SOLID-M: An Ontology-Aware Quality Framework for Conceptual Models Discovered from Event Data

Andrei Tour[a,*], Artem Polyvyanyy[a], Anna Kalenkova[b]

[a] *The University of Melbourne, Parkville, 3010, VIC, Australia*
[b] *The University of Adelaide, North Terrace Campus, Adelaide, 5000, SA, Australia*

Abstract

In Process Mining (PM), "high-level" conceptual models of business processes, in the form of directly-follows graphs, Petri nets, and finite-state automata, are discovered from "low-level" event data recorded by information systems. The quality of the discovered models is usually assessed by measures that depend on assumptions made by discovery algorithms; for example, they often assume that sequences of activities recorded in the event data do not interfere. Models produced by recent discovery algorithms consider domain knowledge and relax these assumptions, making traditional PM measures less suitable for evaluating their quality. This paper proposes an ontology-aware framework, called SOLID-M, for analyzing the quality of conceptual models discovered from event data generated by systems. SOLID-M relies on domain knowledge and provides guidelines for introducing quality measures for models constructed by process discovery algorithms that go beyond the traditional PM assumptions. In addition, the paper describes an instantiation of the framework for assessing the quality of Multi-Agent System models discovered using Agent System Mining techniques, hence addressing a growing demand for data-driven analysis of business processes emerging in interactions of human and artificial intelligence agents.

Keywords: conceptual model quality, ontology, process discovery

*Corresponding author.
Email addresses: atour@student.unimelb.edu.au (Andrei Tour), artem.polyvyanyy@unimelb.edu.au (Artem Polyvyanyy), anna.kalenkova@adelaide.edu.au (Anna Kalenkova)

Postprint, November 2025

1. Introduction

The aim of Process Mining (PM) is to discover, analyze, and repair conceptual models of business processes using event data extracted from information systems [1]. PM discovery algorithms synthesize process models in "high-level" modeling languages (*e.g.*, Petri nets and BPMN) from "low-level" behavior specifications (*e.g.*, event logs) [2].

An important success measure for business process modeling projects is the quality of produced process models [3]. The PM community often reasons about the quality of discovered models using the "four quality dimensions in process discovery" framework (the 4QD framework) [4]. The four dimensions are replay fitness, precision, generalization, and simplicity. The 4QD framework identifies recall, precision, and model size as common measures for the replay fitness, precision, and simplicity dimensions, respectively. An intuitive set-theoretical framework for quantifying the fitness, precision, and generalization dimensions is proposed by Buijs, van Dongen, and van der Aalst (the BDA framework) [5]. The BDA framework defines several precision, recall, and generalization measures by relating the behaviors captured in the model, the log from which the model was discovered, and the system that generated the log. The behaviors are given as sets of traces, where a trace is a sequence of observed events, each referring to an activity executed in some instance of a business process (also known as a case).

Process discovery algorithms based on the traditional process modeling paradigms and the corresponding model quality measures share the same assumptions, *e.g.*, a single control flow for each case and a single case for each event. These assumptions are relaxed in new PM types that extend or mix the existing paradigms, *e.g.*, Object-Centric Process Mining (OCPM) [2], Agent System Mining (ASM) [6], and Queue Mining [7]. This observation motivates the definition of new quality measures and frameworks suitable for evaluating models constructed by new types of discovery algorithms.

We present the SOLID-M framework, an ontology-aware quality framework for the assessment of models discovered from event data. SOLID-M extends and generalizes the existing quality frameworks currently used in PM. Our framework is grounded in the semiotic quality framework (SEQUAL) designed by Lindland, Sindre, and Sølvberg [8] for the quality analysis of conceptual models, and in the data quality framework with ontological foundations based on Bunge's ontology [9, 10]. We extend SEQUAL by adding quality aspects that relate data with models and ontologies using set-

theoretical measures. The key to our approach is the observation that the existing model quality frameworks used in PM (4QD and BDA) are limited to behavior-only models. To address this limitation, we rely on SEQUAL. Like BDA, SEQUAL follows a set-theoretical approach. However, elements of the sets in SEQUAL are specification *statements* representing behavioral or structural concepts, not limited to traces only.

With the rapid growth of Artificial Intelligence (AI), human-AI interactions within business processes are becoming increasingly widespread [11]. Consequently, the importance of Multi-Agent System (MAS) discovery is growing, necessitating new methods for assessing the quality of MAS models. In this context, we instantiate the SOLID-M framework to evaluate the quality of MAS models discovered using Agent Miner [12] based on event data generated by business processes.

The next section describes the methodology used to conduct the research presented in this paper. Section 3 explains the context and background information required for understanding the ideas articulated in this paper. Section 4 provides an overview of the existing related model quality frameworks. Section 5 introduces an example MAS model discovered by Agent Miner. Section 6 presents the SOLID-M framework and exemplifies its instantiation using an agent-based ontology and measures for assessing the quality of MAS models discovered from event data. Section 7 discusses the contributions of SOLID-M in the light of existing frameworks, sketches an alternative instantiation of SOLID-M for OCPM, and acknowledges limitations of the framework. Finally, Section 8 states concluding remarks.

2. Methodology

This paper presents a Design Science Research (DSR) artifact [13], the SOLID-M quality framework designed for the analysis of conceptual models discovered from event data. This artifact was developed using the DSR Methodology (DSRM) by performing the following activities of the DRSM process [14]:

- **Problem identification and motivation.** The measures of the quality of process models produced by traditional process discovery algorithms are not suitable for assessing the quality of models discovered by the new types of process discovery techniques. The reason for this is that the assumptions underpinning the traditional quality measures do

not hold for these process discovery types. For example, the assumptions of a single process instance for each event and a single control flow for each process instance are relaxed in OCPM and ASM, respectively. Consequently, the motivation for this work is to propose ways to evaluate models constructed by the new types of discovery algorithms.

- **Define the objectives for a solution.** To address the identified problem, we designed the SOLID-M framework to meet the following requirements: (a) it must enable assessment of the quality of conceptual models containing both behavioral and structural elements representing organizations and their processes; (b) it must enable explicit specification of the assumptions underpinning the techniques and measures used for creation and quality assessment of the conceptual models; (c) it must enable assessment of the quality of conceptual models discovered from data recorded by information systems.

- **Design and development.** We designed our quality framework, SOLID-M, by combining and extending the existing frameworks and models described in Section 4. The SOLID-M framework is defined in Section 6. Its main components are the modeling artifact sets and quality aspects integrated into the aspect graph depicted in Figure 9. We designed SOLID-M incrementally. Our starting point was the traditional PM quality frameworks (4QD and BDA) that define the quality measures for behavior-only process models. To meet requirement (a), we used SEQUAL [8, 15], a generic model quality framework supporting conceptual models containing not only behavioral but also structural elements. The original SEQUAL framework does not satisfy requirements (b) and (c) because it does not deal with explicit identification of assumptions associated with quality aspects and does not define any quality aspects relating models to data. To overcome this limitation, we extended SEQUAL with ontology-related quality aspects based on the ontology-aware representational model of information systems [16], which proposes the use of ontologies with explicitly defined real world domain concepts. To satisfy requirement (c), we conceptualized a model as a result of the data interpretation process described in the ontology-aware data quality framework [10].

- **Demonstration and Evaluation.** Our example instantiation of SOLID-M demonstrates the use of the framework for analyzing the

quality of agent-based Petri net models discovered from event data by the Agent Miner algorithm [12]. We evaluate SOLID-M by relating the results of the demonstrated framework instantiation example with the solution objectives stated as three requirements. Requirement (a) is met in the example SOLID-M instantiation by including behavioral (*e.g.*, Event) and structural (*e.g.*, Agent and Interface) elements in the example Model set (see Figure 7 and Table 7). Requirement (b) is satisfied by explicit specification of the agent-based concepts in the Ontology set (see Figure 10 and Table 2). Requirement (c) is fulfilled specifying the characteristics (*e.g.*, Model-Data Structural Completeness) and the corresponding measure (*e.g.*, Model-Data Structure Recall) for assessing the Interpretational Model Quality aspect.

- **Communication.** We communicate the artifact we designed, the SOLID-M framework, in the paper at hand.

3. Background

This section explains the context and key concepts underpinning the SOLID-M framework, namely *ontologies* [17], *process discovery* [4], and *systems thinking* [18].

3.1. Ontologies and Description Logic

The term *ontology* is understood differently in different communities. We adopt a computational view frequently used in computer science. In this context, a (computational) *ontology* is a formal, explicit specification of a shared conceptualization [17], where a *conceptualization* is an abstract representation of the world or some domain. Any formally represented knowledge relies on a conceptualization that identifies objects, concepts, and the relations between them, which are assumed to exist in an area of interest. To be an ontology, a conceptualization must be explicitly specified in some formal language, with its meaning shared among its users.

In this paper, the Web Ontology Language (OWL) is used to specify conceptualizations. More precisely, we use the OWL-DL sub-language of OWL that has well-defined decidable semantics mapped to Description Logic (DL) [19]. DL is a subset of First-Order Logic with decidable reasoning problems. For each OWL-DL language construct, there is an equivalent DL construct, which makes OWL-DL a popular language for representing knowledge in the semantic web and knowledge management software [20].

The name "Description Logic" is motivated by the fact that, on the one hand, the important notions of the domain are described by concept descriptions, *i.e.*, expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL. On the other hand, DLs differ from their predecessors, such as semantic networks and frames, in that they are equipped with a formal, logic-based semantics.

DLs have three types of elements: individuals denoting things in the world, concepts denoting sets of individuals, and roles denoting relations between the individuals. A knowledge base represented using DLs has two components: a *TBox* and an *ABox*. The TBox is a set of *terminological axioms*, DL expressions that define the concepts and roles of the domain of interest. The axioms contain the terminology of the knowledge base. The ABox is a set of *assertions* about expressing the facts about the individuals in the domain of interest. The assertions are expressed using the terminology defined in the TBox.

3.2. Process Mining

Process Mining (PM) is a research discipline that studies techniques for extracting knowledge about real world organizations from event logs recorded by information systems [4]. The extracted knowledge is represented in conceptual models of business processes using business process modeling languages (*e.g.*, Petri net, BPMN, and UML activity diagrams). PM conceptualizes a business process as a sequence of events that happen to achieve objectives in the context of an organization. Each event refers to an activity, a well-defined unit of business behavior. The assumption is that repetitive patterns of activity sequences exist in organizations and that the event data captured from these organizations accurately represent the patterns.

The same event data may be interpreted from different PM perspectives, each based on different goals and assumptions. The control flow perspective, the main PM perspective, is grounded in the *one case notion* assumption. According to this assumption, the purpose of a business process is to complete one case (*e.g.*, a customer request), and every event within this case is related to this case only.

Object-Centric Process Mining (OCPM) relaxes the one case notion assumption. It allows an event to be associated with multiple objects of different types [21]. The way the quality of OCPM models is measured depends on

the object types and their relations. The use of process quality frameworks relying on the single case notion is problematic for OCPM.

The Agent System Mining (ASM) perspective assumes that an end-to-end (global) business process is induced by interactions of multiple autonomous agents performing their own (local) processes [12]. This is different from a traditional PM view of centrally controlled resources executing a well-defined end-to-end flow of activities.

3.3. Behavior and Structure of Systems

This paper discusses the quality of conceptual models that describe real world systems in terms of their behavior and structure. We define a *system* as a whole consisting of social and technical components that interact with each other to serve some purpose. This definition is similar to the notion of a socio-technical system discussed by Wu et al. [22], which is used for holistic modeling of artificial and natural phenomena in multiple problem domains. For example, an organization can be conceptualized as a system that contains people, computers, and other equipment as components, all interacting to perform business processes.

A system model typically addresses two questions: "What does the system contain?" and "What does the system do?" To answer these questions, a model needs to describe the structure and behavior of the system, respectively. Event and process are examples of concepts used to model system behavior. Relationships among system components constitute the system structure. Examples of system structure concepts are components, resources, objects, actors, agents, and interfaces. ArchiMate is an example of an enterprise architecture language that explicitly groups its concepts into behavior and structure categories [23]. In this framework, active structure elements (*i.e.*, actors) generate changes in passive structure elements (*i.e.*, objects) by executing behavioral elements (*i.e.*, processes).

The key concepts in PM are behavioral. An *event log* is a collection of traces, where a *trace* is a sequence of events describing the execution of a single *process instance*. The focus is on what happens (behavior), not on who and what makes it happen (structure). Nonetheless, some PM approaches discover models that incorporate both behavioral and structural elements. For example, OCPM discovers objects that represent the passive structure of a system, whereas ASM discovers agents that capture its active structure.

4. Related Frameworks

This section provides an overview of the frameworks that influenced the design of SOLID-M.

4.1. Four Quality Dimensions in Process Discovery

Figure 1 shows the four quality dimensions (4QD) framework for characterizing the quality of process models discovered from event logs [4]. The four dimensions are fitness, precision, generalization, and simplicity.

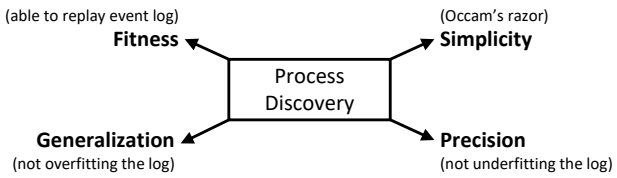


Figure 1: Balancing the four quality dimensions [4].

Fitness and precision relate event logs and the process models discovered from these logs. Fitness characterizes the ability of the process model to represent the behavior captured in the event log. A model with perfect fitness represents all the behavior contained in the corresponding event log. Precision indicates to what extent the model underfits the log, *e.g.*, it shows how much of the behavior described in the model can be found in the log. The model has perfect precision if all its behavior can be found in the log. The generalization dimension characterizes the ability of the process model to represent the actual behavior of the observed system, even if some behavior of this system is not captured in the event log. Finally, simplicity refers to the amount of effort required to understand the discovered model.

The four dimensions are defined in terms of behavior modeled as process traces. The behavior-only focus limits the use of this framework for process discovery methods where, in addition to behavior, the discovered models contain structural entities such as objects, artifacts, or agents.

4.2. Behavior Sets for Precision and Recall

Precision and fitness (also known as recall), the two most commonly used dimensions from the 4QD framework, are further articulated in the framework by Buijs, van Dongen, and van der Aalst, the (BDA) framework [5],

through the lens of three overlapping sets of behavior in terms of process traces: behavior of a real world system, behavior captured in an event log, and behavior represented in a process model. A Venn diagram for the three behavior sets is depicted in Figure 2.

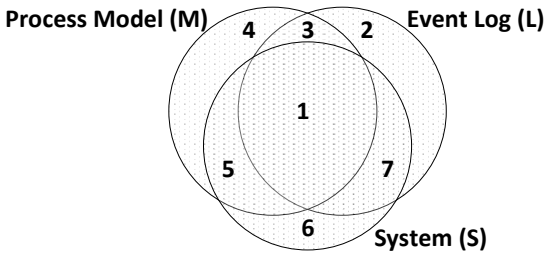


Figure 2: Venn diagram showing how the behavior of the process model (M), event log (L) and system (S) can be disjoint or overlapping [5].

The diagram identifies seven classes of traces. For example, traces in class 1 are the traces of the system recorded in the log and described by the model, whereas the traces in class 2 are the traces that the system does not generate, yet recorded in the log (maybe due to a logging error) and not described by the model (perhaps because they were suppressed by thr discovery algorithm as noise).

The BDA framework defines model-log precision and recall, log-system precision and recall, and model-system precision and recall quality measures for process models discovered from event data based on the relationship between the corresponding behavioral sets. The model-system recall measure is referred to as generalization. The behavior sets defined in BDA do not include structural entities, such as objects or agents. This prevents the direct use of this framework for evaluating models discovered by PM algorithms that relax the traditional PM assumptions.

4.3. Statement Sets for Conceptual Model Quality

The semiotic quality framework (SEQUAL) for discussing the quality of conceptual models was proposed by Lindland, Sindre, and Sølvberg [8]. SEQUAL is based on Morris' semiotics theory of signs [24]. This framework views a conceptual model as a set of specification statements. No assumptions or constraints are imposed on the statements. First, SEQUAL defines four

sets of statements: Model, Domain, Language, and Audience Interpretation. A Model is a set of statements made in a conceptual model. A Domain is a set of true statements about the domain of reality being modeled. A Language is a set of statements that are correct according to the grammar of the modeling language of the model. An Audience Interpretation is a set of statements recognized by the audience in the model. Then, three aspects of the model quality are defined as relations between the four sets: syntactic quality relates the Model to Language, semantic quality relates the Model to Domain, and pragmatic quality relates the Model to Audience Interpretation. Figure 3 visualizes the SEQUAL framework as a graph with four nodes corresponding to the sets and three edges corresponding to the quality aspects.

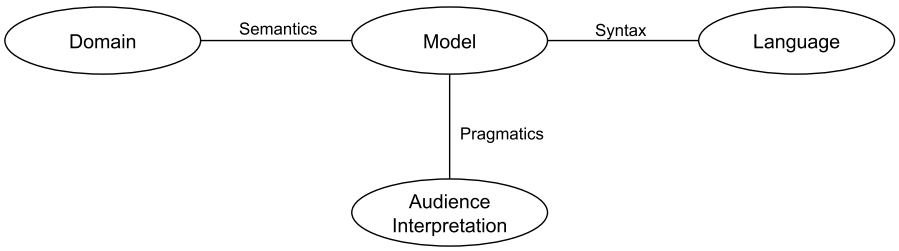


Figure 3: The SEQUAL framework [8].

SEQUAL was extended for assessing the quality of process models in an organizational context [15]. This extension introduces a dynamic perspective where an organization changes a problem domain by performing business activities specified in process models. It creates the need to update the process models. Neither the initial SEQUAL nor its process model extension considers scenarios where process models are created or updated based on data. This gap needs to be addressed when using this framework in the PM context.

4.4. Iterative Ontology-Based Representation of Real World

Wand and Weber formulated the representational model of an information system [16]. The key working premise of this model is that an information system is someone's representation of a real world system. Information system analysis, design, and implementation are viewed as an iterative process of creating representations of a real world system in the form of data in an information system. These representations are called scripts. The

initial iterations create less formal human-oriented scripts. Each new iteration transforms the previously created script into a more formal and detailed script. The final iterations produce machine-oriented scripts that machines can use to process information. This idea of iterative transformations of representation scripts is visualized in Figure 4.

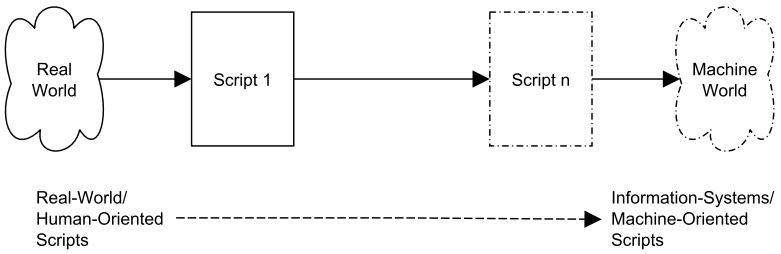


Figure 4: The representational model of an information system [16].

The scripts at different iterations of the representation process are expressed in different languages. These languages have different vocabularies and levels of abstraction and describe invariants of real world structure and behavior. In computer science, an ontology of a domain defines established facts, concepts, and relationships between them for the purpose of transitioning between different domain representations [25]. In this approach, an ontology developed by Bunge [9] is used to identify elements of a real world system preserved in transitions between the iterations so that the produced scripts can be linked and compared.

4.5. Data Quality through Ontology-Based Representation and Interpretation

Wand and Wang proposed a data quality framework that extended the representational model based on Bunge's ontology [10]. We call it the Bunge-Wand-Wang (BWW) data quality framework (see Figure 5).

BWW includes the representation process that produces an information system containing data about a real world system. The framework also introduces the direct observation and interpretation processes. In the former, the user constructs a view of the system by directly observing it. In the latter, the user's view of the system is inferred from data in an information system. The BWW framework defines a data deficiency as a difference between the direct and inferred user views. Bunge's ontology [9] is used to compare real world entities present in the two views, identify data deficiencies, and group the deficiencies into several categories and quality dimensions.

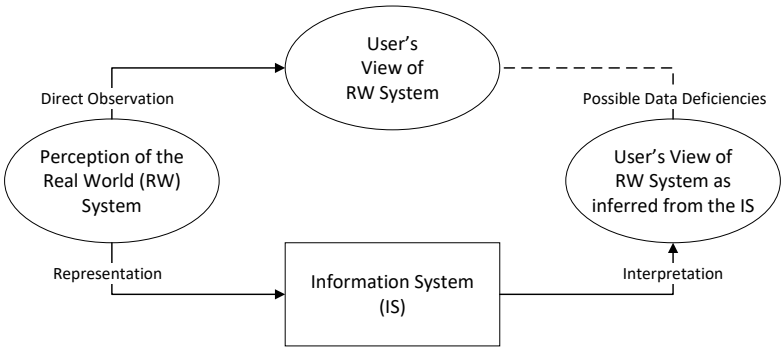


Figure 5: The BWW data quality framework [10].

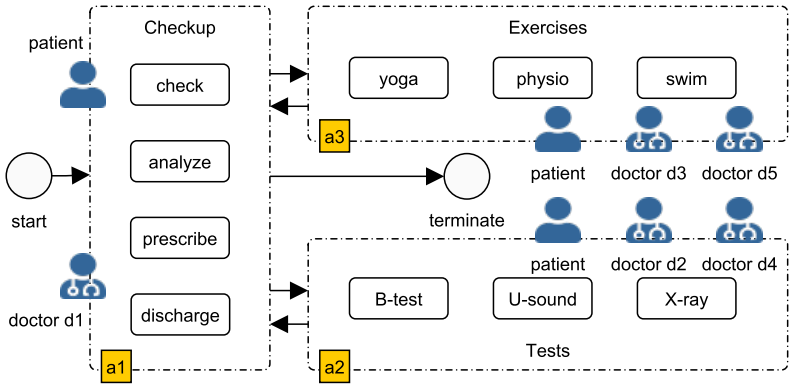


Figure 6: A schematic visualization of the example health surveillance system [12].

5. Motivating Example

This section introduces an example agent-based model of a hypothetical health surveillance process, discovered from event data by Agent Miner, a non-traditional process discovery algorithm grounded in the Agent-Based Modeling paradigm [12]. We use this example in the remainder of this paper to illustrate how the SOLID-M framework can be instantiated to analyze the quality of discovered agent-based models.

Figure 6 shows a schematic visualization of the hypothetical health surveillance system. We observe this system from an agent-based point of view as a distributed process emerging from interactions of five agents within a Multi-Agent System (MAS). The five agents are doctors d1 to d5 belonging to agent types a1, a2, and a3. The agents perform patient diagnostics and preventive

physical activities. Doctor d1 (agent type a1) performs four activities in the 'Checkup' group. Doctors d2 and d4 (agent type a2) perform three activities in the 'Tests' group. Doctors d3 and d5 (agent type a3) perform three activities in the 'Exercises' group . In addition to performing their activities, the agents interact, as shown in Figure 6.

The end-to-end patient health surveillance process emerges as a result of the agent interactions and activities. A patient case is an instance of this end-to-end process representing one episode of checking health for one patient. A case trace is a sequence of events associated with the same patient case where each event is an instance of an activity performed by a corresponding agent type. A typical case trace can be represented as the following sequence of (agent type, activity) pairs: $\langle(\text{a1}, \text{check}), (\text{a1}, \text{analyze}), (\text{a1}, \text{prescribe}), (\text{a2}, \text{B-test}), (\text{a2}, \text{U-sound}), (\text{a1}, \text{check}), (\text{a1}, \text{analyze}), (\text{a1}, \text{prescribe}), (\text{a3}, \text{yoga}), (\text{a3}, \text{physio}), (\text{a1}, \text{check}), (\text{a1}, \text{discharge})\rangle$.

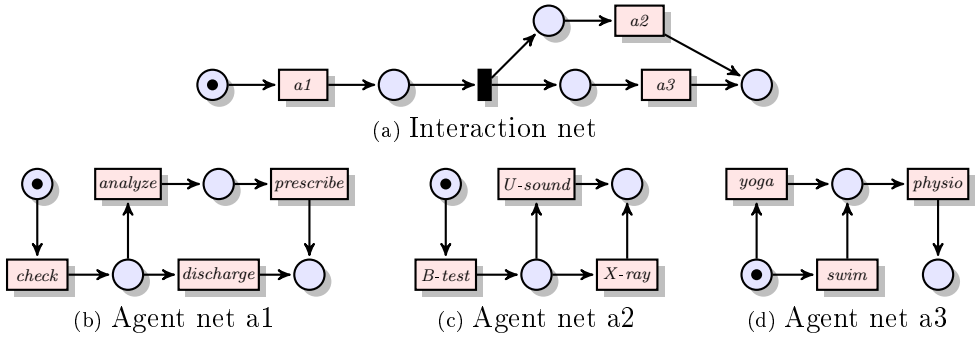
Event data representing multiple case traces produced by the example MAS over a period of time is captured in the example event log. A fragment of this log is presented in Table 1. Each line in the event log represents several facts about structure and behavior elements related to one event produced by the MAS. For example, the first row in the table provides facts about event e1 that happened in case c0 and was produced by agent type a1 performing activity 'check' at time '0:00'.

Table 1: A fragment of the example event log.

Event	Case	Activity	Agent	Time
e1	c0	check	a1	0:00
e2	c0	discharge	a1	1:01
e3	c1	check	a1	2:02
e4	c1	analyze	a1	3:03
e5	c1	prescribe	a1	4:04
e6	c1	B-test	a2	5:05
e7	c1	X-ray	a2	6:06
e8	c1	swim	a3	7:07
e9	c1	physio	a3	8:08
e10	c1	check	a1	9:09
e11	c1	discharge	a1	10:10

The Agent Miner algorithm uses the example event log to discover the MAS model comprising four Petri nets: one interaction net and three agent nets. These four nets are shown in Figure 7. The interaction net contains three labeled transitions (a1, a2, and a3) that correspond to the agent types and two paths between transitions (from a1 to a2 and from a1 to a3) that correspond to the interfaces between the agent types. Each agent net represents a local process executed by the respective agent type. For example, the

four labeled transitions of agent net a1 correspond to the activities executed by agent type a1.



(a) Interaction net

(b) Agent net a1 (c) Agent net a2 (d) Agent net a3

Figure 7: A MAS model comprising one interaction net and three agent nets.

These nets can be composed into the MAS net by refining the labeled transitions in the interaction net with the corresponding agent nets [12].

We aim to answer the following questions about the quality of the MAS model discovered by Agent Miner:

a) How well does the model represent the local behavior of each agent type?
b) How well does the model represent the structural elements of the MAS (agent types and interfaces)?
c) How easily can a model user recognize the agent types and interfaces represented in the model?

The traditional quality measures used in the process mining community (*e.g.*, fitness and precision as defined in the BDA framework) are inadequate for addressing these questions, as they are not grounded in an agent-based ontology. These measures are not formulated in terms of agents, their interfaces, or their local processes. In the following section, we introduce a generic quality framework that overcomes this limitation.

6. The SOLID-M Framework

This section presents the SOLID-M framework, an ontology-aware quality framework for conceptual models mined from event data. Section 6.1 introduces the key concepts and components of the framework, and outlines how

they fit together. Next, Section 6.2 describes the modeling process assumed by the framework. Finally, Section 6.3 and Section 6.4 discuss the modeling artifacts and quality aspects of the SOLID-M framework, respectively.

We use the health surveillance example introduced in Section 5 to show how the framework elements can be instantiated to assess the quality of agent-based models discovered from event data. The full specifications of all the example artifacts developed in this example framework instantiation are publicly available [26].

6.1. Overview

SOLID-M addresses the need of the PM community to evaluate models discovered using non-traditional methods, going beyond the single-case-notion, single-control-flow view of the world. The cornerstone of the SOLID-M framework is the idea of a model as a set of specification statements describing system behavior and structure, where the statements are expressed in some modeling language (*e.g.*, Petri nets) and grounded in some ontology (*e.g.*, an agent-based ontology). Guided by this idea, our framework defines six types of *modeling artifacts* as sets of statements, namely the System, Ontology, Language, Interpretation, Data, and Model sets; the first letters of the six types of artifacts form the framework name, SOLID-M. These artifacts are produced by *modeling activities* of the *ontology-aware modeling process* presented in Figure 8. This modeling process provides a context for the main component of the SOLID-M framework, the *quality aspects graph* depicted in Figure 9.

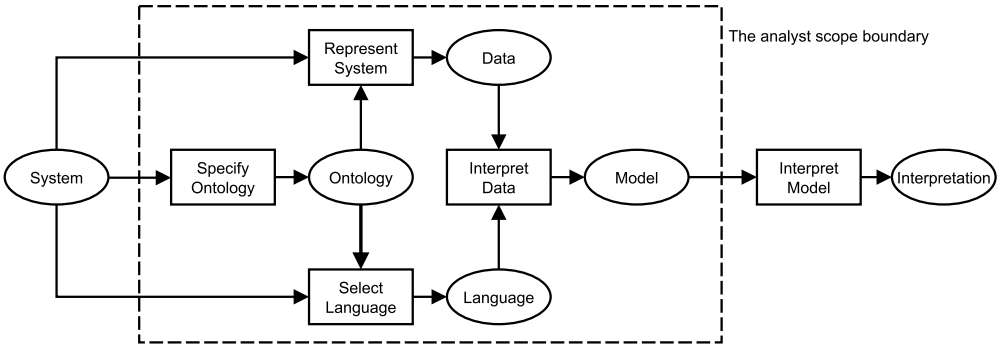


Figure 8: The SOLID Model quality framework: the ontology-aware modeling process.

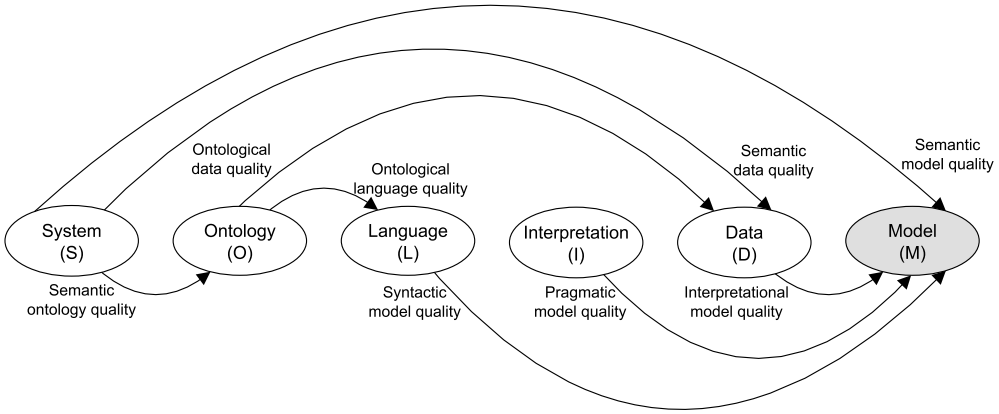


Figure 9: The SOLID Model quality framework: the quality aspects graph.

The SOLID-M *ontology-aware modeling process* is a sequence of *modeling activities*. The inputs and outputs of the activities are *modeling artifacts*. In the figure, the activities and artifacts are depicted as rectangles and ovals, respectively. Each activity is linked to its input and output artifacts by inbound and outbound arrows. The process participants are either model authors (the analysts) or model users (the audience). The dashed frame marks the boundary of the analysts' control. All activities and their output artifacts inside the dashed box are produced by the analyst. An analyst creates the Model artifact by performing the Specify Ontology, Select Language, Represent System, and Interpret Data activities. The Model produced by the analyst is interpreted by the audience in the Interpret Model activity to gain a better understanding of the modeled real world System artifact they are interested in. The Interpretation artifact represents the understanding of the System achieved by the audience as a result of performing the Interpret Model activity.

The SOLID-M *quality aspects graph* (Figure 9) is a directed graph, where nodes and edges correspond to the SOLID-M *modeling artifacts* and *quality aspects*, respectively. A *quality aspect* is a grouping of *quality characteristics* indicating a dependency between two *modeling artifacts*. Each quality aspect is associated with one source-target pair of related modeling artifacts, where the target artifact depends on the source artifact. A *quality characteristic* is the characteristic of the quality of the corresponding target artifact relative to the corresponding source artifact. For example, the "Interpretational model

quality" aspect is a group of quality characteristics that characterize the Model artifact relative to the Data artifact. This aspect is associated with the (Data, Model) pair, where Model (the target artifact) depends on Data (the source artifact). The quality characteristics grouped under this aspect, like precision and recall, indicate the quality of Model relative to Data.

Multiple *quality measures* can be defined for each SOLID-M quality characteristic. To define these measures explicitly, the source and target artifacts associated with each characteristic are understood as sets of *statements*. A *statement* expresses a fact, a conceptual construct, or an opinion in some language. For example, Model and Data artifacts are viewed as sets of model and data statements, respectively. The content and structure of the statements depend on an ontology selected by the analyst. The statements in all the sets are grounded in the same ontology of choice. This makes it possible to compare statements across different sets and define quality measures based on the set intersections, unions, and differences.

All target artifacts for any quality aspect in Figure 9 are the outputs of the modeling activities performed by the analysts (*i.e.*, the activities inside the dashed box in Figure 8). Thus, the quality characteristics associated with any aspect defined in SOLID-M are used to reason about the quality of the Model, Data, Language, and Ontology artifacts produced or used by the analysts.

The SOLID-M framework cannot be used directly. It should be instantiated for the given types of real world systems and modeling objectives. The framework defines the statement sets and quality aspects and guides the definition of the quality characteristics and measures. The characteristics suggested here as part of the quality aspect descriptions are examples only that are intended to aid a better understanding of the corresponding aspects.

An *instantiation* of the framework starts by choosing the types of real world systems of interest and specifying the ontology that defines the concepts, relations, and constraints constituting the Ontology set. Then, the instantiation defines the quality characteristics and measures grounded in the specified ontology. For example, Figure 6 illustrates a real world system of interest for our example SOLID-M instantiation. In this instantiation, we choose to reason about this system of interest as a MAS of interacting agents. Thus, we define "MAS", "Agent", and related concepts in our example ontology as specified in the UML class diagram depicted in Figure 10.

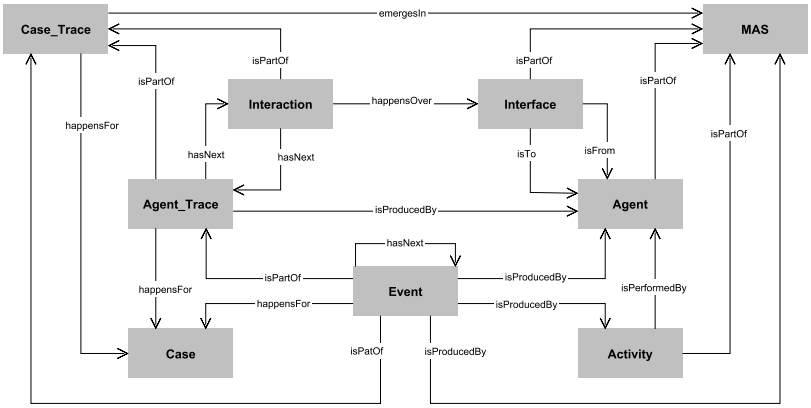


Figure 10: A UML class diagram specifying the example ontology.

6.2. Modeling Process

SOLID-M is based on the assumption that creation of conceptual models from data follows the modeling process depicted in Figure 8. This SOLID-M *modeling process* involves five activities (rectangles) and six artifacts (ovals). Multiple iterations of this process are possible, with each iteration producing an improved modeling artifact.

The "System" oval in Figure 8 is the input to the first step of the process, "Specify Ontology", because it denotes an implicit, informal understanding (not an explicit description in some modeling language) of a real world fragment that the analyst directly observes for the purpose of modeling. The analyst uses this implicit understanding to specify an ontology suitable for describing the observations. Our "System" corresponds to the term "System" in the BDA framework (Section 4.2) and the term "Domain" in the SEQUAL framework (Section 4.3). In our example, Figure 6 informally visualizes the example "System", and the specified agent-based ontology is defined in the UML diagram shown in Figure 10.

The same real world phenomena can be conceptualized using different ontologies [17]. For example, we can conceptualize the same observed behavior as produced by a system of multiple interacting agents (an agent-based ontology) or by a single centralized process (a process-oriented ontology). Therefore, the first step in the SOLID-M modeling process is to specify an ontology. The same ontological concept can be expressed in different modeling languages [16]. For instance, the same process can be described using a Petri net or a BPMN diagram. Hence, in the SOLID-M modeling process,

the ontology specification step is followed by the language selection step.

The "Represent System" and "Select Language" activities can be performed by the analyst in parallel, producing Data and Language artifacts based on direct observations of the system of interest and concepts from the specified ontology. The event log fragment in Table 1 and the language metamodel presented in Figure 11 demonstrate the Data and Language artifacts for our example instantiation of the framework.

The example ontology metamodel and the example language metamodel are specified in two separate diagrams (Figures 10 and 11) because the concepts they present differ in nature. Figure 10 specifies ontological terms that conceptualize entities in the real world, while Figure 11 specifies language concepts that are used to define the language syntax and grammar. In addition, the same ontology may correspond to several modeling languages, so defining the ontological concepts in a separate diagram allows using the same version of Figure 10 across multiple alternative modeling languages, with the one shown in Figure 11 being one example.

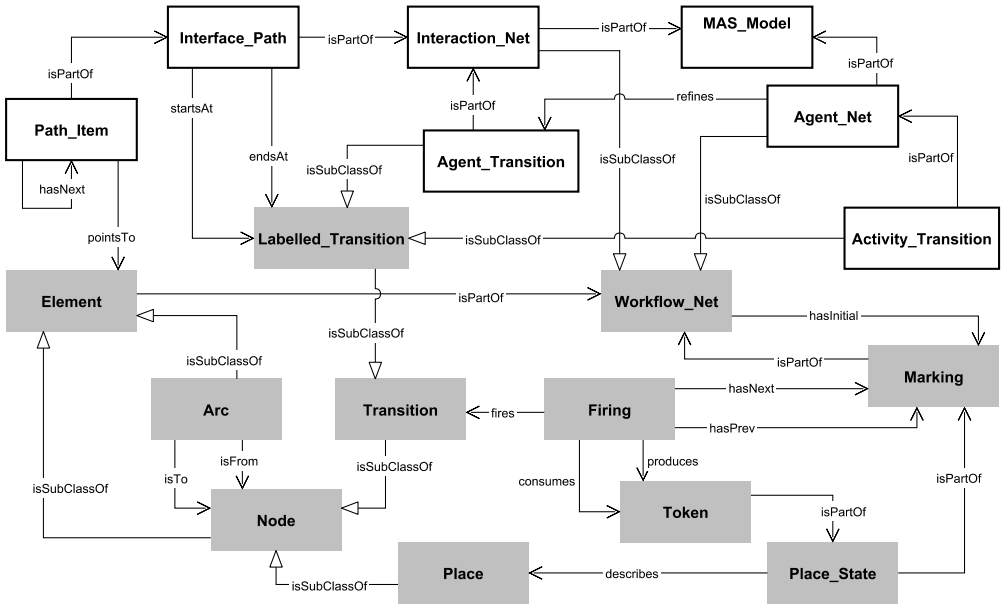


Figure 11: A UML class diagram specifying concepts used in the Language set (white and gray boxes are Agent Miner specific and traditional Petri net concepts, respectively).

The "Interpret Data" activity generates a Model artifact. This activity is

automated by the Agent Miner algorithm [12] in our example. It discovers the example Model artifact, a MAS model containing four Petri nets, as shown in Figure 7. This MAS model is discovered from the example event log (Table 1) and modeled using agent-based Petri nets defined by the language metamodel in Figure 11.

The final activity of the process is "Interpret Model". It results in the Interpretation artifact. In this activity, a model user creates their understanding of the model by interpreting the Model artifact created in the previous step. User understanding of the model may be incomplete due to missing model elements. In our example, the better the user understands the model, the closer the Interpretation artifact is to the example model in Figure 7.

6.3. Modeling Artifacts

To use the SOLID-M modeling artifacts in model quality assessments, these artifacts need to be presented as sets of logical statements using the same formal language so that statements from different sets can be compared and reasoned about in a consistent way. We use Description Logic statements expressed in the Manchester OWL notation [27] as a language for statements of the SOLID-M modeling artifact sets. This notation is used because it can be understood by users with no Description Logic background, and processed by OWL tools for formal reasoning and visualization [20]. The full OWL specification of all artifact sets for the example agent-based SOLID-M instantiation is publicly available [26].

From the Description Logic formalism perspective, the artifact sets can be grouped into two types: TBox sets and ABox sets. The statements in the ABox sets are assertions about the individual facts associated with a single observation of the system of interest. Different observations may produce different corresponding ABox sets. The TBox sets contain terminological axioms that define the concepts and relations constituting the vocabulary for constructing the ABox sets. The TBox sets are constructed first, as they provide the conceptual constructs for the ABox sets. Inspired by the ontological meta-modeling framework proposed by Yonglin et al. [28], we define a TBox set using a UML class diagram and translate it into the corresponding Manchester OWL statements. The use of UML offers a concise yet formal representation that simplifies the understanding of a TBox set as a whole and can be unambiguously translated into OWL. The SOLID-M artifact sets are defined as follows.

System (S) is the set of all true statements about the individual elements included in the real world system of interest. The audience and analysts perceive this set as an area of the real world that is relevant to modeling goals. This set is an assertion box (ABox) that postulates true facts about the system's individual elements. Our example System set is constructed based on the informal description of the system provided in Figure 6. We assume this is a true description of the actual system and convert it into the formal OWL statements of the System set using the concepts defined in the terminological axioms of the example Ontology set OWL TBox (see Table 2). A fragment of the System set OWL ABox is presented in Table 3.

Table 2: A fragment of the Ontology set in Manchester OWL.

ID	Ontology set statement
1	Class: o:Event
2	Class: o:Agent
3	Class: o:Agent_Trace
4	Class: o:Activity
5	Class: o:MAS
6	ObjectProperty: o:isProducedBy Domain: o:Event Range: o:Agent
7	ObjectProperty: o:isProducedBy Domain: o:Event Range: o:Activity
8	ObjectProperty: o:isPartOf Domain: o:Event Range: o:Agent_Trace
9	ObjectProperty: o:isPerformedBy Domain: o:Activity Range: o:Agent
10	ObjectProperty: o:isPartOf Domain: o:Activity Range: o:MAS

Ontology (O) is the set of statements that explicitly define the constructs (concepts and relationships) and constraints in the real world. This set describes the common terminology (TBox), the vocabulary that is used to compare the statements in the other five sets of the framework. The analysts specify this set in formal knowledge representation (*e.g.*, OWL or RDFS) or meta-modeling (*e.g.*, UML) languages. Figure 10 demonstrates the UML class diagram for our example Ontology set. UML classes and named uni-directional associations define TBox concepts and relations, respectively. For example, the 'isPerformedBy' relation between the 'Activity' and 'Agent' concepts of the Ontology set is represented in Figure 10 as the 'isPerformedBy' association line from the 'Activity' class box to the 'Agent' class box. We do not specify cardinality constraints in our TBox sets. This way, we keep the diagram simple and adaptable to various instantiations of the cardinality constraints. Table 2 shows a fragment of the example Ontology set in the Manchester OWL syntax. This fragment defines five concepts as OWL classes (statements 1 to 5) and five relations as OWL object properties (statements 6 to 10). One OWL class is defined for each UML class, and

one OWL object property is defined for each UML association. For example, statement 9 corresponds to the association 'isPerformedBy' from the class 'Activity' to the class 'Agent'. These two classes are specified in statements 4 and 2, respectively.

Table 3: A fragment of the System set in OWL.

ID	System set statement in Manchester OWL
1	Individual: a1 Types: o:Agent
2	Individual: a2 Types: o:Agent
3	Individual: a3 Types: o:Agent
4	Individual: ifc_a1_a2 Types: o:Interface Facts: o:isFrom a1, o:isTo a2
5	Individual: ifc_a1_a3 Types: o:Interface Facts: o:isFrom a1, o:isTo a3
6	Individual: ifc_a2_a1 Types: o:Interface Facts: o:isFrom a2, o:isTo a1
7	Individual: ifc_a3_a1 Types: o:Interface Facts: o:isFrom a3, o:isTo a1
8	Individual: check Types: o:Activity Facts: o:isPerformedBy a1
9	Individual: c0 Types: o:Case
10	Individual: interaction1 Types: o:Interaction Facts: o:happensOver ifc_a1_a2, o:hasNext atrace1
11	Individual: atrace1 Types: o:Agent_Trace Facts: o:happensFor c0, o:isProducedBy a1, o:hasNext interaction1
12	Individual: e1 Types: o:Event Facts: o:isProducedBy check, o:happensFor c0, o:isPartOf atrace1, o:hasNext e2
13	Individual: patioent1
14	Individual: doctor2

Language (L) is the set of statements that explicitly define the constructs (concepts and relationships) and constraints of a modeling language. It specifies the grammar of the language used to construct the model. This set describes the terminology (TBox) that enables ontological analysis of the modeling grammar and syntactic verification of the model. Figure 11 and Table 4 present the UML class diagram and a fragment of the OWL TBox for the example Language set. The set includes generic Petri net concepts (*e.g.*, 'Transition', 'Place', 'Workflow_Net') and concepts specific to the agent-based models discovered by Agent Miner (*e.g.*, Agen_Net, Agent_Transition, MAS_Model). In addition to UML associations, the Language set UML diagram uses UML generalization to specify sub-class relations between the language concepts. For example, 'Agent_Net' is a sub-class of 'Workflow_Net'.

Interpretation (I) is the set of statements that the audience acknowledges and/or understands in the model. The statements in this set reflect a subjective audience's view of the real world area postulated by the System set. This set is an assertion box (ABox) that articulates the audience's interpretation of the statements in the Model set. In our example, we assume that the user incorrectly associates the B-test activity with agent a3 and does not recognize the interface from a1 to a2. A fragment of the example Interpreta-

Table 4: A fragment of the Language set in Manchester OWL.

ID	Language set statement
1	Class: l:Firing
2	Class: l:Agent_Net
3	Class: l:Agent_Transition
4	Class: l:Activity_Transition
5	Class: l:MAS_Model
6	Class: l:Transition
7	ObjectProperty: l:fires Domain: l:Firing Range: l:Transition
8	ObjectProperty: l:isPartOf Domain: l:Activity_Transition Range: l:Agent_Net
9	ObjectProperty: l:isPartOf Domain: l:Agent_Net Range: l:MAS_Model
10	ObjectProperty: l:refines Domain: l:Agent_Net Range: l:Agent_Transition

tion set OWL ABox in Table 5 represents this incorrect interpretation by the user. If a user had a complete understanding of the MAS model discovered by Agent Miner, then the Interpretation set OWL ABox would be identical to the Model set OWL ABox.

Table 5: A fragment of the Interpretation set in OWL.

ID	Model set statement in Manchester OWL
1	Individual: a1 Types: o:Agent
2	Individual: a2 Types: o:Agent
3	Individual: a3 Types: o:Agent
4	Individual: prescribe Types: o:Activity Facts: o:isPerformedBy a1
5	Individual: analyze Types: o:Activity Facts: o:isPerformedBy a1
6	Individual: yoga Types: o:Activity Facts: o:isPerformedBy a3
7	Individual: check Types: o:Activity Facts: o:isPerformedBy a1
8	Individual: B-test Types: o:Activity Facts: o:isPerformedBy a3
9	Individual: swim Types: o:Activity Facts: o:isPerformedBy a3
10	Individual: U-sound Types: o:Activity Facts: o:isPerformedBy a2
11	Individual: ifc_a1_a3 Types: o:Interface Facts: o:isFrom a1, o:isTo a3

Data (D) is the set of all statements recorded during observations of the real world system. This set is used as input for model discovery techniques. A common way to specify this set is an event log with individual event attributes used as the set statements. This set is an assertion box (ABox) containing facts about individual real world objects represented by events captured by the information system. Table 6 shows a fragment of the Data set OWL ABox. It is a translation of the given event log to OWL using the terminology defined in the Ontology set. Every line in the event log has a corresponding event statement in the Data set. For example, the event log lines 1, 4, and 5 in Table 1 correspond to OWL statements with IDs 9, 10, and 11 in Table 6. Instead of using the time attribute, the OWL object property "hasNext" is used in the OWL statements to indicate ordering of events, because the Ontology defines this object property and does not define

any time relationships for events. For example, the "hasNext" property in line 8 of the Data set OWL (Table 1) specifies that the next event after event e1 is event e2.

Table 6: A fragment of the Data set in OWL.

ID	Data set statement in Manchester OWL
1	Individual: a1 Types: o:Agent
2	Individual: a2 Types: o:Agent
3	Individual: a3 Types: o:Agent
4	Individual: check Types: o:Activity Facts: o:isPerformedBy a1
5	Individual: B-test Types: o:Activity Facts: o:isPerformedBy a2
6	Individual: swim Types: o:Activity Facts: o:isPerformedBy a3
7	Individual: c0 Types: o:Case
8	Individual: c1 Types: o:Case
9	Individual: e1 Types: o:Event Facts: o:happensFor c0, o:isProducedBy check, o:isProducedBy a1, o:hasNext e2
10	Individual: e4 Types: o:Event Facts: o:happensFor c0, o:isProducedBy B-test, o:isProducedBy a2, o:hasNext e5
11	Individual: e5 Types: o:Event Facts: o:happensFor c0, o:isProducedBy swim, o:isProducedBy a3, o:hasNext e6

Model (M) is the set of all statements about the system inferred from the data. This set is an assertion box (ABox) containing facts about the individual objects discovered from the statements in the Data set. The statements in this set are formulated using the mapping between the terminology of the modeling grammar defined in the Language set and the terminology from the Ontology set. Table 7 shows a fragment of the resulting Model set OWL ABox constructed from the MAS model of four Petri nets discovered by Agent Miner (Figure 7). We use the mapping between the concepts defined in the Ontology and Language sets to convert the elements of the discovered Petri nets into the corresponding OWL statements of the Model set. For example, the "Agent" from the Ontology is mapped to "Agent_Net" and "Agent_Transition" in the Language. The full ontology-to-language mapping and corresponding Python code are publicly available [26].

6.4. Quality Aspects

The *quality aspects graph* (Figure 9) represents all quality aspects and modeling artifacts defined in the SOLID-M framework. The graph edges and nodes correspond to the quality aspects and modeling artifacts, respectively. Each aspect groups quality characteristics and measures.

In the remainder of this section, we provide definitions for all SOLID-M quality aspects. The quality aspect definitions are accompanied by examples of agent-based quality characteristics and measures over the instantiation of the SOLID-M artifact sets described in Section 6.2.

Table 7: A fragment of the Model set in OWL.

ID	Model set statement in Manchester OWL
1	Individual: a1 Types: o:Agent
2	Individual: a2 Types: o:Agent
3	Individual: a3 Types: o:Agent
4	Individual: prescribe Types: o:Activity Facts: o:isPerformedBy a1
5	Individual: analyze Types: o:Activity Facts: o:isPerformedBy a1
6	Individual: yoga Types: o:Activity Facts: o:isPerformedBy a3
7	Individual: check Types: o:Activity Facts: o:isPerformedBy a1
8	Individual: B-test Types: o:Activity Facts: o:isPerformedBy a2
9	Individual: swim Types: o:Activity Facts: o:isPerformedBy a3
10	Individual: U-sound Types: o:Activity Facts: o:isPerformedBy a2
11	Individual: ifc_a1_a2 Types: o:Interface Facts: o:isFrom a1, o:isTo a2
12	Individual: ifc_a1_a3 Types: o:Interface Facts: o:isFrom a1, o:isTo a3

All the example measures are grounded in the ratio model for similarity between two sets proposed by Tversky [29]. In addition, the measures related to the Ontology set (LOCC, OSCC, and DOCC) are the adaptations of the *semantic coverage of ontology* measures proposed in the Quality Model of Ontology for Semantic Descriptions of Web Services [30]. The presented recall measures (MSSR, DSSR, and MDSR) are derived from the three set-theoretical recall measures defined in the BDA framework [5]. Each of the proposed measures can take values between zero and one. Zero and one correspond to the lowest and the highest levels of the related characteristic, respectively.

The example measures are linked to MAS structure elements. These elements are conceptualized as agents and interfaces connecting the agents in the given Ontology set. Table 8 shows the OWL statements for all system structure elements in the example System, Data, and Model sets.

Table 8: Structure elements in the example System, Data, and Model sets.

Structure element OWL statement	In System	In Data	In Model
Individual: a1 Types: o:Agent	Yes	Yes	Yes
Individual: a2 Types: o:Agent	Yes	Yes	Yes
Individual: a3 Types: o:Agent	Yes	Yes	Yes
Individual: ifc_a1_a2 Types: o:Interface Facts: o:isFrom a1, o:isTo a2	Yes	No	Yes
Individual: ifc_a1_a3 Types: o:Interface Facts: o:isFrom a1, o:isTo a3	Yes	No	Yes
Individual: ifc_a2_a1 Types: o:Interface Facts: o:isFrom a2, o:isTo a1	Yes	No	No
Individual: ifc_a3_a1 Types: o:Interface Facts: o:isFrom a3, o:isTo a1	Yes	No	No

Semantic Model Quality This aspect focuses on the effectiveness of the Model in depicting the System to achieve modeling objectives. Generalization is an example of a semantic quality characteristic existing in the

process mining literature [4]. It characterizes the validity of the model in relation to the system as the proportion of the statements in the Model set that are also present in the System set. We propose *Model-System Structural Completeness* as a possible characteristic for this quality aspect. This characteristic describes the extent to which the model represents the actual system structure. It can be measured by *Model-System Structure Recall (MSSR)*. We define this measure as $MSSR = |S_s \cap M_s|/|S_s|$, where $S_s \subset S$ is the set of all the statements in S that specify the system structure elements (agents and interfaces) present in reality, and $M_s \subset M$ is the set of all statements in M that specify the system structure elements represented in the model. According to Table 8, the total number of the OWL statements specifying the system structure elements in the System set is seven (three agents and four interfaces). The intersection of the system structure elements in the System and Model sets has a size of five (a1, a2, a3, ifc_a1_a2, and ifc_a1_a3). Therefore, $MSSR = 5/7 = 0.71$. It is a good result showing that the model correctly represents most of the system structure.

Interpretational Model Quality This aspect examines how well the Model interprets the Data used to discover it. It is named after the interpretation process from the BWW data quality model. Model-event log precision and recall of the 4QD and BDA frameworks can be adopted as characteristics for this quality aspect. Precision indicates the validity of the model in relation to the data. It captures the proportion of statements in the Model set that appear in the Data set. Recall indicates the completeness of the model in relation to the data. It shows what proportion of the Data set appears in the Model set. We propose *Model-Data Structural Completeness* as an example characteristic for this aspect. This characteristic describes the extent to which the model represents the system structure elements captured in the data. It can be measured by *Model-Data Structure Recall (MDSR)*. We define this measure as $MDSR = |D_s \cap M_s|/|D_s|$, where $D_s \subset D$ is the set of all the statements in D that specify the system structure elements represented in the data, and $M_s \subset M$ is the set of all the statements in M that specify the system structure elements represented in the model. According to Table 8, D_s is a subset of M_s. This means $|D_s \cap M_s| = |D_s|$ and $MDSR = 1$. The model fully represents all the system structure elements captured in the data.

Pragmatic Model Quality This aspect pertains to the degree of understanding of the model by its users. Simplicity and modularity are possible characteristics under this aspect. Model size, defined as the number of model

elements, is often used as a measure of simplicity [4]. The smaller the model size, the higher the pragmatic quality of the model. An increase in the modularity of a model enhances the understanding of the model by its users [31]. We propose *System Structure Comprehension* as a possible characteristic for this aspect. This characteristic describes how well a user comprehends the system structure as represented in the model. It is measured by *Degree of System Structure Comprehension (DSSC)*. We define this measure as $DSSC = |I_s \cup M_s|/|M_s|$, where $I_s \subset I$ is the subset of the Interpretation set I that specifies all the agents and interfaces recognized by a user, and $M_s \subset M$ is the set of all the statements in M that specify all the agents and interfaces represented in the model. As per Table 8, M_s contains five elements (a1, a2, a3, ifc_a1_a2, and ifc_a1_a3). Let us assume that a user does not recognize interface ifc_a1_a3. In this case, $DSSC = 4/5 = 0.8$.

Syntactic Model Quality This aspect concerns the correctness of the model with respect to the syntax and grammar of the language used to describe it. One of the characteristics of this aspect is the syntactic correctness of the model, which is measured by the number of syntactic errors in the model. We propose *Model Syntactic Correctness* as an example characteristic for this aspect. This characteristic addresses the correct use of the modeling language syntax in the model. It is quantified using the *Degree of Correct Model Statements (DCMS)* measure. $DCMS = |M_c|/|M|$. M is the Model set. $M_c \subseteq M$ is the subset of M containing all the statements of the Model set that are correct according to the modeling grammar specified in the Language set. *DCMS* is equal to 1, because all the statements of the Model set are syntactically correct (*i.e.*, $|M_c| = |M|$). This is a typical situation, as syntactic defects are trivial to detect and need to be fixed before analyzing other quality aspects.

Semantic Data Quality This aspect is concerned with how well the data used to discover the model represents the system. It is named after the iterative process of constructing a representation of a real world system according to the BWW representational model. The event log-system precision and recall from the BDA framework can be adopted as possible characteristics of this quality aspect. We propose *Data-System Structural Completeness* as an example characteristic for this aspect that describes how well the data collected from the system represents the system structure. This characteristic is measured by *Data-System Structure Recall (DSSR)*. We define this measure as $DSSR = |S_s \cap D_s|/|S_s|$, where $S_s \subset S$ is the set of all the statements in S that specify the system structure elements (agents and interfaces) present

in reality, and $D_s \subset D$ is the set of all the statements in D that specify the system structure elements represented in the data. According to Table 8, the total number of OWL statements specifying the system structure elements in the System set is seven (three agents and four interfaces). The size of the intersection of the system structure elements in the System and Model sets is three (a1, a2, and a3). Thus, $DSSR = 3/7 = 0.43$. This low value indicates that the data in the Data set does not capture the system structure well. This result is consistent with the complete absence of the interface data in the given event log.

Ontological Data Quality This aspect entails the degree of alignment and conformity between the Data statements and the concepts and relationships defined by the ontology. It involves mapping the individual data elements to the ontological constructs. Data construct correspondence can be used as a characteristic of how well the concepts and relationships of the chosen ontology are represented in the data. We propose *Data-Ontology Conceptual Completeness* as an example of a characteristic for this aspect. This characteristic concerns how well the ontology concepts are represented in data captured from the system. It is measured by *Data-Ontology Concept Coverage (DOCC)*. We define this measure as $DOCC = |O_c^d|/|O_c|, O_c^d \subseteq O_c \subseteq O$, where O is the Ontology set, O_c contains all the classes in O, and O_c^d contains the OWL classes from O_c that are explicitly instantiated in the Data set. The event log (Table 1) and the corresponding Data set explicitly specify the instances of the four classes from the Ontology set ('Agent', 'Activity', 'Case', and 'Event'). The remaining five classes ('MAS', 'Interface', 'Interaction', 'Agent_Trace', and 'Case_Trace') are not directly represented by the data. Thus, $DOCC = 4/9 = 0.44$. A value below 0.5 indicates that the data lacks explicit information about a significant number of concepts in the ontology (Figure 10).

Ontological Language Quality This aspect revolves around the assessment of how effectively the language aligns with the ontology. It involves an ontological analysis of the modeling grammar, which identifies construct deficits and redundancies by mapping the metamodels of the modeling grammar and ontology. We propose *Language-Ontology Conceptual Completeness* as an example of a characteristic for this aspect. This characteristic assesses how well the concepts of the ontology are represented in the modeling language. It is measured by *Language-Ontology Concept Coverage (LOCC)*. We define this measure as $LOCC = |O_c^l|/|O_c|, O_c^l \subseteq O_c \subseteq O$ where O is the Ontology set, O_c contains all the classes in O, and O_c^l contains the OWL

classes from O_c that have at least one matching class in the Language set. To calculate $LOCC$, we match the classes of the Ontology set to the corresponding OWL classes in the Language set. Table 9 presents the Ontology to Language class matching results. Three of the nine classes in O_c (classes with IDs 6, 7, and 9) are not matched to the Language set. The six matched classes belong to set O_c^l. Based on the matching, $LOCC = 6/9 = 0.67$. This value indicates a high level of ontological completeness for the modeling language, with the majority of its concepts represented. It can be improved by adding explicit support for the trace concepts to the language.

Table 9: Class matching.

ID	Ontology class	Language class
1	MAS	MAS_Model
2	Agent	Agent_Net
3	Interface	Interface_Path
4	Activity	Activity_Transition
5	Event	Firing
6	*Agent_Trace*	no match
7	*Interaction*	no match
8	Case	Token
9	*Case_Trace*	no match

Semantic Ontology Quality This aspect emphasizes the ability of an ontology to provide a conceptual foundation for the system. We propose *Ontology-System Semantic Completeness* as an example of a characteristic for this aspect. This characteristic addresses the suitability of the ontology for describing the real world system. It is measured by *Ontology-System Concept Coverage (OSCC)*. We define this measure as $OSCC = |S_c^o|/|S_c|$ where S_c is a set of all concepts required to explain the semantics of the individual entities in the System set, and $S_c^o \subseteq S_c$ is a subset of S_c containing the concepts explicitly defined as classes in the Ontology set. All the entities in the System set are classified using eleven concepts (see [26]). Nine of them are specified in the Ontology set and two, 'Patient' and 'Doctor', are assumed for individuals with names 'patient1' and 'doctor2', respectively (Table 3). This results in $OSCC = 9/11 = 0.82$. This value of $OSCC$ points to a high ability of the ontology to define the concepts assumed in system observations.

7. Discussion

We designed SOLID-M to overcome the limitations of traditional PM quality measures, which cannot adequately assess models discovered by new

types of PM algorithms, as explained in Section 5. In our example healthcare surveillance MAS, the traditional frameworks are unsuitable because they are grounded in a process-oriented ontology that lacks the concepts needed to capture structural elements (e.g., agents and their interfaces). SOLID-M addresses this gap by introducing a generic, flexible mechanism for defining quality measures based on an explicitly specified ontology. Using this mechanism, we instantiate SOLID-M with an agent-based ontology and associated quality measures, enabling the evaluation of MAS models in terms of agents, their interfaces, and their local behaviors.

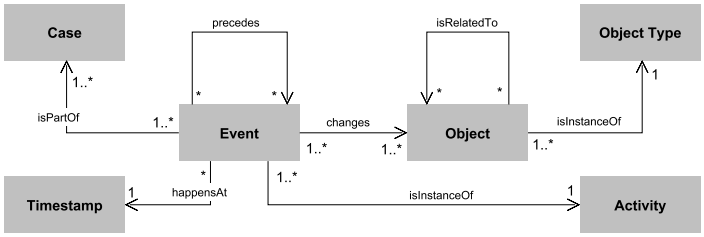


Figure 12: A UML diagram capturing an example object-centric ontology.

The example in Section 5 can be viewed from the Object Centric Process Mining (OCPM) perspective [2] as a process involving four object types (patient type and three doctor types). In this case, one can create an OCPM instantiation of SOLID-M grounded in an object-centric ontology, selecting Object-Centric Petri Nets (OCPNs) as a modeling language. Such an object-centric ontology is depicted in Figure 12. Consequently, OWL statements in all sets of SOLID-M artifacts and quality measures across all SOLID-M quality aspects should be expressed in terms of the concepts from this ontology. To assess the Interpretational Model Quality aspect of OCPN models discovered from object-centric event data, one can define the Model-Data Behavioral Completeness and Model-Data Structural Completeness characteristics. These characteristics describe the extent to which OCPN models represent behavior (case traces) and structure (objects) elements, respectively, as captured in the object-centric event logs. The behavioral completeness can be measured using the OCPN fitness measure defined in [32]. An example of a measure that can be used to assess the structural completeness is the object-centric structure fitness (OCSF) $OCSF = |D_o \cap M_o|/|D_o|$, where D_o is the subset of the Data set that contains all the statements specifying object instances included in the log, and M_o is the subset of the Model set with all

the statements specifying object instances represented in the OCPN model.

Table 10 maps the elements of SOLID-M to the elements of the existing quality frameworks. In this table, the first column lists the SOLID-M modeling artifacts and quality aspects from Figure 9. The remaining four columns list the corresponding elements of the four quality frameworks reviewed in Section 4. For example, Data in SOLID-M corresponds to Event Log in 4QD and BDA frameworks, and to Information System State in the BWW framework. It has no corresponding element in the SEQUAL framework. If a cell in the table contains the "no match" label, it indicates that the corresponding SOLID-M element has no equivalent in the respective framework.

Table 10: Mapping of SOLID-M elements to elements of other quality frameworks.

SOLID-M Elements	**4QD Elements**	**BDA Elements**	**SEQUAL Elements**	**BWW Elements**
System	System	System	Domain	User's Observed View of Real World System
Ontology	no match	no match	no match	Ontology
Language	no match	no match	Language	Grammar
Interpretation	no match	no match	Audience Interpretation	User's Inferred View of Real World System
Data	Event Log	Event Log	no match	Information System State
Model	Process Model	Process Model	Model	User's Inferred View of Real World System
Semantic model quality	Generalization	Model - System Precision and Recall	Semantic quality	no match
Interpretational model quality	Fitness, Precision	Model - Event Log Precision and Recall	no match	no match
Pragmatic model quality	Simplicity	no match	Pragmatic quality	no match
Syntactic model quality	no match	no match	Syntactic quality	no match
Ontological model quality	no match	no match	no match	no match
Semantic data quality	no match	Event Log - System Precision and Recall	no match	Data deficiencies
Ontological data quality	no match	no match	no match	no match
Ontological language quality	no match	no match	no match	no match
Semantic ontology quality	no match	no match	no match	no match
Pragmatic ontology quality	no match	no match	no match	no match

SOLID-M extends the BDA framework by generalizing the BDA behavior sets into ontology-based statement sets. Elements of the BDA sets are process traces. In SOLID-M, elements of the statement sets are not limited

to process traces. They can represent any real world entities (behavioral or structural) formulated in terms of a given ontology. We follow the idea from the BWW representational model to use an ontology as a reference language that defines fundamental concepts representing the structure and behavior of the real world. These concepts remain invariant across the languages used for all statement sets of the SOLID-M framework. This makes it possible to compare and reason about the statements from different sets in a consistent manner.

In SOLID-M, the Ontology set is an explicit variable parameter. A specific SOLID-M instantiation specifies an ontology that fits the intended use. This approach is different from the 4QD and BDA frameworks, which assume the underlying behavior-only ontology of the single-case notion. The ability to select an explicit ontology allows a straightforward instantiation of the SOLID-M framework for new types of process mining. First, an ontology is selected that supports the concepts and assumptions of the new PM type. Then, the statements in all sets of the framework are formulated using the concepts from the selected ontology. Finally, ontology-specific definitions of characteristics and measures are provided for each SOLID-M quality aspect.

SOLID-M extends the SEQUAL framework by introducing the Data set of statements and the quality aspects related to this set. We achieve this by examining the relationships between the Model, Data, System, and Interpretation sets through the lenses of the representation and interpretation processes in the BWW data quality framework. From the perspective of the representation process, the System and Data sets of SOLID-M play the roles of the real world system and the information system state, respectively. From the perspective of the interpretation process, the Model and Interpretation sets of SOLID-M play the roles of an externalized and internalized user's views of the real world system (the system) as inferred from the information system state (the data). A model discovered from the data is understood as an intermediate externalized result of the interpretation process. This is why the SOLID-M quality aspect linking the Data and Model sets is called "Interpretational model quality."

The SOLID-M treatment of the interpretation process is different from the BWW framework. In BWW, this process produces a user's view of a real world system directly from data captured in an information system. In SOLID-M, the interpretation process is split into two sub-processes: (i) an analyst interprets data to construct a conceptual model of a real world system represented by the data using model discovery techniques, (ii) users interpret

the model to infer their view (interpretation) of the system.

The proposed framework has several limitations. Every quality aspect is associated with a directed pair (source, target) of modeling artifacts, where the quality of the target is characterized in relation to the source. Quality measures calculated over one target artifact, such as model size measures of pragmatic model quality, are not directly supported.

The framework does not cover the pragmatic quality of the Ontology and Language artifacts. This excludes the definition of ontology and language measures related to user interpretation. A possible direction for extending SOLID-M is to define additional quality aspects, focusing on the pragmatic quality of ontologies and modeling languages.

Our framework focuses on auto-discovered conceptual models that are intended for direct use by human users. Quality characteristics and measures specific to the Business Process Simulation (BPS) context are not considered. SOLID-M can be extended to support BPS model quality assessment by combining our ontology-aware approach with the BPS-specific multi-perspective quality measures defined in the framework for measuring the quality of BPS models [33].

SOLID-M addresses the quality of conceptual models discovered from data, not discovery algorithms themselves. Quality characteristics of discovery algorithms (such as time and space complexity) are important factors in the evaluation of new PM techniques. Extending SOLID-M with quality aspects of discovery algorithms is an interesting topic for future research.

This paper does not develop a standard reference ontology for MAS discovery. A fully flagged reference ontology for data-driven agent-based modeling would enable improved SOLID-M instantiations for MAS model quality assessments. A study to propose a MAS ontology and investigate representational bias for MAS discovery is future work.

8. Conclusion

This article presents SOLID-M, a quality framework for conceptual models discovered from event data. The framework enables a systematic assessment of the quality of discovered models by capturing both behavioral and structural aspects. It also supports the explicit specification of the assumptions underlying the techniques and measures used in model construction and quality evaluation.

We provide an example of how to instantiate this framework to assess the quality of MAS models discovered using the Agent Miner algorithm. As the adoption of AI agents in business processes continues, we expect greater availability of data representing the local behavior of AI and human agents contributing to emergent system-level business outcomes. The use of agent-based process mining techniques to generate MAS models from these data will increase. To address the need for better ways to assess the quality of discovered MAS models, one can extend the MAS instantiation of SOLID-M with new quality measures that reflect the specifics of interactions between human and AI agents across different business domains.

Acknowledgments. Andrei Tour was supported via an "Australian Government Research Training Program Scholarship."

References

[1] G. Vossen, The process mining manifesto—An interview with Wil van der Aalst, Information Systems 37 (3) (2012) 288–290. doi:10.1016/j.is.2011.10.006.

[2] W. M. P. van der Aalst, A. Berti, Discovering object-centric Petri nets, Fundamenta Informaticae 175 (1–4) (2020) 1–40. doi:10.3233/FI-2020-1946.

[3] W. Bandara, G. G. Gable, M. Rosemann, Factors and measures of business process modelling: Model building through a multiple case study, European Journal of Information Systems 14 (4) (2005) 347–360. doi:10.1057/PALGRAVE.EJIS.3000546.

[4] W. M. P. van der Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer, 2011. doi:10.1007/978-3-642-19345-3.

[5] J. C. A. M. Buijs, B. F. van Dongen, W. M. P. van der Aalst, Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity, International Journal of Cooperative Information Systems 23 (1) (2014). doi:10.1142/S0218843014400012.

[6] A. Tour, A. Polyvyanyy, A. A. Kalenkova, Agent system mining: Vision, benefits, and challenges, IEEE Access 9 (2021) 99480–99494. doi:10.1109/ACCESS.2021.3095464.

[7] A. Senderovich, M. Weidlich, A. Gal, A. Mandelbaum, Queue mining for delay prediction in multi-class service processes, Information Systems 53 (2015) 278–295. doi:10.1016/j.is.2015.03.010.

[8] O. I. Lindland, G. Sindre, A. Sølvberg, Understanding quality in conceptual modeling, IEEE Software 11 (2) (1994) 42–49. doi:10.1109/52.268955.

[9] M. Bunge, Ontology I: The Furniture of the World, Treatise on Basic Philosophy, Springer Netherlands, 1977. doi:10.1007/978-94-010-9924-0.

[10] Y. Wand, R. Y. Wang, Anchoring data quality dimensions in ontological foundations, Communications of the ACM 39 (11) (1996) 86–95. doi:10.1145/240455.240479.

[11] T. Jiang, Z. Sun, S. Fu, Y. Lv, Human-AI interaction research agenda: A user-centered perspective, Data and Information Management 8 (4) (2024) 100078. doi:10.1016/j.dim.2024.100078.

[12] A. Tour, A. Polyvyanyy, A. A. Kalenkova, A. Senderovich, Agent miner: An algorithm for discovering agent systems from event data, in: Business Process Management, Vol. 14159 of Lecture Notes in Computer Science, Springer, 2023, pp. 284–302. doi:10.1007/978-3-031-41620-0_17.

[13] K. Peffers, T. Tuunanen, M. A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, Journal of Management Information Systems 24 (3) (2007) 45–77. doi:10.2753/MIS0742-1222240302.

[14] K. Peffers, M. Rothenberger, T. Tuunanen, R. Vaezi, Design science research evaluation, no. v.7286 in Lecture Notes in Computer Science, Springer, 2012, pp. 398–410.

[15] J. Krogstie, G. Sindre, H. Jørgensen, Process models representing knowledge for action: A revised quality framework, European Journal of Information Systems 15 (1) (2006) 91–102. doi:10.1057/PALGRAVE.EJIS.3000598.

[16] Y. Wand, R. Weber, Toward a theory of the deep structure of information systems, in: International Conference on Information Systems, Association for Information Systems, 1990, p. 3.

[17] N. Guarino, D. Oberle, S. Staab, What is an ontology?, in: Handbook on Ontologies, International Handbooks on Information Systems, Springer, 2009, pp. 1–17. doi:10.1007/978-3-540-92673-3_0.

[18] L. von Bertalanffy, An outline of general system theory, The British Journal for the Philosophy of Science 1 (2) (1950) 134–165.

[19] G. Antoniou, F. van Harmelen, Web ontology language: OWL, in: Handbook on Ontologies, International Handbooks on Information Systems, Springer, 2009, pp. 91–110. doi:10.1007/978-3-540-92673-3_4.

[20] J. Méndez, C. Alrabbaa, P. Koopmann, R. Langner, F. Baader, R. Dachselt, Evonne: A visual tool for explaining reasoning with OWL ontologies and supporting interactive debugging, Computer Graphics Forum 42 (6) (2023). doi:10.1111/CGF.14730.

[21] W. M. P. van der Aalst, Object-centric process mining: Dealing with divergence and convergence in event data, in: Software Engineering and Formal Methods, Vol. 11724 of Lecture Notes in Computer Science, Springer, 2019, pp. 3–25. doi:10.1007/978-3-030-30446-1_1.

[22] P. P.-Y. Wu, C. Fookes, J. Pitchforth, K. Mengersen, A framework for model integration and holistic modelling of socio-technical systems, Decision Support Systems 71 (2015) 14–27. doi:10.1016/J.DSS.2015.01.006.

[23] R. Pérez-Castillo, F. Ruiz, M. Piattini, A decision-making support system for Enterprise Architecture Modelling, Decision Support Systems 131 (2020) 113249. doi:10.1016/J.DSS.2020.113249.

[24] C. W. Morris, Writings on the General Theory of Signs, no. 16 in Approaches to Semiotics, De Gruyter, 2014.

[25] J. Krogstie, Model-Based Development and Evolution of Information Systems, Springer, 2012. doi:10.1007/978-1-4471-2936-3.

[26] A. Tour, A. Polyvyanyy, A. A. Kalenkova, SOLID-M instantiation example (2025).
URL `https://doi.org/10.26188/25458967.v4`

[27] M. Horridge, N. Drummond, J. Goodwin, A. L. Rector, R. Stevens, H. Wang, The manchester OWL syntax, in: OWL: Experiences and

Directions, Vol. 216 of CEUR Workshop Proceedings, CEUR-WS.org, 2006.

[28] L. Yonglin, Z. Zhi, L. Qun, An ontological metamodeling framework for semantic simulation model engineering, Journal of Systems Engineering and Electronics 31 (3) (2020) 527–538. doi:10.23919/JSEE.2020.000032.

[29] A. Tversky, Features of similarity, Psychological Review 84 (4) (1977) 327–352. doi:10.1037/0033-295X.84.4.327.

[30] H. Zhu, D. Liu, I. Bayley, A. Aldea, Y. Yang, Y. Chen, Quality model and metrics of ontology for semantic descriptions of web services, Tsinghua Science and Technology 22 (3) (2017) 254–272. doi:10.23919/TST.2017.7914198.

[31] H. A. Reijers, J. Mendling, Modularity in process models: Review and effects, in: Business Process Management, Vol. 5240 of Lecture Notes in Computer Science, Springer, 2008, pp. 20–35. doi:10.1007/978-3-540-85758-7_5.

[32] J. N. Adams, W. M. P. van der Aalst, Precision and fitness in object-centric process mining, in: International Conference on Process Mining, IEEE, 2021, pp. 128–135. doi:10.1109/ICPM53251.2021.9576886.

[33] D. Chapela-Campa, I. Benchekroun, O. Baron, M. Dumas, D. Krass, A. Senderovich, A framework for measuring the quality of business process simulation models, Information Systems 127 (2025) 102447. doi:10.1016/J.IS.2024.102447.