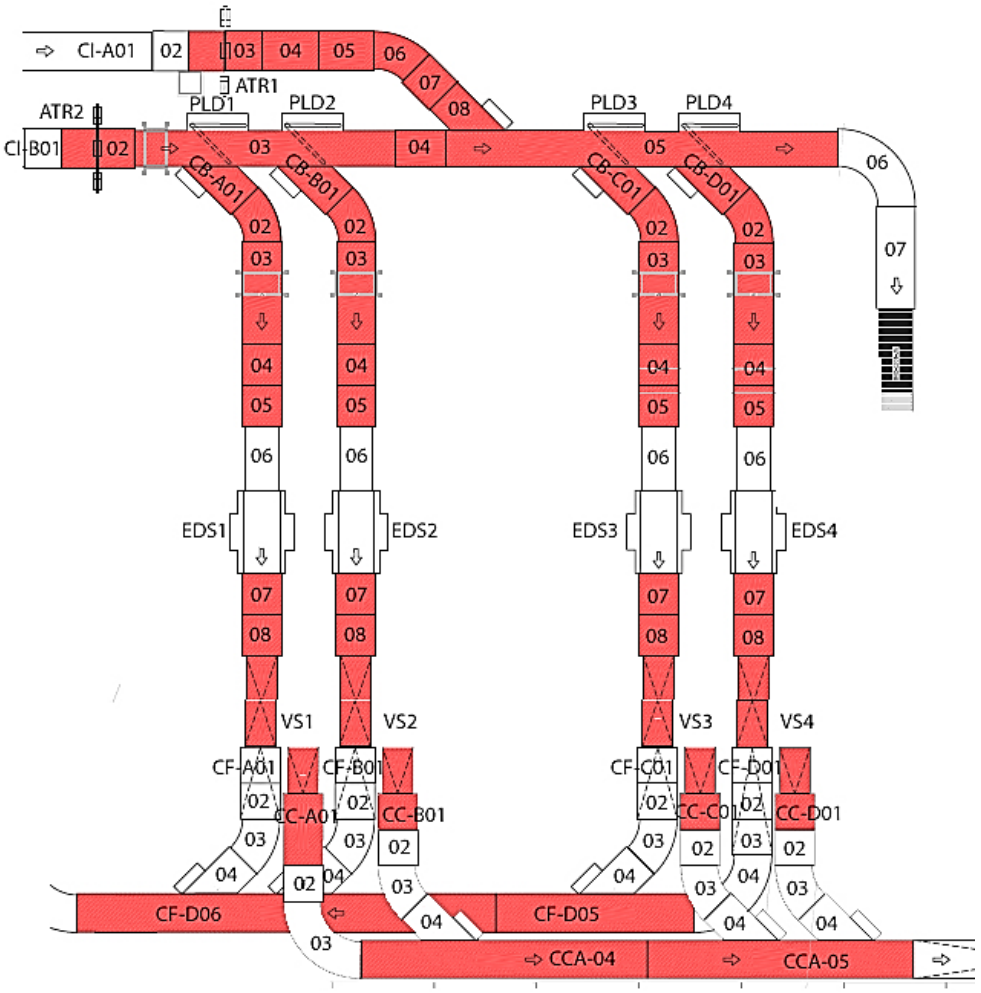

Рис. 5. Структура системы транспортировки багажа аэропорта

Для моделирования использовалась упрощенная структура (рис. 6). Входящий багаж поступает на первый конвейер C1. Перед началом его работы запускается сканер штриховых кодов (баркод-сканер), считывающий информацию о месте назначения пришедшей единицы багажа. За ветвление маршрута багажа отвечает второй элемент выталкивания D2. После отработки баркод-сканера запускается первый конвейер. Доставив элемент до второго конвейера, он завершает свою работу. Далее запускается система

рентгеновской проверки содержимого багажа (X1). При наличии подозрительных объектов багажный элемент будет перемещен на пятый конвейер (C5) первым элементом выталкивания (D1), в противном случае багаж продолжит свое следование по третьему конвейеру (C3). В зависимости от целевого направления багажа, определенного баркод-сканером, багажный элемент может двигаться по линии C4 или C6 (этим управляет второй выталкиватель D2).

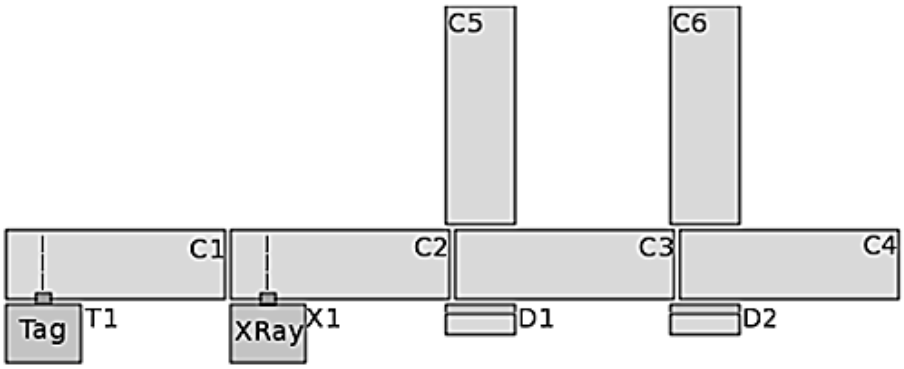

Рис. 6. Моделируемый фрагмент системы транспортировки багажа

Для моделирования структуры СТБ (см. рис. 6) были разработаны следующие элементарные сервисы: сервис баркод-сканера (*TagReaderWebService*), сервис конвейерной линии (*ConveyorWebService*), сервис рентгеновской установки для проверки багажа (*XRayWebService*), сервис выталкивателя (*DiverterWebService*). Элементарные сервисы взаимодействуют в рамках композитного сервиса *CompositeApp*. Для практической реализации модели СТБ аэропорта была выбрана платформа *OpenESB* [14], представляющая сервисную шину с открытым исходным *Java*-кодом. *OpenESB* основана на стандартах *JBI*, *XML*, *XML Schema*, *WSDL*, *BPEL* и композитных приложениях SOA. *OpenESB* является свободной версией стандартизированной интеграционной платформы для сервера приложений *Glassfish*. Инструмент разработки интеграционных процессов и сервисов позволяет проектировать процессы в спецификации языка *BPEL* и взаимодействовать с другими системами предприятия на основе интеграционных компонентов *Java Business Integration* (*JBI*). В качестве среды разработки была выбрана *Netbeans IDE*, обеспечивающая удобное взаимодействие с инструментами *OpenESB* [15].

Сервис конвейера будет иметь одну операцию *move*, на вход которой будет подаваться идентификатор *id* конвейера. Ответом об успешности выполненных операций будет текстовое сообщение «*done*». Остальные элементарные сервисы СТБ в нашем случае будут иметь подобные входы-выходы. Необходимо сгенерировать *WSDL*-документ (где указывается адрес сервиса) для доступа к сервису со стороны других объектов системы.

Основное приложение реализовано на языке *BPEL*. В проекте использовалось два оператора условного выполнения операций. В первом случае анализировалось выполнение работы датчика рентгеновской проверки багажа. Второй случай – это анализ значения, поступившего с сервиса чтения

штрихкода багажа. На рисунке 7 представлен фрагмент *BPEL*-диаграммы, содержащий операции ветвления для второго случая.

Полная диаграмма *BPEL* содержит 28 операторов. Начало и конец диаграммы составляют операции приема входных значений и отправки результата в инициализирующую связь *initWSDL*. Далее идут парные операции присвоения идентификаторов требуемых объектов СТБ и вызов соответствующего веб-сервиса с заданными параметрами: сервиса баркод-сканера, сервиса конвейерной линии для первого конвейера, рентген-сервиса, сервиса конвейерной линии для второго конвейера. Сервис баркод-сканера возвращает число из диапазона доступных параллельных линий конвейера (в нашем случае их две). Рентгеновский сервис – одно из константных значений *accept* или *reject* при успешном прохождении проверки либо наличии подозрительных объектов соответственно. Далее процесс разветвляется. В зависимости от значения, возвращенного рентген-сервисом, вызывается либо сервис для первого элемента выталкивания с сервисом для пятого конвейера – в случае наличия подозрительных объектов на проверяемом багаже, либо «конвейерный» сервис для третьего конвейера – при успешном прохождении проверки багажным элементом.

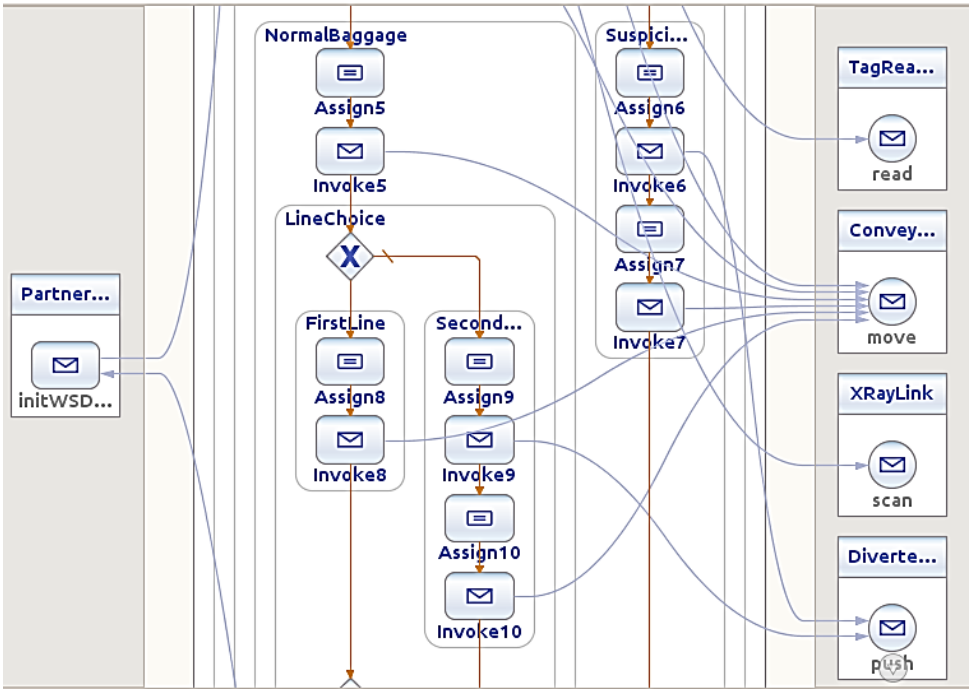

Рис. 7. Элемент *BPEL*-диаграммы, содержащий операции ветвления

BPEL-модуль не может быть самостоятельно развернут на веб-сервере. Для этого требуется создание *Composite Application* проекта и помещение в него *BPEL*-проекта (рис. 8). Для тестирования системы в каждый веб-сервис ее устройств была добавлена система логирования, позволяющая отслеживать состояние в реальном режиме времени.

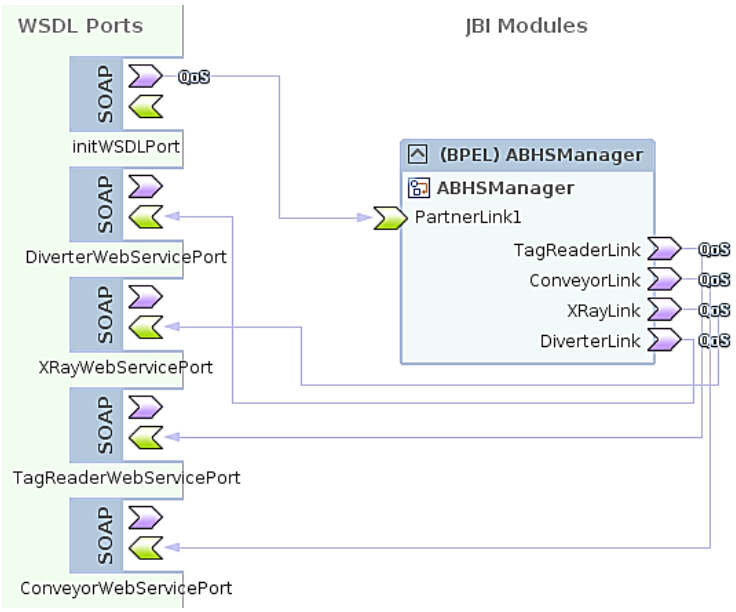

Рис. 8. Композитное приложение

Выводы

В представленной статье показано, как на основе доступных инструментов разработать программные модели мехатронных систем, основанные на принципах СОА. Данный вид моделей имеет ряд преимуществ, среди которых гибкость, возможность быстрой реконфигурации, интеграции, комплексирования и повторного использования модулей, масштабируемость, интероперабельность, работа в контексте сетевых архитектур, возможность построения гетерогенных систем. Сервисы могут быть реализованы на любом языке программирования и на различных платформах. Благодаря гибкости модели и модульности построения возможна быстрая замена программной модели физической части на реальное оборудование (через программные адаптеры) и, таким образом, переход к реальной системе.

На основе предложенной методики разработаны и протестированы программные модели PnP-манипулятора и системы транспортировки багажа в аэропорту, пригодные для их прототипирования. На основе полученного опыта можно сделать вывод, что для быстрого прототипирования систем предпочтительнее использовать инструмент OpenESB в связке с Netbeans IDE, чем Oracle SOA Suite, из-за сложности и громоздкости последнего.

Направлением дальнейшей работы является использование семантических сервисов (в том числе на основе OWL-S) [16], облачных сервисов, а также технологий семантического веб при построении программных моделей МС.

Библиографический список

1. *Erl, T.* Service-Oriented Architecture: Concepts, Technology and Design: 1st Edition / T. Erl. – Prentice Hall, 2016. – 792 p.
2. *Робинсон, Р.* Сценарии и решения использования шины Enterprise Service Bus в сервис-ориентированной архитектуре / Р. Робинсон. – 2007. – Ч. 1. – 11 с. – URL: https://www.ibm.com/developerworks/ru/library/ws-esbscen/index.html

3. *Wang, W.* Service-oriented simulation framework: An overview and unifying methodology / W. Wang, W. Wang, Y. Zhu, Q. Li // Simulation. – 2011. – Vol. 87, № 3. – P. 221–252.
4. *Komoda, N.* Service Oriented Architecture (SOA) in Industrial Systems / N. Komoda // 4th IEEE International Conference on Industrial Informatics (INDIN'2006), Singapore 16–18 Aug. 2006. – Singapore, 2006. – P. 1–5.
5. The arrowhead approach for SOA application development and documentation / F. Blomstedt, L. L. Ferreira, M. Klisics, C. Chrysoulas, I. M. Soria, B. Morin, A. Zabasta, J. Eliasson, M. Johansson, P. Varga // 40th Annual Conference of the IEEE Industrial Electronics Society (IECON 2014). – Dallas, TX, USA, 2014. – P. 1–7.
6. *Puttonen, J.* Semantics-based Composition of Factory Automation Processes Encapsulated by Web Services / J. Puttonen, A. Lobov, J. L. M. Lastra // IEEE Transactions on Industrial Informatics. – 2013. – Vol. 9, iss. 4. – P. 2349–2359.
7. *Dai, W.* Bridging Service-Oriented Architecture and IEC 61499 for Flexibility and Interoperability / W. Dai, V. Vyatkin, J. H. Christensen, V. N. Dubinin // IEEE Transactions on Industrial Informatics. – 2015. – Vol. 11, iss. 3. – P. 771–781.
8. *Dai, W.* The Application of Service-Oriented Architectures in Distributed Automation Systems / W. Dai, V. Vyatkin, J. H. Christensen // IEEE International Conference on Robotics & Automation (ICRA'2014). – Hong Kong, China, 2014. – P. 252–257.
9. *Christensen, J. H.* Design patterns for systems engineering with IEC 61499 / J. H. Christensen // Verteilte Automatisierung – Modelle und Methoden fur Entwurf, Verifikation, Engineering und Instrumentierung (VA2000). – Magdeburg, Germany, 2000. – P. 63–71.
10. BPEL Designer and Service Engine User's Guide. – SunMicrosystems, Inc., 2009. – 180 p.
11. Automation Services Orchestration with Function Blocks: Web-service Implementation and Performance Evaluation / E. Demin, V. Dubinin, S. Patil, V. Vyatkin // Studies in Computational Intelligence. – 2016. – Vol. 640. – P. 213–221.
12. *Reynolds, A.* Oracle SOA Suite 11g R1 Developer's Guide / A. Reynolds, M. Wright. – Packt Publishing, 2010. – 722 p.
13. *Black, G.* Intelligent Component – based Automation of Baggage Handling Systems with IEC 61499 / G. Black, V. Vyatkin // IEEE Transactions on Automation Science and Engineering. – 2009. – Vol. 7, iss. 2. – P. 337–351.
14. OpenESB: Manage your services in a smart way // OpenESB official website. – URL: https://www.open-esb.net/
15. *Shin, S.* SOA using OpenESB, BPEL and Netbeans / S. Shin // Huihoo website – URL: https://docs.huihoo.com/openesb/soabpelopenesb.pdf
16. *Pedrinaci, C.* Semantic Web Services / C. Pedrinaci, J. Domingue, A. P. Sheth // Handbook of Semantic Web Technologies. – Berlin ; Heidelberg : Springer, 2011. – P. 977–1035.

References

1. Erl T. *Service-Oriented Architecture: Concepts, Technology and Design: 1st Edition*. Prentice Hall, 2016, 792 p.
2. Robinson R. *Stsenarii i resheniya ispol'zovaniya shiny Enterprise Service Bus v servis-orientirovannoy arkhitekture* [Scenarios and solutions for using the Enterprise Service Bus in a service-oriented architecture]. 2007, part 1, 11 p. Available at: https://www.ibm.com/developerworks/ru/library/ws-esbscen/index.html [In Russian]
3. Wang W., Zhu Y., Li Q. *Simulation*. 2011, vol. 87, no. 3, pp. 221–252.
4. Komoda N. *4th IEEE International Conference on Industrial Informatics (INDIN'2006), Singapore 16–18 Aug. 2006*. Singapore, 2006, pp. 1–5.
5. Blomstedt F., Ferreira L. L., Klisics M., Chrysoulas C., Soria I. M., Morin B., Zabasta A., Eliasson J., Johansson M., Varga P. *40th Annual Conference of the IEEE Industrial Electronics Society (IECON 2014)*. Dallas, TX, USA, 2014, pp. 1–7.

6. Puttonen J., Lobov A., Lastra J. L. M. *IEEE Transactions on Industrial Informatics*. 2013, vol. 9, iss. 4, pp. 2349–2359.
7. Dai W., Vyatkin V., Christensen J. H., Dubinin V. N. *IEEE Transactions on Industrial Informatics*. 2015, vol. 11, iss. 3, pp. 771–781.
8. Dai W., Vyatkin V., Christensen J. H. *IEEE International Conference on Robotics & Automation (ICRA'2014)*. Hong Kong, China, 2014, pp. 252–257.
9. Christensen J. H. *Verteilte Automatisierung – Modelle und Methoden fur Entwurf, Verifikation, Engineering und Instrumentierung (VA2000)* [Distributed automation models and methods for design, verification, Engineering, and instrumentation (VA2000)]. Magdeburg, Germany, 2000, pp. 63–71.
10. *BPEL Designer and Service Engine User's Guide*. SunMicrosystems, Inc., 2009, 180 p.
11. Demin E., Dubinin V., Patil S., Vyatkin V. *Studies in Computational Intelligence*. 2016, vol. 640, pp. 213–221.
12. Reynolds A., Wright M. *Oracle SOA Suite 11g R1 Developer's Guide*. Packt Publishing, 2010, 722 p.
13. Black G., Vyatkin V. *IEEE Transactions on Automation Science and Engineering*. 2009, vol. 7, iss. 2, pp. 337–351.
14. *OpenESB: Manage your services in a smart way*. OpenESB official website. Available at: https://www.open-esb.net/
15. Shin S. *SOA using OpenESB, BPEL and Netbeans*. Huihoo website Available at: https://docs.huihoo.com/openesb/soabpelopenesb.pdf
16. Pedrinaci C., Domingue J., Sheth A. P. *Handbook of Semantic Web Technologies*. Berlin; Heidelberg: Springer, 2011, pp. 977–1035.

Дубинин Виктор Николаевич
доктор технических наук, профессор,
кафедра вычислительной техники,
Пензенский государственный университет
(Россия, г. Пенза, ул. Красная, 40)
E-mail: dubinin.victor@gmail.com

Dubinin Victor Nikolaevich
doctor of technical sciences, professor,
sub-department of computer engineering,
Penza State University
(40 Krasnaya street, Penza, Russia)

Дубинин Алексей Викторович
студент,
Пензенский государственный университет
(Россия, г. Пенза, ул. Красная, 40)
E-mail: dubinin.aleksey@gmail.com

Dubinin Alexey Victorovich
student,
Penza State University
(40 Krasnaya street, Penza, Russia)

Ручкин Михаил Алексеевич
студент,
Пензенский государственный университет
(Россия, г. Пенза, ул. Красная, 40)
E-mail: ruchmix@mail.ru

Ruchkin Mikhail Alekseevich
student,
Penza State University
(40 Krasnaya street, Penza, Russia)

Образец цитирования:
Дубинин, В. Н. Программные модели мехатронных систем на основе сервис-ориентированной архитектуры / В. Н. Дубинин, А. В. Дубинин, М. А. Ручкин // Модели, системы, сети в экономике, технике, природе и обществе. – 2020. – № 4 (36). – С. 97–108. – DOI 10.21685/2227-8486-2020-4-10.