



Применение онтологического подхода в задаче генерации событийных данных с помощью имитационных моделей

© 2023, А.М. Наместников

Ульяновский государственный технический университет, Ульяновск, Россия

Аннотация

Описывается применение онтологического подхода к решению задачи генерации событийных данных, поступающих из журналов имитационных экспериментов. В настоящее время в рамках научного направления «Интеллектуальный анализ процессов» развиваются методы и алгоритмы, позволяющие решать задачи машинного обучения применительно к событийным данным. Имитационное моделирование в данном случае может играть важную роль для формирования обучающих выборок. Однако экспериментальные результаты имитации в виде журналов определённой структуры необходимо приводить к виду событийных журналов так, как они понимаются в интеллектуальном анализе процессов. В данной работе приводится постановка задачи формирования онтологического ресурса, позволяющего сформировать журнал событий по результатам имитационных экспериментов с дискретно-событийной моделью, в которой заявки на обработку представлены в виде агентов. Приводится формальное описание онтологии предметной области и алгоритм её доопределения на основе данных журналов имитационной модели. В качестве объекта имитации в работе предлагается рассматривать иерархическую систему принятия решений, в которую поступают задачи различной сложности. Уровень сложности задач является определяющим для выбора уровня иерархии, на котором данную задачу требуется решать. Приводится архитектура разработанной онтологической системы, а также структура понятий с соответствующими семантическими отношениями и наборами экземпляров.

Ключевые слова: онтология, интеллектуальный анализ процессов, событийные данные, имитационная модель, агент.

Цитирование: Наместников А.М. Применение онтологического подхода в задаче генерации событийных данных с помощью имитационных моделей // Онтология проектирования. 2023. Т.13, №2(48). С.243-253. DOI:10.18287/2223-9537-2023-13-2-243-253.

Финансирование: исследование выполнено в рамках государственного задания №075-00233-20-05 по проекту «Исследование интеллектуального предиктивного мультимодального анализа больших данных и извлечения знаний из разных источников»

Конфликт интересов: автор заявляет об отсутствии конфликта интересов.

Введение

Интеллектуальный анализ процессов (ИАП) является относительно новой областью исследования, которая находится на пересечении вычислительного интеллекта, интеллектуального анализа данных, моделирования и анализа процессов [1, 2]. Интеллектуальный анализ процессов предназначен для решения задач извлечения, мониторинга и усовершенствования реальных процессов. В последнее время растущий интерес исследователей к задачам ИАП привёл к созданию рабочей группы [2].

Журналы событий имитационных моделей (ИМ) сложных систем могут содержать важную информацию для последующего анализа их поведения. Результаты анализа событийных данных позволяют выполнять постоянный мониторинг отклонений в работе реальных систем, извлекая необходимую информацию из журналов событий ИМ и сравнивая её с модельным поведением.

Методы ИАП должны иметь доступ к последовательности событий, записанных в журнале. Каждое событие относится к определённому действию и является частью экземпляра процесса. Допускается, что журналы событий могут содержать дополнительную информацию о возникающих в системе событиях. Для этого используются дополнительные атрибуты событий, например, такие как ресурс или временная отметка.

Для систем имитационного моделирования особую важность приобретают следующие ракурсы ИАП.

- Ракурс потока управления нацелен на демонстрацию структуры выполнения действий и их порядок. Его цель заключается в получении описания и характеристик всех используемых путей выполнения процесса.
- Организационный ракурс применяется для представления информации об используемых ресурсах, зафиксированных в журнале событий (например, люди, системы, организационные структуры и то, как они взаимосвязаны друг с другом).
- Временной ракурс непосредственно связан с показателями времени и частоты событий. В том случае, когда у события имеются временные метки, записанные в журнале, появляется возможность анализировать «узкие места», эффективность использования ресурсов и прогнозировать оставшееся время обработки.

Имитационное моделирование является очевидным подходом для генерации событийных данных, однако его непосредственное применение для указанных задач является затруднительным. Основная сложность заключается в существенном отличии представления событийных данных в ИМ и в ИАП. Требуется выполнить трансформацию событийных данных из контекста имитационного моделирования (где используются понятия агента, сообщения, состояния, функционального блока модели) в контекст ИАП (включающего понятия экземпляра процесса, действия, события, ресурса и т.д.).

С точки зрения задач анализа ИМ прикладная онтология, как средство представления знаний, может найти применение для решения следующих задач [3, 4].

- Системный анализ предметной области (ПрО) функционирования сложных систем. Онтологические ресурсы предоставляют частично формализованный и структурированный базис для выполнения системного анализа ПрО.
- Интегрирование данных и знаний. В процессе семантического анализа журналов событий функциональных подсистем прикладная онтология способна устанавливать семантическую эквивалентность тождественных событий, действий и дополнительных атрибутов, которые, как правило, представляются с использованием различных терминов.

В работе рассмотрена формальная модель онтологии событийных данных и её применение для формирования журнала событий в иерархической системе обработки заявок различной сложности.

1 Онтологическая модель событийных данных

Онтология событийных данных, генерируемых ИМ, включает в себя следующие компоненты:

$$O_D = \langle C_D, R_D, F_D, I_D \rangle, \quad (1)$$

где C_D – множество понятий ПрО, R_D – множество семантических отношений онтологии, F_D – множество функций интерпретации, позволяющее формировать новые факты базы знаний, I_D – множество экземпляров (индивидов), относящиеся к понятиям онтологии.

Множество понятий C_D включает три компоненты знаний о терминологии ПрО моделирования и анализа иерархических систем:

$$C_D = \{C_D^{St}, C_D^{Ms}, C_D^{Im}, C_D^{Pm}\}, \quad (2)$$

где C_D^{St} – множество понятий диаграмм состояний ИМ; C_D^{Ms} – множество понятий, включаемых в процессы обмена сообщениями между агентами модели; C_D^{Im} – множество понятий, описывающих структурные элементы ИМ; C_D^{Pm} – множество понятий, представляющих структурные особенности журнала событий в контексте ИАП.

Аналогичным образом определяются множества семантических отношений $R_D = \{R_D^{St}, R_D^{Ms}, R_D^{Im}, R_D^{Pm}\}$ и множество функций интерпретации $F_D = \{F_D^{St}, F_D^{Ms}, F_D^{Im}, F_D^{Pm}\}$.

Место онтологии событийных данных представлено на рисунке 1. В контексте ИАП журнал событий формируется на основе трёх видов журналов имитационного моделирования: обмена сообщениями; состояний агентов; прохождения агентов по ИМ.



Рисунок 1 – Место онтологии событийных данных в процессе формирования журнала событий

Перечисленные виды журналов генерируются по результатам прогонов ИМ. Основной состав атрибутов каждого журнала представлен на рисунке 1. В журнале обмена сообщениями содержится информация о времени возникновения сообщений в модели и об агентах, которые отправляют и принимают различного вида сообщения. Журнал состояний агентов содержит информацию о динамике поведения агентов с привязкой к временным отметкам, которые представляют время вхождения агентов в определённое состояние и время выхода из состояния. Журнал прохождения агентов по ИМ содержит данные о том, какие агенты поступают в элементы ИМ с привязкой ко времени.

Таким образом, в рамках данной статьи рассматривается комбинированный подход к построению ИМ, который подразумевает использование принципов агентного моделирования и дискретно-событийного моделирования.

Онтология ПрО состоит из двух компонент: *TBox* и *ABox* (множества логических утверждений о терминах и множества фактов, соответственно). Механизм логического вывода позволяет генерировать новые факты, которые позволяют формировать журнал событий так, как он представляется в задачах ИАП. Состав журнала событий включает информацию об

экземпляре процесса. Под процессом понимается конкретная реализация процесса обработки задачи в иерархической системе. Идентификатор события определяет уникальный номер строки журнала событий. Метка времени является обязательной для процедуры упорядочивания событий. Деятельность представляет собой функцию преобразования данных в процессе обработки задачи. Дополнительно журнал событий может включать атрибуты, которые представляют информацию о субъектах деятельности, стоимости или других параметрах, которые относятся к конкретному событию.

На рисунке 2 представлена структура онтологии событийных данных, используемая для формирования журнала событий. Множество понятий включает в себя два подмножества:

$$C_D = C_D^{fix} \cup C_D^{gen},$$

где C_D^{fix} – множество понятий ПрО, которое формируется экспертным способом и является инвариантным относительно конкретной ИМ; C_D^{gen} – генерируемое множество понятий ПрО на основе содержимого событийных журналов конкретной ИМ.

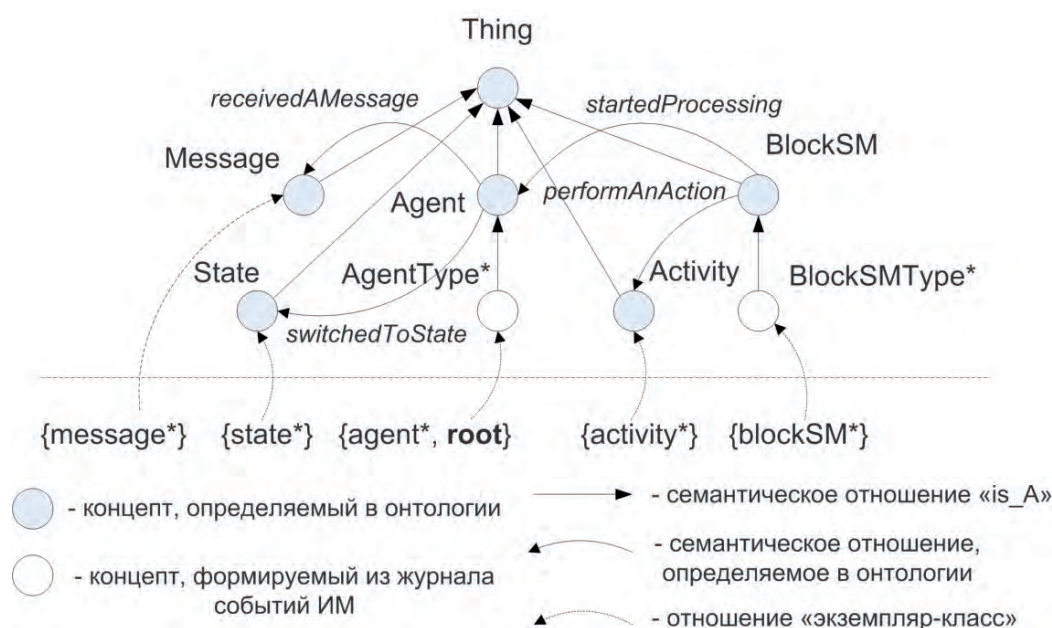


Рисунок 2 – Структура онтологии событийных данных

Обозначение класса с символом (*) поясняет факт наличия нескольких классов, количество которых заранее неизвестно. На рисунке 2 такими понятиями являются *AgentType** и *BlockSMType**. Индивиды понятий онтологии ПрО на рисунке представлены с помощью множеств $\{message^*\}$, $\{state^*\}$, $\{agent^*, root\}$, $\{activity^*\}$ и $\{blockSM^*\}$. Здесь индивид *root* выделен среди остальных индивидов как предопределённый (т.е. он присутствует в каждой ИМ в любом случае). Концепт «Агент» имеет вид:

$$Agent \equiv T \sqcap \exists receivedMessage.Message \sqcap \exists switchedToState.State,$$

где *receivedMessage* и *switchedToState* – наименование ролей «получено сообщение» и «перешёл в состояние», соответственно; *Message* – домен типа «Сообщение», *State* – домен типа «Состояние».

Концепт «Блок ИМ» имеет вид логического выражения онтологии:

$$BlockSM \equiv T \sqcap \exists startedProcessing.Agent \sqcap \exists performAnAction.Activity,$$

где *startedProcessing* и *performAnAction* – наименование ролей «начата обработка» и «выполнение действия», соответственно; *Agent* – домен типа «Агент», *Activity* – домен типа «Действие».

2 Алгоритм онтологического формирования журнала событий

Для генерации журнала событий необходимо выполнить логический вывод по онтологии. Существует несколько причин, по которым такая процедура должна иметь место.

- Все экземпляры процессов, которые подвергаются анализу, должны быть полностью завершёнными. Для этого определяется отдельный класс онтологии *CompleteAgent*, который является подклассом понятия *Agent* и содержит только те экземпляры (конкретные агенты, появляющиеся в ИМ), которые получили начальное и конечное сообщения:

$$\begin{aligned} \text{CompleteAgent} &\equiv \\ \text{Agent} \sqcap \exists \text{receivedMessage}.\text{StartMessage} \sqcap \\ &\exists \text{receivedMessage}.\text{FinalMessage}, \end{aligned}$$

где *StartMessage* – подкласс понятия *Message*, который содержит сообщения, инициирующие создание агента, *FinalMessage* – подкласс понятия *Message*, который содержит сообщения, относящиеся к моменту удаления агента из ИМ.

Рассматривая агентов с полным жизненным циклом, имеется возможность учитывать только те фрагменты журнала обмена сообщениями, которые описывают полные экземпляры процессов.

- Не все блоки ИМ (экземпляры класса *BlockSM*) имеют отношение к моделированию пользовательских действий по обработке заданий. Некоторые из блоков носят исключительно системный характер, и соответствующие действия модели не должны быть включены в процедуру анализа поведения системы. Для исключений действий системных блоков ИМ создан класс онтологии *SystemActivity*, который является подклассом понятия *Activity* и содержит только те действия, которые выполняют системную функцию в ИМ (которые в последующем анализе не включены в поведенческую модель процесса):

$$\text{SystemActivity} \equiv \text{Activity} \sqcap \forall \text{performedBy}.\text{SystemBlockSM},$$

где свойство *performedBy* является обратным свойством для *performAnAction*, концепт *SystemBlockSM* – подкласс концепта *BlockSM*. В данном случае используется универсальное ограничение на класс онтологии.

Алгоритм онтологического формирования журнала событий включает следующие шаги.

Шаг 1. Доопределение множества понятий онтологии (1) C_D^{fix} понятиями C_D^{gen} . Для этого необходимая информация извлекается из журнала обмена сообщениями, журнала состояний агентов и журнала прохождения агентов по ИМ.

Шаг 2. Доопределение онтологии фактической информацией. На данном шаге каждое событие из журналов, анализируемых на предыдущем шаге, преобразуется в факты вида:

$$\text{instance:Class} \text{ и } < \text{instance1}, \text{instance2} >:\text{Relation}.$$

Таким образом, множество понятий $C_D = C_D^{fix} \cup C_D^{gen}$ с соответствующими семантическими отношениями и функциями интерпретации, а также содержимое базы фактов *ABox*, сформированное на данном шаге, определяют то содержимое онтологии, которое необходимо для выполнения логического вывода.

Шаг 3. Выполнение логического вывода по базе знаний (*TBox* $\cup$ *ABox*) с целью формирования класса событий, которые являются инициаторами деятельности, входящих в состав событий, как экземпляров соответствующего журнала.

Шаг 4. Восстановление структуры $< \text{instance}, \text{event}, \text{timestamp}, \text{activity} >$ как кортежа журнала событий для последующего анализа.

3 Реализация системы генерации событийных данных

3.1 Архитектура системы

Архитектура системы генерации событийных данных приведена на рисунке 3 и включает подсистемы онтологического моделирования и генерации журнала событий.



Рисунок 3 – Архитектура системы генерации событийных данных

В основе подсистемы онтологического моделирования находятся: библиотека *OWLReady 2*, *RDF*-хранилище, реализованное с помощью *SQL*-базы данных *SQLite 3*, механизм логического вывода *HermiT*. Среда выполнения *OWLReady 2* позволяет оперировать классами (понятиями) онтологии, экземплярами (индивидами) и свойствами. Указанные структуры сохраняются в *RDF*-хранилище [5-7]. Реализация данной подсистемы позволяет выполнять сохранение онтологии в формате *RDF/XML* [8] с последующей загрузкой в редактор *Protege*. Выполнение логического вывода возможно как непосредственно в среде *OWLReady*, так и средствами онтологического редактора.

Задачей подсистемы онтологического моделирования является формирование семантической модели ПрО, которая определяет ограничения для генерации журнала событий.

Подсистема генерации журнала событий состоит из двух основных модулей: анализа ограничений и формирования событийной структуры.

Модуль анализа ограничений взаимодействует с онтологией ПрО с целью определения классов-ограничений. Их содержимое позволяет определить перечень агентов ИМ, который имеют полный жизненный цикл и, следовательно, могут использоваться в генерации журнала событий. Кроме того, формируются ограничения на действия, выполняемые в процессе прогона ИМ. Содержимое журналов ИМ позволяет выполнить доопределение онтологии ПрО необходимыми классами (понятиями) и экземплярами классов (индивидами).

Модуль анализа ограничений возвращает список агентов ИМ для передачи его в модуль формирования событийной структуры. Входами для данного модуля являются список агентов, список системных действий и журналы ИМ.

3.2 Иллюстративный пример задачи моделирования

В качестве примера системы, поведение которой представляется в виде журнала событий так, как он понимается в ИАП, рассмотрена иерархическая система обработки задач. Для каждой задачи определяется её сложность, которая влияет на то, на каком уровне иерархии системы она будет решаться. Чем сложнее задача, тем выше по иерархии располагается уровень её решения (рисунок 4). С фоновым заполнением обозначены те узлы иерархической системы, которые принимают участие в решении конкретной задачи. Утолщёнными рёбрами между узлами определён путь прохождения задачи.

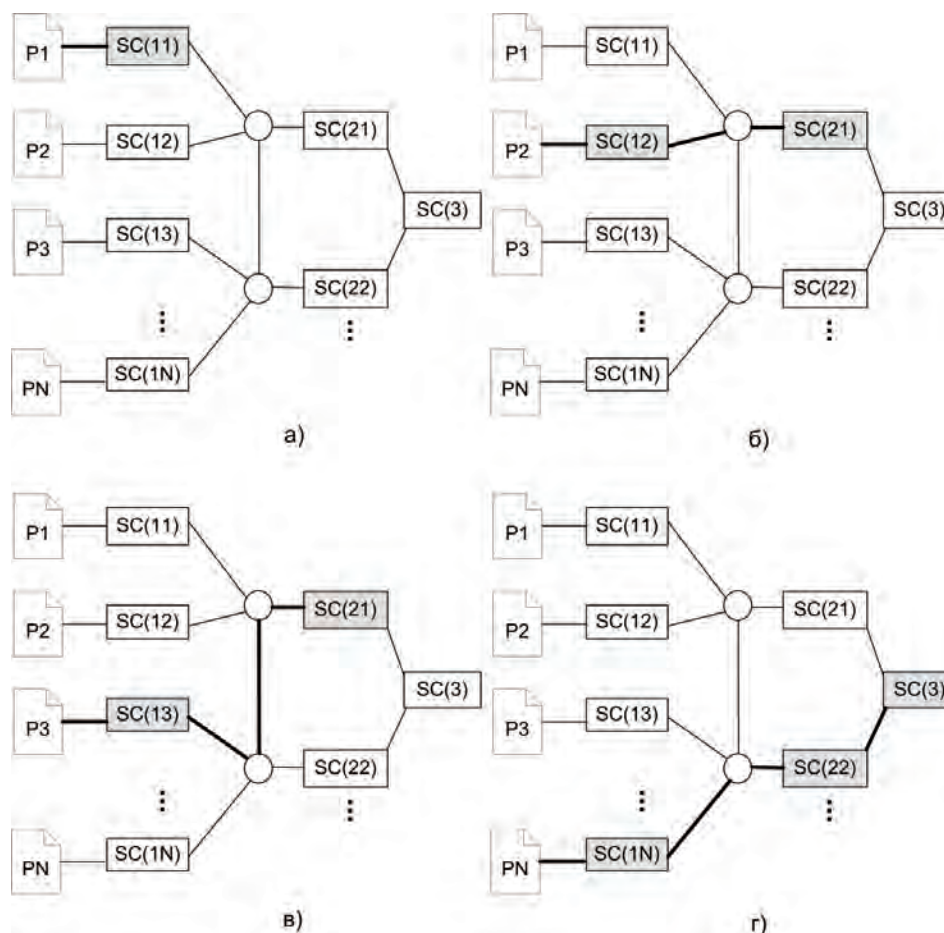


Рисунок 4 – Иллюстративный пример иерархической системы обработки задач

На рисунке 4а представлена самая простая ситуация, когда задача невысокой сложности (P1) решается на первом уровне принятия решения SC(11). Рисунок 4б соответствует ситуа-

ции с более сложной задачей (P2), для решения которой требуется прохождение узлов на первом уровне SC(12) и на втором уровне SC(21).

Построенная ИМ учитывает, что узлы иерархической системы могут находиться в нерабочем состоянии (в модели для этих целей используются стохастические параметры выхода из строя узлов системы). Рисунок 4в иллюстрирует такую ситуацию. Задача (P3) является сложной и требует своего решения на втором уровне. Узел SC(22) является недоступным и система моделирования передаёт задачу (P3) на узел SC(21). Наконец, на рисунке 4г представлен иллюстративный пример решения задачи повышенной сложности (PN), где задействованы все уровни принятия решения.

3.3 Реализация онтологии событийных данных

Онтология событийных данных формируется в два этапа: сначала создаётся «каркас» онтологии из понятий верхнего уровня, инвариантных относительно реализуемых ИМ; после имитации на основе журналов доопределяются остальные понятия и индивиды онтологии. На рисунке 5 представлен пример сформированной онтологии после её автоматического доопределения из журналов имитации иерархической системы решения задачи.

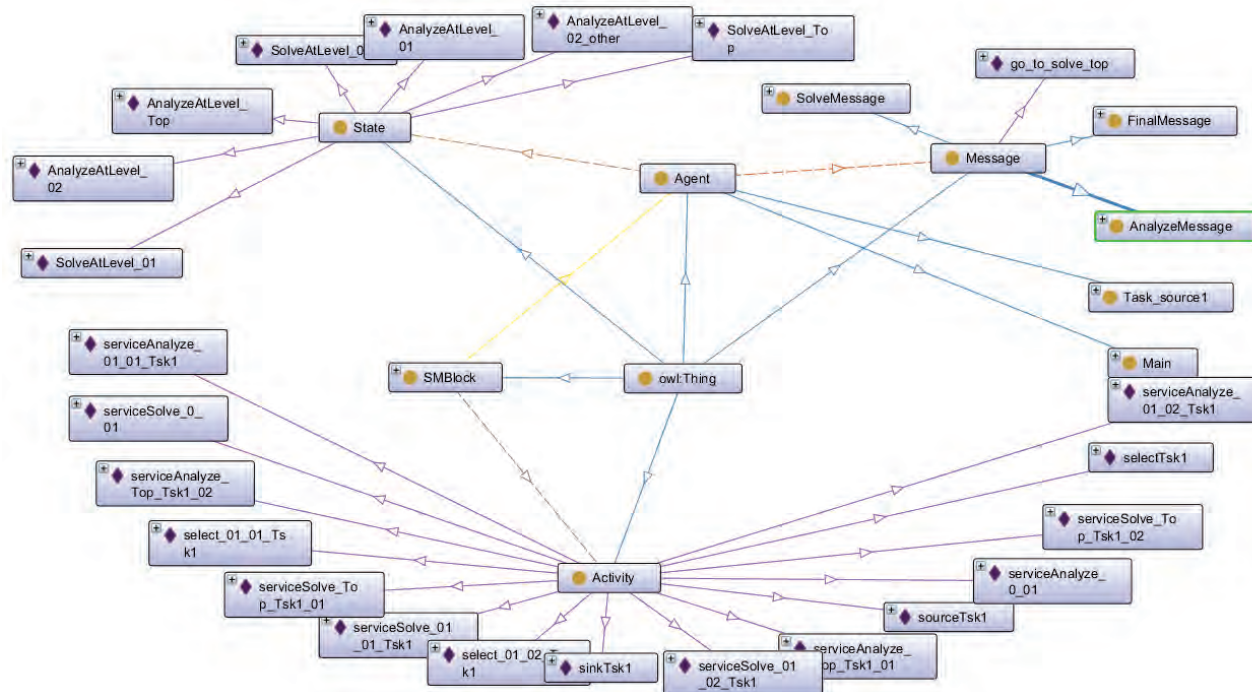


Рисунок 5 – Состояние онтологии событийных данных после её автоматического доопределения

3.4 Вычислительный эксперимент формирования журнала событий

Формирование журнала событий на основе журналов ИМ иерархической системы обработки задач иллюстрирует пример, который представлен на рисунке 4. Был выполнен прогон ИМ в среде *AnyLogic PLE*. Таблица 1 содержит подмножество строк журнала обмена сообщениями, соответствующее нескольким вариантам решения задачи. Каждая задача в ИМ представляется в виде агента (см. таблицы 2 и 3). В таблице 1 содержатся данные о получателе сообщений (*recipient*), который относится к определённому классу (*recipient_type*). Известно сообщение (*message*) и время, когда оно было получено агентом (*date*). Следует отметить, что идентификатор агента не изменяется в процессе нахождения решения задачи и является идентификатором экземпляра процесса.

Таблица 1 – Фрагмент журнала обмена сообщениями

Message	Recipient	Recipient_Type	Sender	Date
back_to_level_01	<population>[7] : 3477	Task_source1	root	08.02.2023 0:05:37
finish_processing	<population>[7] : 3477	Task_source1	root	08.02.2023 0:07:30
next_level_02	<population>[7] : 3478	Task_source1	root	08.02.2023 0:14:29
back_to_level_02	<population>[7] : 3478	Task_source1	root	08.02.2023 0:21:26
next_level_02	<population>[8] : 3479	Task_source1	root	08.02.2023 0:21:56
back_to_level_01	<population>[9] : 3480	Task_source1	root	08.02.2023 0:24:30
back_to_level_01	<population>[7] : 3478	Task_source1	root	08.02.2023 0:26:02
finish_processing	<population>[9] : 3480	Task_source1	root	08.02.2023 0:26:12

Таблица 2 – Фрагмент журнала состояний агентов

Agent_Type	Agent	State	Entry_Date	Exit_Date
Task_source1	<population>[10] : 3481	AnalyzeAtLevel_01	08.02.2023 0:25	08.02.2023 0:29
Task_source1	<population>[10] : 3481	AnalyzeAtLevel_02	08.02.2023 0:29	08.02.2023 3:15
Task_source1	<population>[10] : 3487	AnalyzeAtLevel_01	08.02.2023 1:07	08.02.2023 1:10
Task_source1	<population>[10] : 3487	AnalyzeAtLevel_02	08.02.2023 1:10	08.02.2023 1:19
Task_source1	<population>[10] : 3487	AnalyzeAtLevel_Top	08.02.2023 1:19	08.02.2023 1:24
Task_source1	<population>[10] : 3487	SolveAtLevel_Top	08.02.2023 1:24	08.02.2023 1:26

Таблица 3 – Фрагмент журнала прохождения агентов по ИМ

Agent_Type	Agent	Block_Type	Block	Entry_Date
Task_source1	<population>[7] : 3477	Source	sourceTsk1	08.02.2023 0:01
Task_source1	<population>[7] : 3477	Service	serviceAnalyze_0_01	08.02.2023 0:01
Task_source1	<population>[7] : 3477	SelectOutput5	selectTsk1	08.02.2023 0:05
Task_source1	<population>[7] : 3477	Service	serviceSolve_0_01	08.02.2023 0:05
Task_source1	<population>[7] : 3477	Sink	sinkTsk1	08.02.2023 0:07

Фрагмент созданного журнала событий, сформированного из представленных журналов с учётом ограничений, которые зафиксированы в онтологии, приведён в таблице 4. Здесь атрибуты *Case_id*, *Activity*, *Timestamp* – обязательные компоненты журнала событий. Кроме того, могут присутствовать дополнительные атрибуты (*Difficulty_level*), характеризующие экземпляр в целом или отдельное действие.

Таблица 4 – Фрагмент созданного журнала событий

Case_id	Activity	Timestamp	Difficulty_level	...
<population>[10] : 3487	AnalyzeAtLevel_01	08.02.2023 1:07	0.85	
	AnalyzeAtLevel_02	08.02.2023 1:10	0.85	
	AnalyzeAtLevel_Top	08.02.2023 1:19	0.85	
	SolveAtLevel_Top	08.02.2023 1:24	0.85	

Заключение

Предложен онтологический подход к генерации событийных данных на основе имитационного моделирования так, как они представляются в ИАП. Новизна полученных результатов заключается в применении онтологического анализа в переходе от контекста ИМ к контексту ИАП.

Рассмотрен способ применения данного подхода в моделировании иерархической системы решения задач. Показана архитектура программной системы генерации событийных данных, содержимое онтологического ресурса и генерируемый фрагмент журнала событийных данных применительно к решаемой задаче.

СПИСОК ИСТОЧНИКОВ

- [1] **W. van der Aalst.** Process Mining – Data Science in Action. Second Edition. Springer. 2016.
- [2] **Van der Aalst, W., Adriansyah, A., De Medeiros, A.K.A., Arcieri, F., Baier, T., Blickle, T., Bose, J.C., Van Den Brand, P., Brandtjen, R., Buijs, J., et al.:** Process mining manifesto. In: International Conference on Business Process Management. Springer (2011). P.169-194.
- [3] **Загоруйко Н.Г., Налетов А.М., Соколова А.А., Чурикова В.А.** Формирование базы лексических функций и других отношений для онтологии предметной области // Труды международной конференции Диалог-2004. М.: Наука, 2004. С.202–204.
- [4] **Гуськов Г.Ю., Наместников А.М., Романов А.А., Филиппов А.А.** Формирование базы знаний для поддержки процесса архитектурного проектирования программных систем // Онтология проектирования. 2021. Т.11, №2(40). С.154-169. DOI: 10.18287/2223-9537-2021-11-2-154-169.
- [5] **David Beckett.** The Design and Implementation of the Redland RDF Application Framework. In Proceedings of Semantic Web Workshop of the 10th International World Wide Web Conference, Hong-Kong, China, May 2001.
- [6] **Bertails A., Arenas M., Prud'hommeaux E., Sequeda J.,** Editors. A Direct Mapping of Relational Data to RDF – <http://www.w3.org/TR/rdb-direct-mapping/>
- [7] **D. Brickley and R.V. Guha.** Resource Description Framework (RDF) Schema Specification 1.0. Candidate recommendation, World Wide Web Consortium, March 2000. See <http://www.w3.org/TR/2000/CR-rdf-schema-20000327>.
- [8] **Patrick Hayes.** RDF Model Theory. Working draft, World Wide Web Consortium, September 2001. See <http://www.w3.org/TR/rdf-mt/>.

Сведения об авторе

Наместников Алексей Михайлович, 1974 г. рождения. Окончил Ульяновский государственный технический университет (УлГТУ) в 1996 г., д.т.н. (2018). Профессор кафедры «Информационные системы» УлГТУ. В списке научных трудов более 100 работ в области САПР и прикладных систем искусственного интеллекта. Author ID (RSCI): 392690; Author ID (Scopus): 9277806100. am.namestnikov@gmail.com.



Поступила в редакцию 10.04.2023, после рецензирования 19.05.2023. Принята к публикации 25.05.2023.



Scientific article

DOI: 10.18287/2223-9537-2023-13-2-243-253

Application of the ontological approach in the problem of event data generation using simulation models

© 2023, A.M. Namestnikov

Ulyanovsk State Technical University, Ulyanovsk, Russia

Abstract

The article describes an application of the ontological approach to solving the problem of generating event data coming from the logs of simulation experiments. Currently, as part of the scientific direction Intelligent analysis of processes, methods and algorithms are being developed that allow solving machine learning problems in relation to event data. Simulation modeling can play an important role in the formation of training samples. However the experimental results of simulation in the form of logs of a certain structure must be brought to the form of event logs as they are understood in the intellectual analysis of processes. This paper provides a statement of the problem for forming an ontological resource that makes it possible to generate an event log based on the results of simulation experiments with a discrete-event model in which requests for processing are presented in the form of agents. A formal description of the domain ontology and an algorithm for its redefinition based on the log data of the simulation model are given. As an object of simulation the paper proposes to consider a hierarchical decision-making system, which receives tasks of varying complexity. The level of complexity of tasks is used for choosing the level of hierarchy at which this task needs to be

solved. The architecture of the developed ontological system is given, as well as the structure of concepts with the corresponding semantic relations and sets of instances.

Keywords: *ontology, intellectual process analysis, process mining, event data, simulation model, agent.*

For citation: *Namestnikov AM.* Application of the ontological approach in the problem of event data generation using simulation models [In Russian]. *Ontology of designing.* 2023; 13(2): 243-253. DOI:10.18287/2223-9537-2023-13-2-243-253.

Financial Support: This work was supported Ministry of Education and Science of Russia in framework of project № 075-00233-20-05 from 03.11.2020 «Research of intelligent predictive multimodal analysis of big data, and the extraction of knowledge from different sources».

Conflict of interest: The author declares no conflict of interest.

List of figures and tables

Figure 1 – The place of the event data ontology in the process of generating an event log

Figure 2 – The event data ontology structure

Figure 3 – The architecture for the event data generation system

Figure 4 – An illustrative example of a hierarchical task processing system

Figure 5 – The state of the event data ontology after its automatic redefinition

Table 1 – A messaging log fragment

Table 2 – An agent state log fragment

Table 3 – A Fragment of the log of passing agents through the simulation model

Table 4 – A Fragment of the generated event log

References

- [1] **W. van der Aalst.** Process Mining – Data Science in Action. Second Edition. Springer. 2016.
- [2] **Van der Aalst, W., Adriansyah A, De Medeiros AKA, Arcieri F, Baier T, Blickle T, Bose JC, Van Den Brand P, Brandtjen R, Buijs J,** et al.: Process mining manifesto. In: International Conference on Business Process Management. p.169-194. Springer (2011).
- [3] **Zagoroyko NG, Naletov AM, Sokolova AA, Churikova VA.** A Formation of the lexical functions base and other relations for the ontology of the subject area [In Russian]. Proceedings of the Conference “Dialog-2004”. M.: Science, 2004. 202–204 p.
- [4] **Guskov GY, Namestnikov AM, Romanov AA, Filippov AA.** Formation of a knowledge base to support the architecting process of software systems [In Russian]. *Ontology of designing.* 2021; 11(2): 154-169. DOI: 10.18287/2223- 9537-2021-11-2-154-169.
- [5] **David Beckett.** The Design and Implementation of the Redland RDF Application Framework. In Proceedings of Semantic Web Workshop of the 10th International World Wide Web Conference, Hong-Kong, China, May 2001.
- [6] **Bertails A, Arenas M, Prud'hommeaux E, Sequeda J.** Editors. A Direct Mapping of Relational Data to RDF – <http://www.w3.org/TR/rdb-direct-mapping/>
- [7] **Brickley D, Guha RV.** Resource Description Framework (RDF) Schema Specification 1.0. Candidate recommendation, World Wide Web Consortium, March 2000. See <http://www.w3.org/TR/2000/CR-rdf-schema-20000327>.
- [8] Patrick Hayes. RDF Model Theory. Working draft, World Wide Web Consortium, September 2001. See <http://www.w3.org/TR/rdf-mt/>.

About the author

Alexey Mikhailovich Namestnikov (b. 1974) graduated from the Ulyanovsk State Technical University (Ulyanovsk, Russian Federation) in 1996, Doctor of science (2018). He is a professor at the Department of Information systems. He is a co-author of about 100 scientific articles and abstracts in the field of CAD and applied AI. Author ID (RSCI): 392690; Author ID (Scopus): 9277806100. am.namestnikov@gmail.com. ✉.

Received April 10, 2023. Revised May 19, 2023. Accepted May 25, 2023.