PRODUCT Management & Development
An International Journal for Transdisciplinary Research in Innovation

ISSN 2237-5228

Journal homepage
www.pmd.igdp.org.br

Research Article

Class-Activity-Status model for object-oriented ontology construction supporting domain knowledge integration to achieve business-IT alignment

Min-Hua Chao[1]*, Amy J. C. Trappey[1]

[1]*Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu, 300, Taiwan*

Abstract

Enterprises are investing in digital transformation, hoping to create convenience and high values toward system-based product and service provisions in the paradigm of smart manufacturing. The achievement of digital transformation is not just the adoption of digital technologies, but information systems' agility and dynamic capabilities. This research proposes a domain knowledge integration method to achieve business-IT alignment (BITA), complying with the reality and essence of business activities analyzed and synthesized from object-oriented perspectives. The four characteristics of object-oriented ontology (OOO), i.e., essentiality, interpretability, stability, and incompleteness, provide critical directions for detecting the gaps between knowledge and reality. This study also deploys the unified modeling language-based (UML-based) Class-Activity-Status (CAS) model as a communication language for expressing diverse enterprise realities by standardizing basic principles. The knowledge integration ability of the proposed methodology is shown by implementing the CAS model for manufacturing bill-of-materials (MBOM) domain knowledge. The proposed OOO characteristics (as the mindset), CAS model (as the modeling tool), and the theoretical MBOM domain modeling example have shown great novelty and implication of the transdisciplinary nature of manufacturing service philosophy, domain knowledge engineering, and digital technology integration.

Keywords: digital transformation, knowledge engineering (KE), object-oriented ontology (OOO), business-IT alignment (BITA).

List of abbreviations

Abbreviation	Full name
AI	Artificial Intelligence
BITA	Business-IT Alignment
BPM	Business Process Management
BPMN	Business Process Model and Notation
BPR	Business Process Reengineering
CAS model	Class-Activity-Status model
DTO	Digital Twin of an Organization
EA	Enterprise Architecture
IPA	Intelligent Process Automation
IoT	Internet of Things
IT	Information Technology
MBOM	Manufacturing Bill-of-Materials
NLP	Natural Language Processing
OOO	Object-Oriented Ontology
UID	Unique IDentifier
UML	Unified-Modeling Language

1. Introduction

Despite the growing adoption of Industry 4.0 and emerging digital technologies, many companies fail in their digital transformation efforts (Ramesh, 2019). People and organizational culture are often cited as key factors contributing to this low success rate (Liu et al., 2022). To foster a positive cycle and ensure a successful digital

Received: November 19, 2022. Accepted: April 14, 2023.
***Corresponding author:** Min-Hua Chao. E-mail: s106034803@m106.nthu.edu.tw

transformation, businesses should adopt a gradual improvement approach that identifies and solves process problems (Malaurent & Karanasios, 2020). Digital transformation should focus on customer needs, domain knowledge, and digital technology capabilities to create value (Chen et al., 2021). This study proposes using OOO as a shared mental model to enable BITA at all levels of an organization's digital transformation agenda.

The study is structured as follows: Section 2 provides a literature review on digital transformation, OOO, and BITA. Section 3 introduces the key methodologies, including the OOO concept as the core mindset and the UML-based CAS model as a practical method to facilitate domain knowledge integration. Section 4 illustrates modeling issues with OOO principles and CAS modeling rules. Section 5 demonstrates the knowledge integration process using the MBOM domain knowledge modeling as an example. Section 6 compares the study with related research and discusses it. Finally, Section 7 concludes the study, claiming its contribution and implications.

2. Literature review

In this section, the literature on OOO is reviewed, with a mind to examining the appropriate framework for the primary determinant of digital transformation.

2.1. Digital transformation

This section reviews related topics of digital transformation, including comparing with BPR, IPA, and hyperautomation.

2.1.1. The essence of digital transformation and BPR

Studies have conflicting views on the importance of IT in both BPR and digital transformation (Al Tal et al., 2020; Hammer & Champy, 2009). While IT can offer process automation and innovative thinking for BPR (Dinata, 2020), digital transformation requires established businesses to adopt new structures for seamless goods and services (Sebastian et al., 2017). The key difference between BPR and digital transformation is the process of knowledge formation from data to facilitate decision-making (Schallmo & Williams, 2018), and integrated alignment is emphasized as a digital transformation strategy for corporations. Therefore, this study focuses on digital strategy and methodology rather than specific technologies.

2.1.2. IPA

Feio & Santos (2022) propose a strategic model and framework for IPA to facilitate digital transformation. The model consists of four strategic groups: environment, pillars, visualization, and creation. The environment highlights the importance of BITA. The pillars consist of BPM, knowledge management, and business intelligence. The visualization group focuses on demonstrating IPA value to the organization. The creation group consists of emerging digital technologies, including robotic process automation, chatbots, big data, and AI models.

2.1.3. Hyperautomation

Gartner's top technology trends for 2020 list hyperautomation as the most important one (Panetta, 2019), but few organizations have mastered their business processes enough to realize the benefits of digital technology (Kirchmer & Franz, 2020). Hyperautomation automates knowledge labor and aims to involve everyone in an organization. It combines advanced techniques like RPA, process mining, AI, NLP, and OCR (Haleem et al., 2021) and requires a DTO to model scenarios, connect data, digital technologies, and organizational reality (Schneider & Kokshagina, 2021). Section 2.2 reviews related works of OOO to model reality.

2.2. OOO

Harman (2018) presents a detailed framework for OOO concepts, and Yoran (2018) finds similarities between object-oriented programming and new metaphysics, suggesting potential for contribution to the ontographic project.

2.2.1. Essence and reality

Quantified manufacturing predicts a new automated world made of fully autonomous non-human entities, which requires philosophical movements such as speculative realism and OOO to deal with emerging challenges beyond human correlations (McCormack, 2018; Recht & Wiberg, 2018). These movements emphasize the independent existence of entities, making them more suitable for issues affecting a posthuman world. By studying objects in their genuine appearance and considering their unknown layers, more possible essences can be revealed (Ferro, 2019).

The relationship between knowledge and truth is not always clear, and according to OOO, reality should be acknowledged and revered to overcome idealism. No one is actually in possession of knowledge or truth, and reality must be approached indirectly. Hence, OOO is a discipline that detects the gap between knowledge and reality (Harman, 2018).

2.2.2. OOO perspective in enterprise system development

The spirit of OOO can be seen in smart manufacturing where each machine corresponds to an entity object with its own properties and behaviors, allowing for a complete system when individuals can interact smoothly from their respective perspectives (Bao et al., 2019; Sahal et al., 2021; Yu et al., 2023). OOO argues that objectivity and reality exist independently of humans, and it describes reality from a materialistic philosophy (Harman, 2018).

To avoid half-baked requirement analysis, each object's unique definition and role should be determined by its perspective, not just the human perspective. Otherwise, the enterprise's essence becomes obscured. This misalignment often occurs due to inadequate IT training for business personnel who can overlook the importance of IT in the enterprise system development (Jonathan et al., 2019; Mongale et al., 2021).

2.3. BITA

This study aims at bridging the alignment gap between business and IT. Related works of BITA are reviewed in this section.

2.3.1. BPM

BPM involves three important logics: process modeling, infrastructure alignment, and procedural actor. However, maintaining the business process model, IT-business interface, and system integrations poses challenges due to a lack of up-to-date modeled documentation. Digital transformation creates tension when applying these prior logics, leading to the need for advanced BPM logic with light touch processes, infrastructural flexibility, and mindful actors. This has practical implications for managing changes in business processes (Baiyere et al., 2020).

2.3.2. Modeling methods

EA provides a high-level overview of an enterprise’s business and IT systems and their interrelationships, and is widely recognized among software practitioners and used as a communication tool for employees at all levels (Buriánek, 2021; Gong & Janssen, 2019; Wilson Júnior et al., 2021; Venkatesan & Sridhar, 2019) (Wohlrab et al., 2020), despite criticism from some experts (Kotusev, 2021).

UML is commonly used as a standard for describing the information system framework in EA (Kobryn, 1998; Ozkaya & Erata, 2020), and its ability to describe models is also used in various fields (Li et al., 2022). Despite its constantly evolving and increasingly complex legends, only 20% of simple UML legends and notation are required for 80% of software usage (Siau & Loo, 2006). By selecting the core elements of UML, it can provide an effective communication language for the enterprise and enable communication with manufacturing execution systems and ERP systems (Cupek et al., 2019).

2.3.3. Discussion

What is the modeling goal in digital transformation? Agile development offers flexibility, but lacks a clear definition of final requirements. Waterfall development is useful for contracts but is inflexible to change. Fundamental objectives can provide direction despite difficulty.

The goal of information system development is to improve efficiency and streamline procedures, rather than creating a new reality (May et al., 2022). Blindly promoting information systems without critical evaluation is unnecessary since there may be better ways to achieve the same objectives. Instead of replicating existing operating models, information systems should offer better solutions and be flexible enough to accommodate changes in business. Better understanding of domain knowledge is required to predict changes, and predictable changes allow for quicker reactions.

3. Key methodologies: OOO concept and UML-based CAS model

To help enterprises navigate the challenges of digital transformation and respond quickly to changes, this research proposes a comprehensive framework and methodology centered around OOO, including an overall domain knowledge integration framework, the OOO concept, and a CAS model workflow. This section provides an overview of these key methodologies, with guidance offered for their implementation.

3.1. Domain knowledge integration framework

This study combines the UML-based CAS model with OOO to create a domain knowledge integration framework. The four OOO characteristics - essentiality, interpretability, stability, and incompleteness - guide the exploration of domain ontology and underpin this framework. The class diagram in the CAS model is crucial for knowledge acquisition and structural modeling, while the activity diagram bridges static and dynamic modeling, and the state diagram ensures stability. We propose an OOO construction process using these three diagrams, established rules, and a three-phase workflow to integrate domain knowledge (see Figure 1).

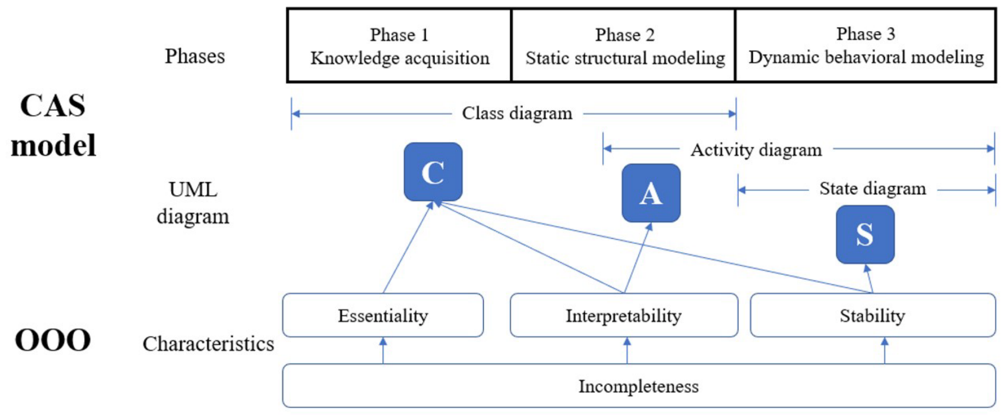

Figure 1. The domain knowledge integration framework.

3.2. Four characteristics of OOO

This study focuses on OOO, a philosophy of realism and materialism that maintains the finitude of objects (Harman, 2018). Each object has its own point of view and interacts with others, resulting in diverse business behaviors. OOO takes the core of objects as the essence to explain various phenomena, similar to atomic theory. The proposed four OOO characteristics - essentiality, interpretability, stability, and incompleteness - define the nature and interactions of independent objects, enabling reasonable explanations of all business activities.

3.2.1. Essentiality

Essentiality is a key characteristic of OOO and requires the modeling of an enterprise based on the nature of its operations. In large and stable enterprises, the concept behind each form must be clear and specific. This study proposes two principles to ensure essentiality:

Principle 1-1: Every form should correspond to at least one implicit essence; otherwise, the form may not need to exist.

Principle 1-2: The existence of the essence is sufficient and does not need to be verified by form.

In Industry 2.0, assembly lines improved process efficiency, leading to standardized operating procedures. In Industry 3.0, the development of computers made enterprises seek electronic work processes. However, relying too heavily on paper or electronic processes can enslave organizations and hinder their ability to adapt to changes in the real world. Digitization and systematization aim to enhance digital transformation and enable enterprises to adapt to future challenges. Business personnel prioritize the success of all operations over data accuracy on the information system, resulting in the system becoming a significant burden for the company. Despite being unsound, the system has become deeply ingrained in the organizational process, hindering any potential progress.

High collaboration between experts and designers is needed for essence-oriented modeling. This modeling approach utilizes business operation essence as the modeling criterion, eases some restrictions, and supports digital transformation. However, the presentation of the model should not be complex and must show the necessary structure to involve all employees. Common frameworks such as Zachman, TAFIM, and TOGAF are too complex for successful implementation in enterprises. Taking the Zachman framework as an example. The Zachman framework has 36 subunits, making it difficult for managers and high-level employees to understand and implement. It covers many facets of enterprise operations but cannot provide a complete answer in practical applications. To bridge this gap, the proposed CAS model is a clear and implementable first step for domain knowledge integration in the electronization and digital transformation agenda.

3.2.2. Interpretability

OOO should be interpretable for all business activities, rather than modeled by all requirements. Traditional systems engineering and software engineering require fulfilling the requirements seen as contracts, but in the OOO modeling process, developing an essence-oriented business model is prioritized, and requirements only need to be interpreted rather than fully satisfied. This study proposes the following principles:

Principle 2a: Customers cannot fully express their system requirements.

Principle 2b: Any information system developed according to the requirements cannot fully meet them.

Principle 2a and 2b state that user requirements are limited by their understanding of current operations and technology. IT departments face two scenarios when fulfilling these requirements: replicating existing operations offers no benefits while new solutions may spur business process improvements. Collaboration between business and IT departments is crucial for successful digital transformation, with continuous cycles of discussions, prototypes, design, validation, deployment, and promotion.

The need for corporate agility and successful digital transformation can be seen in the scenarios described above. Constantly updating a large number of EA documents contradicts the agile approach, but documentation is still necessary for an enterprise. OOO must be able to interpret both the front-end behavior of business activities and the back-end architecture of information services. The interpretability of OOO is critical as it expresses domain ontology.

3.2.3. Stability

OOO stability determines enterprise operating framework health. The low success rate of ERP system imports is explained by the lack of awareness of domain knowledge ontology. While ERP is viewed as the Holy Grail on the journey from Industry 2.0 to 3.0, it is almost impossible for all enterprises to have the same knowledge ontology. New enterprises that completely adopt an ERP system at the outset risk shutting down existing business activities if the business logic implied by the system is adopted.

In practical application, ERP systems often require customization to fit existing operations, but this customization often ignores the enterprise's knowledge ontology. Full-module ERP system authorization is expensive, leading companies to use different versions of ERP systems or information systems in different business areas, creating disjointed knowledge chains. The enterprise's operation encompasses many areas with substantial and stable knowledge ontology, and representing this ontology is crucial.

Principle 3: Knowledge ontology that can contribute to the stable operation of an enterprise is bound to exist.

To promote effective transformation, the idea that “ERP is the only standard answer” does not respect the existing knowledge of the enterprise and may not be convincing. Alignment of the knowledge ontology is crucial for successful promotion, as failure to do so may limit the benefits.

3.2.4. Incompleteness

Whenever new business activities are added or existing ones change within the enterprise, a reliable domain knowledge ontology is necessary to accurately interpret these changes. Otherwise, the incompleteness of the domain knowledge ontology becomes apparent.

Principle 4: The domain ontology of any enterprise is always incomplete.

The incompleteness of knowledge ontology in an enterprise can result from incomplete or biased knowledge acquisition. Assessing the ontology's interpretability for actual business activities is crucial to determine its completeness. Verification through repeated business activities is necessary to demonstrate an ontology's stability. Unexplainable business activities indicate an incomplete ontology. Hence, constructing an OOO model that corresponds to the domain ontology requires cross-disciplinary experts. On the other hand, the development of emerging technologies enables previously impossible behaviors and contributes positively to the incompleteness of enterprise domain knowledge.

Domain ontology is inevitably incomplete due to various reasons, and it is difficult to ensure its completeness at any given time. Enterprises need to recognize this incompleteness to adapt to new changes and challenges in business activities and maintain momentum for continuous improvement and growth. This incompleteness also supports the saying, “There is no best, for there is always better,” in the context of OOO.

3.3. CAS model workflow

The proposed CAS model in this research serves to create a domain knowledge ontology that unifies enterprise employees. It facilitates communication for the business department and quick architecture system construction for the IT department. The model must remain stable while also being easily modified to support the diversity of external requirements in the digital transformation era. A challenging iterative process is used to ensure consistency between domain knowledge representation and reality. A CAS model construction process is illustrated, including two aspects and three phases, to bridge this gap (Schmidt et al., 2020), as show in Figure 2.

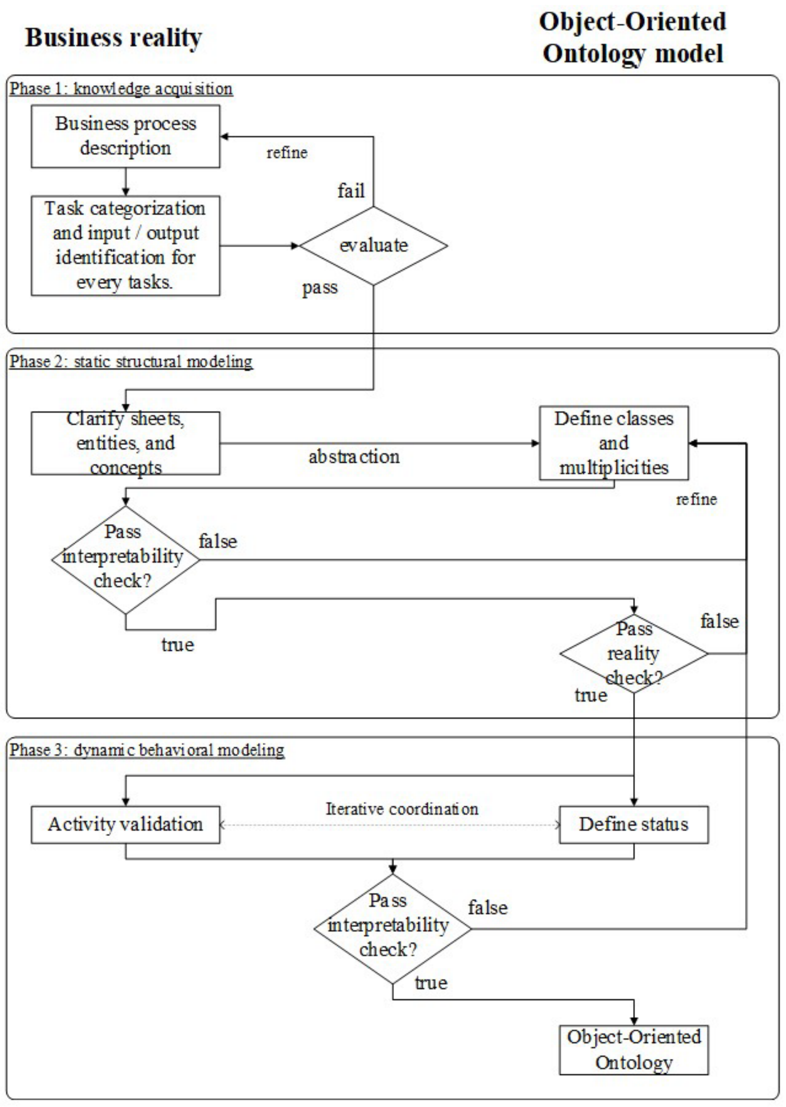
Business reality
Object-Oriented Ontology model
Phase 1: knowledge acquisition
Business process description
Task categorization and input / output identification for every tasks.
evaluate
refine
fail
pass
Phase 2: static structural modeling
Clarify sheets, entities, and concepts
abstraction
Define classes and multiplicities
refine
Pass interpretability check?
false
true
Pass reality check?
false
true
Phase 3: dynamic behavioral modeling
Activity validation
Iterative coordination
Define status
Pass interpretability check?
false
true
Object-Oriented Ontology

Figure 2. OOO model construction workflow.

The two aspects are the business reality and the OOO model. The three phases are knowledge acquisition, static structural modeling, and dynamic behavioral modeling. The latter two are inspired by the two main UML categories, i.e., structural and behavioral, which have been adopted not only in software engineering but also in digital twins modeling in manufacturing (Delgado & Oyedele, 2021). The CAS model is divided into three parts: class, activity, and status, which are intuitively associated with the UML class diagram, activity diagram, and state diagram.

3.3.1. Phase 1: knowledge acquisition

Phase 1 involves describing a business process using UML activity diagrams, which are based on object creation and status changes. It consists of two parts: task categorization and input/output identification. Part one involves object construction, business operation, and analysis, while part two involves identifying the input or output attached to all arrowheads. The flowchart is evaluated for completeness and variability before entering phase two. If the evaluation is passed, phase two is entered, but if not, the flowchart must be refined, and task categorization and input/output identification must be reclarified.

3.3.2. Phase 2: static structural modeling

The input and output on the flowchart are divided into forms, entities, or concepts, then abstracted to create the static structural model in a UML class diagram. This recursive process strengthens and stabilizes the OOO static structure by confirming interpretability and reality. UML class diagram components like class, enumeration, interface, and package are used, with relationships such as association, inheritance, realization, aggregation, and composition marked with multiplicity, as per CAS specification.

This leads to the first rule of the static structure of the CAS model:

Rule 1: One end of multiplicity must be finite.

In the CAS model, many-to-many associations are not allowed as they do not accurately reflect how things operate in OOO. Such relationships must be broken down into smaller pieces to correspond to specific individual objects when viewed from a first-person perspective. Many-to-many associations symbolize the incompleteness of OOO and represent a non-conformity of the first normalization of the database. The second rule of OOO static structure is thus defined based on this.

Rule 2: The existence of a many-to-many relationship represents the incompleteness of OOO, and one of the following adjustments must be made: adopt the conjunctional class, or abandon this relationship.

Constructing an object-oriented domain model supports service providers to face diverse environments, emphasizing the importance of essentiality, interpretability, and stability of OOO (Venkatraman & Venkatraman, 2019). The matching concept validates the consistency of OOO's essence and reality based on the definition of classes to ensure interpretability to meet user requirements. Although meeting user requirements is crucial in software engineering, it's important to find an object-centered existential meaning based on Principle 2a and Principle 2b since requirements cannot be fully satisfied.

3.3.3. Phase 3: dynamic behavioral modeling

In phase 3, the focus is on dynamic behavioral modeling, which involves iteratively coordinating activity validation and status defining until successful interpretability check. Dynamic business processes and behaviors are depicted using activity diagrams, with each activity corresponding to the workflow level and the essence of business reality. This leads to the proposal of Rule 3.

Rule 3: An essence will be realized by exactly one activity, while an activity might contain some essence or no essence.

When the scopes are not correctly correlated with the essence contained in it, the class status cannot be precisely described to satisfy its essence. Covering the class status with the activity level of business behavior fails to meet OOO's stability requirements. This ambiguity confuses the status of classes and business activities, leading to the variability of activities affecting the enterprise business model. This leads to the proposal of Rule 4.

Rule 4: The status should be defined to reflect the essence instead of the workflow step.

3.4. Domain knowledge integration guidance

Table 1 describes the connections between OOO characteristics and principles, CAS model components, domain knowledge integration guidance, and common situations or challenges.

Table 1. The connections between OOO, CAS model, and the guidance for knowledge integration.

OOO characteristics and principles	Enterprise domain knowledge integration guidance	Components in the CAS model	Common situations or challenges
Essentiality	· Inside-out essence-oriented modeling.	Class	Modeling with worksheets.
Principle 1-1	· Pay attention to the essential concepts of business operations.	(Rule 1)	
Principle 1-2	· The worksheet should not be mistaken for ontology.		
Interpretability	· Provide flexibility for dynamically changing businesses.	Class, Activity	· Service-oriented modeling
Principle 2a, 2b	· Outside-in requirements verification	(Principle 5)	· Process-oriented modeling
			· Misalignment of business and IT
Principle 3	· Conscientious business personnel and proactive IT personnel	Class, Status	· Business personnel as the clients
	· Focus on the status of the object's life cycle.	(Principle 6)	· IT personnel as the suppliers
			· Focus on the steps of operation
Incompleteness	· Incomplete knowledge acquisition	Class, Activity, Status	· Lack of awareness of knowledge ontology.
Principle 4	· The application of digital technologies	(Rule 2)	· Lack of interdisciplinary experts as a bridge between business and IT.
			· Lack of proper tools.

Essentiality is the primal characteristic of the OOO concept, which suggests an inside-out essence-oriented modeling of domain knowledge. Enterprises should focus on the essential concepts of business operations instead of worksheets when defining knowledge ontology classes. Modeling with worksheets causes data of the same essence to be repeated in several business departments, leading to broken links in data flow or information islands. Outside-in requirement verification examines interpretability. In the CAS model construction process, the essences are confirmed as components of business behavior through the class and activity diagrams. However, service or process-oriented modeling can result in the misalignment of business and IT, as IT may only focus on requirements or process appearance without considering knowledge ontology. To achieve stability in knowledge ontology, both dedicated business personnel and proactive IT personnel are required. Business personnel should focus on object ontology and recognize the object's life cycle, while proactive IT personnel aim to clarify the object's status. However, in common situations, business process steps are often used as the statuses of the object ontology, resulting in continuous adjustments to the core ontology when business changes, which undermines stability. Incompleteness can arise from infrequent situations or insufficient data quality. These situations provide opportunities to examine the essentiality, interpretability, and stability of the knowledge ontology, but are often treated as special cases. The incompleteness characteristic is often unnoticed due to the lack of awareness, interdisciplinary experts, or proper tools, which this research aims to address.

A concrete methodology for supporting domain knowledge integration is presented through the proposal of four OOO characteristics and a CAS model workflow, which highlights the significance of information sensitivity among business personnel and proactive IT departments (Wessel et al., 2021; Wolf et al., 2018).

4. Modeling issues for domain knowledge integration

This section illustrates how the domain knowledge integration framework be applied with the OOO concepts and UML-based CAS model. The modeling issues are listed in Table 2 and introduced in the following subsections.

Table 2. The modeling issues for domain knowledge integration.

Issues	Phase	Related OOO characteristics	Principles/rules	Components in the CAS model
Class type	Phase 1	Essentiality	Principle 1-1	Class
			Principle 1-2	
Data key and business key	Phase 2	Essentiality	Principle 1-2	Class
			Rule 1	
Multiplicity and conjunctional pattern	Phase 2	Essentiality	Rule 1	Class
		Incompleteness	Rule 2	
Workflow level and object status	Phase 3	Interpretability	Principle 2b	Activity
		Stability	Principle 3	Status
		Incompleteness	Principle 4	
			Rule 3	
			Rule 4	
Multi-status-attribute class description	Phase 3	Essentiality	Principle 1-2	Class
		Stability	Rule 4	Status

4.1. Class type

CAS model classes are categorized as fundamental, conjunctional, or complementary, with the first two being essential and the last one being non-essential. Fundamental classes represent actual objects and important concepts and account for most of them. Conjunctional classes define relationships between fundamental classes, while complementary classes are used for staging data for data analysis, which doesn't affect the integrity of the CAS model.

4.2. Data key and business key

CAS model classes have data keys and business keys. Data keys are UIDs based on the Principle 1-2 and represent the object space's only existence. Business keys are identification codes given by business logic, but not all classes have them. Fundamental and complementary classes usually have business keys, while conjunction classes connect fundamental classes and should not have business keys.

4.3. Multiplicity and conjunctional pattern

Only one type of multiplicity between classes, the n-to-many-conjunctional pattern, is available in the CAS model. The association is recorded in the data key of the associated class. The multiplicity of the two classes explains the following. When *ClassA* needs to correspond to *ClassB*, through the directed association starting from *ClassA* to *ClassB*, the multiplicity of the end with the arrow is 1. Assuming that *ClassA* is defined as a purchase contract, and *ClassB* is defined as a supplier, each purchase contract will correspond to a supplier, so the attribute *bUid* on *ClassA* records the corresponding supplier, which is shown in Figure 3a. Consider a scenario in which Class A is still a purchase agreement but Class B is a company. The multiplicity of the directed association near the endpoint of the arrow changes to 2 at this point due to the presence of data key attributes corresponding to the buyer and seller on Class A, as shown in Figure 3b. Alternatively, if Class B is defined as a purchase transaction, the purchase contract becomes a component of the purchase transaction, the directed association's association is changed to composition, and the multiplicity close to the figure's endpoint remains 1 (see Figure 3c). According to Rule 1 in static structural modeling, as long as one end of the multiplicity is limited, the nature of the business can be flexibly interpreted in a variety of ways while maintaining the stability of the ontology.

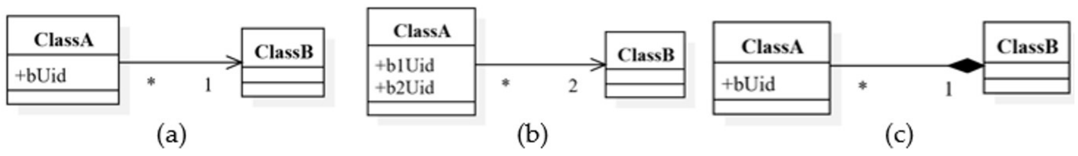

Figure 3. n-* conjunctional pattern.

4.3.1. Rule 2 – Action 1:adopt the conjunctional class

When there is a many-to-many relationship between *ClassA* and *ClassB*, there must be an attribute on *ClassA* to record multiple data keys (*uid*) of *ClassB*, and vice-versa (see Figure 4a). The attribute for the association must use an encoding principle to convert the multiple pieces of data into strings or a particular format, such as JSON, when designing the database structure directly with the attribute type on the category, to contain multiple pieces of data. JSON is a widely used data storage format for NoSQL databases that is extensible and readable and is based on plain text to store structured data. Nevertheless, the purpose of OOO is to construct a model that can be understood and implemented by both business and IT department personnel. If JSON is used in the description, the domain knowledge ontology cannot be clearly expressed, which leads to the incompleteness of OOO. To clearly express the relationship between *ClassA* and *ClassB*, it must be connected through a conjunctional class (see Figure 4b) that records the data keys of *ClassA* and *ClassB* on *ClassC* and connects the relationship between *ClassA* and *ClassB*.

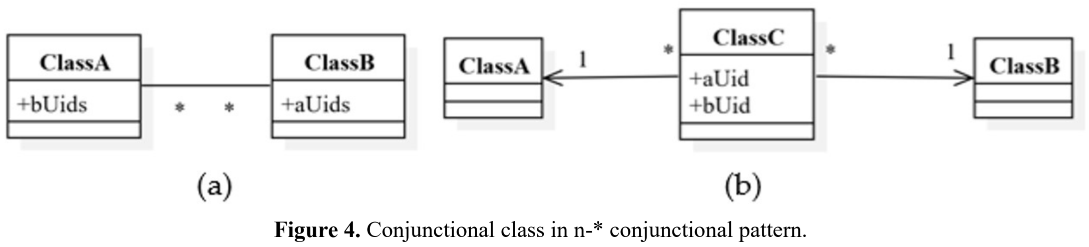

Figure 4. Conjunctional class in n-* conjunctional pattern.

Actual cases give examples. Two departments want to purchase 15 and 5 raw materials (*ClassDI*), respectively. The procurement department makes the purchase from two suppliers *X* and *Y* with an equal number of 10 items respectively (*ClassOI*). There is a many-to-many relationship between *ClassDI* and *ClassOI* as shown in Figure 5a. In this connection, the correspondence between required and purchased items is ambiguous. When supplier *X* delivers, it is unclear which department the 10 item corresponds to. To fully express the correspondence between them, it is necessary to construct a conjunctional class *ClassConj* that defines the assignment of items as shown in Figure 5b.

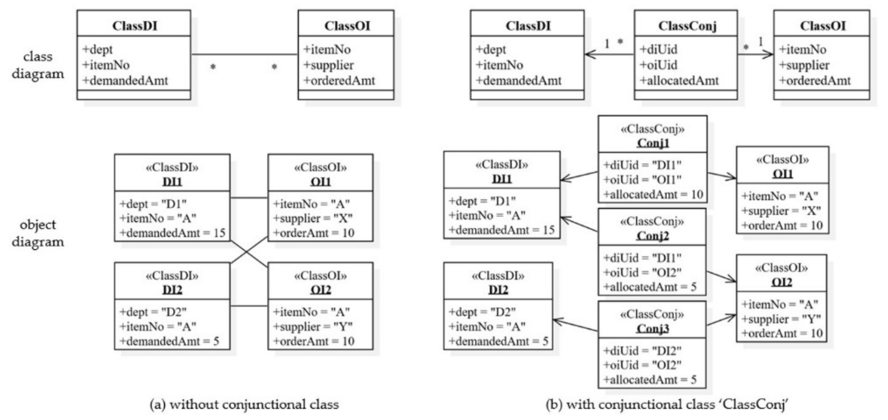

Figure 5. Example of a conjunctional pattern.

4.3.2. Rule 2 – Action 2:abandon the relationship

Abandoning the many-to-many relationship is suggested as an alternative approach. *ClassDI* and *ClassOI* belong to the fundamental class, while *ClassConj* belongs to the conjunctional class and deals with allocation between demand and purchased items. As allocation is subject to dynamic changes, *ClassConj* may be regarded as a complementary class and removed from OOO. This leads to abandoning the many-to-many association between *ClassDI* and *ClassOI*.

Whether the *ClassConj* should be regarded as a conjunctional class or a complementary class depends on whether the cognition of the relationship between the essence of the two objects is "real" or "sensual". Through the simple class diagram description, the domain knowledge ontology can be explored to help create a data model that is more in line with the business and facilitate the integration of domain knowledge.

4.4. Workflow step and object status

Recalling Rule 4, the workflow step is often directly used as the definition of "status" in modeling, which leads to the inability to maintain the stability of knowledge ontology and easily leads to incompleteness. A common example of the review process is illustrated below.

Considering a business operation with two-level review procedures, the activity diagram includes 4 activities: filling out the form, submitting, 1st review, and 2nd review. If the workflow steps are used as the statuses directly, the 4 statuses are: editing, 1st reviewing, 2nd reviewing, and approved (see Figure 6a). Recalling Principle 2b, "any information system developed according to the requirements cannot fully meet the requirements." If the behavior changes and the original two-level review is now changed to a three-level review, the original 4 statuses become deficient and a new "to be 3rd reviewed" status must be added and the transition between statuses must also be adjusted, as shown in Figure 6b.

To ensure the stability of OOO, the status should remain unchanged even when there are changes in dynamic behavior. This violates OOO's stability if the static structure has to be adjusted continuously. Rule 3 states that "an activity might contain no essence." By redefining the three statuses as editing, reviewing, and approved (see Figure 6c), stability and interpretability are achieved, and the "reviewing" status is not affected by dynamic changes.

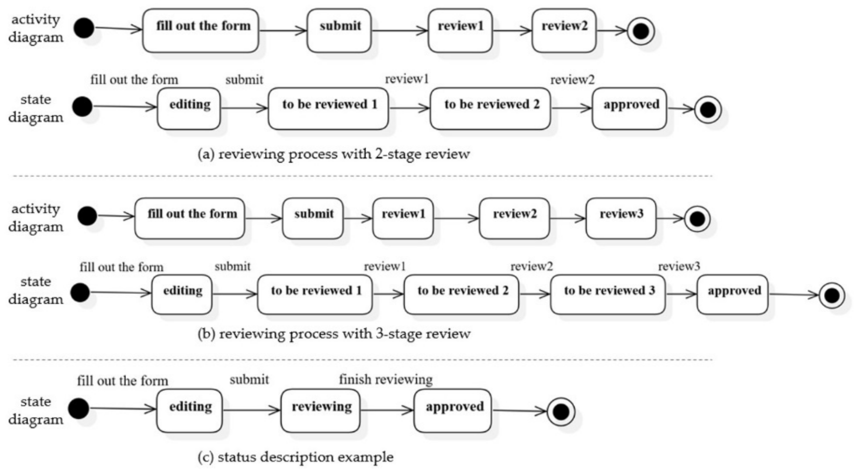

Figure 6. Status: a reviewing process example.

4.5. Multi-status-attribute class description

OOO classes may have multiple status definitions, but in workflow design, only one status definition is typically used. This can lead to a complex status network, but considering different essences from an object's viewpoint can simplify it. Multi-status-attribute class description reflects OOO essentiality and stability.

For example, a job needs to go through four operations A, B, C, and D. The order of these four operations is determined by the four types of products. The job order of Type W is expressed as A→B→C→D, the order of Type X is A→C→B→D, the one of Type Y is B→A→D→C, and the one of Type Z is A→B→D→C. The statuses corresponding to these four job sequences are S1, S2, S3, and S4. The job is described with an activity diagram and a state diagram in Figure 7a. Although there are four types of job flows, the order of job A and job C as well as the order of job B and job D is fixed. If using two statuses T and U instead, the activity and state diagram are replaced in Figure 7b), which is simplified. Consistent behaviors developed while working are by no means accidental and must encapsulate their industry. State S indicates that objects of this class have two distinct behavioral aspects that are fundamental to their nature as business objects. States T and U can be distinguished from state S. OOO aims to use simple diagrams as a shared language between business and information personnel. The multi-status-attribute class description principle is essential for expressing business domain knowledge ideas concisely.

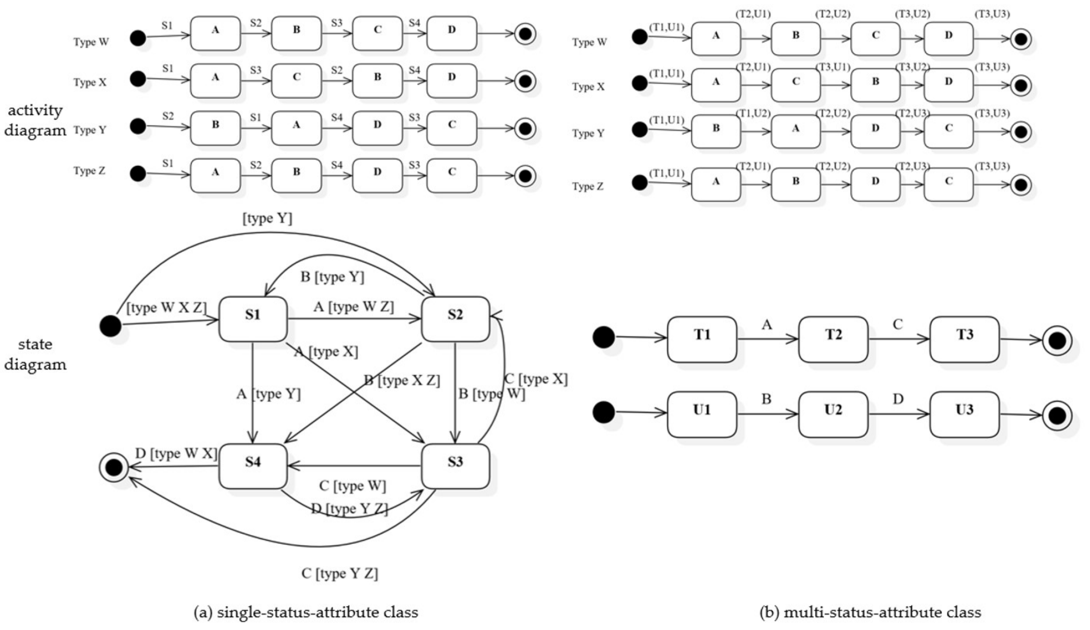

Figure 7. Multi-status-attribute class description example.

5. Theoretical knowledge ontology for MBOM domain

Product data or configuration management is the basis for smart manufacturing services that cater to diverse and highly customized product specifications while balancing enterprise-scale manufacturing and market customization. OOO's focus on the essence of objects aligns with Industry 4.0's digital thread and digital twin concepts. When integrated with MES and ERP systems, the digital thread allows for real-time adjustments and improved value streams, enabling business executives to assess and improve processes quickly (Pang et al., 2021). OOO requires comprehensive and cross-disciplinary collaboration for designing and maintaining complex systems, using the same standards for communication and consistent understanding of the business operation. This study demonstrates the implementation process of the proposed CAS model and the four characteristics of OOO using MBOM domain modeling in smart manufacturing services as an example.

5.1. Phase 1: knowledge acquisition

The first step in knowledge acquisition is defining the business process, categorizing tasks into object construction, business operation, and analysis, and marking clear inputs and outputs. Figure 8 illustrates this process. Product design is the first task, generating blueprints that must be transformed into a structured MBOM tree model. The configuration of the product is determined after acquisition planning is confirmed. If engineering changes occur, the process must start over. If production issues arise, it goes back to the acquisition planning task.

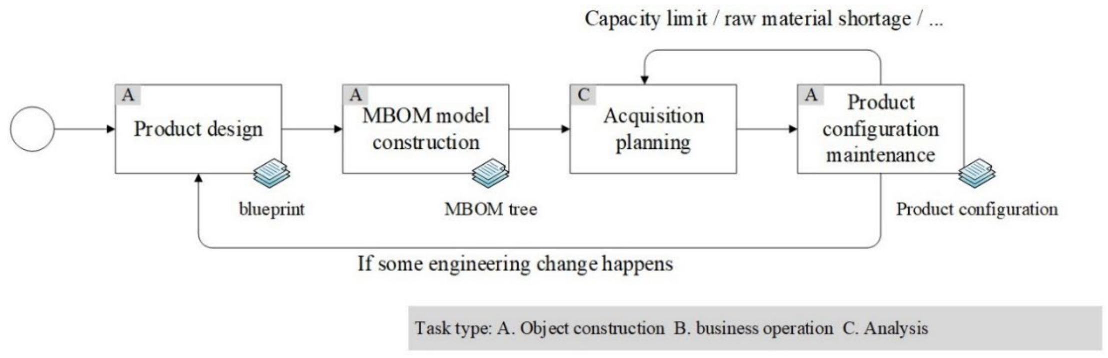
Capacity limit / raw material shortage / ...
A
Product design
A
MBOM model construction
C
Acquisition planning
A
Product configuration maintenance
blueprint
MBOM tree
Product configuration
If some engineering change happens
Task type: A. Object construction B. business operation C. Analysis

Figure 8. MBOM domain modeling flowchart.

Products have diverse specifications, which manufacturers may create to appeal to customers. Products are continuously developed and altered, resulting in different versions that require management. MBOM configurations have repeated structures that should not be managed repeatedly. The MBOM is a dynamic decision-making process despite being a static structure. Initially planning for a product's self-production may change if the company decides to outsource or purchase ready-made modules due to factors such as insufficient production capacity or schedule changes. This can alter the MBOM configuration and increase operational complexity.

5.2. Phase 2: static structural modeling

Phase 2 involves constructing a class diagram for static structural modeling after understanding business operations.

5.2.1. Class diagram

Objects are clarified as sheets, entities, and concepts, and then abstracted to corresponding classes as part of the static structure modeling process. Table 3 shows the defined classes that match the MBOM business process description.

Table 3. Class definitions in the MBOM domain model.

Class	Description	Object type	Class type	Real / Sensual
Part	Part is the main raw materials and components directly put into production operations, as well as materials required to cooperate with the production operation process.	Entity	Fundamental	Real
PartType	To facilitate identification, improve management efficiency, or serve as a basis for data analysis, *Parts* can be classified into different types.	Concept	Fundamental	Sensual
Acquisition	The acquisition is an abstract class that defines how *Parts* are acquired.	Concept	Fundamental	Real
Self-Producing	*Self-Producing* is a sub-class of *Acquisition* when the *Part* is entirely self-produced.	Concept	Fundamental	Sensual
Outsourcing	*Outsourcing* is a sub-class of *Acquisition* when *Part* is outsourced, which means that the manufacturer still has to provide some raw materials or go through some operation processes first while other ones are outsourced.	Concept	Fundamental	Sensual
Purchasing	*Purchasing* is a sub-class of *Acquisition* when the *Part* is obtained by direct purchase. Compared with *Outsourcing*, the manufacturer does not provide raw materials or go through any operation procedure. The only existing process is the inspection of incoming Parts.	Concept	Fundamental	Sensual
RoutingStep	*RoutingStep* is the required routing step for each *Acquisition*.	Concept	Fundamental	Real
RoutingStep-Part	The child *Parts* are marked when being used for the first time in the *RoutingStep*. The class is used to mark the *Parts* that must be prepared when the *Acquisition* proceeds to this *RoutingStep*, not to list all the Parts related to each *RoutingStep*.	Concept	Fundamental	Real
RoutingStep-Process	It is the operation processes corresponding to each *RoutingStep*.	Concept	Fundamental	Real
Part Configuration	A product series may contain multiple specifications, each of which is recorded as a *PartConfiguration*.	Concept	Fundamental	Real
Part ConfigurationConj	A conjunctional class records the relations of all *Acquisitions* used in each *PartConfiguration*, which is the key to being able to support the recursive structure of the MBOM tree.	Concept	Conjunctional	Real

This paragraph discusses the different object types and classes defined for MBOM modeling. Blueprints are the only object type belonging to the sheet entity. The majority of classes in Table 3 fall under the basic category and are related to business operations. *PartConfigurationConj* is a conjunctional class that adds flexibility but is harder to detect.

The OOO vocabulary includes real and sensual objects, with the latter being more flexible and easier to handle when changes occur. The ability to distinguish between them is important for BITA during digital transformation. Only the *PartType* enumeration and the subclasses of Acquisition are sensual classes, which can bring richer interpretability and help the enterprise domain knowledge model evolve with changing business behaviors. Changing real classes, however, can lead to instability in the knowledge model. It's best to consider sensual objects first, and it's often unintentional if real objects are changed due to a lack of understanding of the knowledge model. This aspect is not easily detected in general business process analysis. Once classes are defined, the relationship between them can be clarified with a UML class diagram as shown in Figure 9.

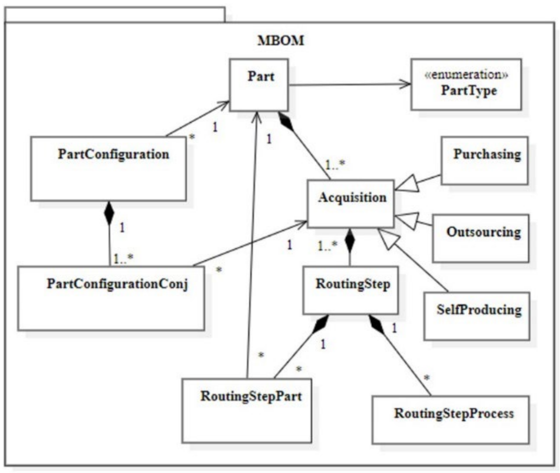

Figure 9. MBOM domain model – a UML class diagram.

- *Part* vs. *PartType*: Each Part can be defined as a different *PartType* according to the practical management purpose or data analysis purpose. *PartType* is represented by enumeration.
- *Part* vs. *Acquisition: Parts* have various acquisition methods, such as self-producing, outsourcing, and purchasing, which are represented as subclasses of the *Acquisition* class. The one-to-many multiplicity between *Part* and *Acquisition* is clear, and they share the same life cycle. The composition relationship is used to express the tight coupling between *Part* and *Acquisition*.
- *Acquisition* vs. *RoutingStep: Acquisition* and *RoutingStep* have a one-to-many multiplicity since each *Acquisition* needs a set of *RoutingSteps* with the same life cycle, and their relationship is expressed using composition.
- *RoutingStep* vs. *RoutingStepPart*: When each *RoutingStep* is executed, there may be a list of the number of raw materials that need to be prepared. Therefore, the one-to-many multiplicity and composition relationship between *RoutingStep* and *RoutingStepPart* are both clear.
- *RoutingStepPart* vs. *Part: RoutingStepPart* expresses child *Parts* required by *RoutingStep*. Each *RoutingStepPart* assigns only one *Part*, identified by the business key "part number attribute." Each *Part* may be referred to by several *RoutingStepParts*, resulting in a many-to-one direct association in the class diagram.
- *RoutingStep* vs. *RoutingStepProcess*: The execution of each *RoutingStep* includes some manufacturing procedures, such as turning, milling, surface treatment, heat treatment, etc., each of which corresponds to each *RoutingStepProcess*. Therefore, the one-to-many multiplicity and composition relationship between *RoutingStep* and *RoutingStepProcess* are both clear.
- *Part* vs. *PartConfiguration*: The unique specification identifies each *Part*, but there may be different acquisition methods related to different child parts or processes. *PartConfiguration* serves as a summary of the product's revision history, and the association between *Part* and *PartConfiguration* is used to express the loose coupling relationship.
- *PartConfigurationConj: PartConfigurationConj* is a standard conjunctional pattern, linking *PartConfiguration* with *Acquisition* through one-to-many multiplicity. It simplifies the many-to-many multiplicity without repetition and is expressed through tight coupling composition with *PartConfiguration* and loose coupling association with *Acquisition*.

5.2.2. Interpretability

Figure 10 is an example for illustrating the interpretability of the proposed MBOM CAS domain model. A *PartConfiguration-α*, shown in Figure 10a, is composed of *Part-B* and *Part-C*, which are assigned as the corresponding

RoutingStepParts. And *Part-D* and *Part-E* are assigned as the *RoutingStepParts* of the self-producing acquisition of *PartConfiguration-α* in *Part-B*, whose acquisition types are purchasing since they are both raw materials in the *PartConfiguration-α*, namely *Acquisition-d* and *Acquisition-e*, respectively. The *RoutingStepParts* required for the self-producing acquisition *Acquisition-*c_1 of *Part-C* are *Part-F* and *Part-G*, which are also raw materials corresponding to purchasing acquisition *Acquisition-f* and *Acquisition-g*. When the capacity required by *Acquisition-*c_1 is insufficient or the acquisition for *Part-C* changes from self-producing to outsourcing, *PartConfiguration-β* appears as shown in Figure 10b.

From the perspective of the structure between *Parts*, *PartConfiguration-α* and *PartConfiguration-β* are the same, and the difference between them is the *Acquisition* of *Part-C*. The former is *Acquisition-*c_1 and the latter is *Acquisition-*c_2. According to Table 4, among the four routing *RoutingSteps* of *Acquisition-c1*, P_2 and P_3, corresponding *to* C_1*-*RS_2 and C_1*-*RS_3, disappeared in *Acquisition-*c_2 due to outsourcing. When the outsourcing product returns, the *Acquisition-*c_2 is completed via P_4 (inspection). When the inventory of *Part-F* and *Part-G* gradually runs out, *Part-C* is changed to direct purchasing, and *PartConfiguration-γ* appears. In *PartConfiguration-γ*, *Part-C* is acquired by purchasing *Acquisition-*c_3, in which *Part-F* and *Part-G* are no longer required, as shown in Figure 10c. The recursive tree structure of *Parts* is typically the only way that the MBOM structure is conceptualized in the literature. There is only the class *Part* and a composition pointing to the *Part* itself at both ends when shown as a UML class diagram. The data for connecting the relationships between *PartConfigurations* and *Acquisitions* in Figure 10 is shown in Table 5 as *PartConfigurationConj*.

Table 4. Acquisitions in the MBOM domain modeling example.

Acquisition	Routing step	Process	Supplied part
c_1	RS_1	P_1	F、G
c_1	RS_2	P_2	
c1	RS_3	P_3	
c_1	RS_4	P_4 (Inspection)	
c_2	RS_1	P_1	F、G
c_2	RS_4	P_4 (Inspection)	
c_3	RS_4	P_4 (Inspection)	

Table 5. Part configuration conjunctions in the MBOM domain modeling example.

Conj. data index	Part configuration (corresponding part)	Acquisition (corresponding part)
1	α (A)	a (A)
2	α (A)	b (B)
3	α (A)	c1 (C)
4	α (A)	d (D)
5	α (A)	e (E)
6	α (A)	f (F)
7	α (A)	g (G)
8	β (A)	a (A)
9	β (A)	b (B)
10	β (A)	C_2 (C)
11	β (A)	d (D)
12	β (A)	e (E)
13	β (A)	f (F)
14	β (A)	g (G)
15	γ (A)	a (A)
16	γ (A)	b (B)
17	γ (A)	c_3 (C)
18	γ (A)	d (D)
19	γ (A)	e (E)

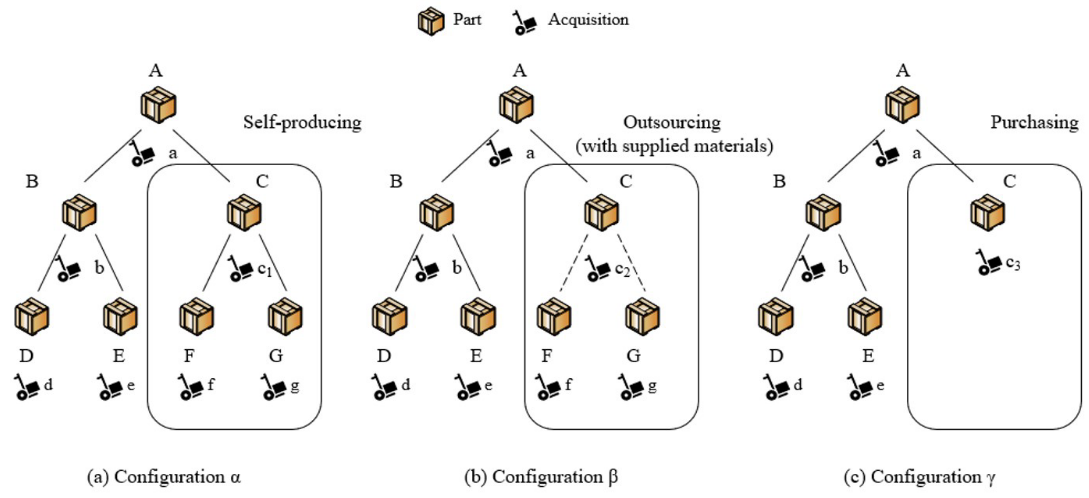

Figure 10. Product configurations in the MBOM domain modeling example.

If *PartConfiguration* is defined as an instance of class X, and *PartDto* is defined as a complementary class, then function F can be used to obtain Y. (X). The relaxation process is shown in Figure 11 to demonstrate compatibility when the *PartConfiguration* is provided. First, *PartDto* focuses on the relationship between *Parts*. Therefore, *PartType*, the three subclasses of *Acquisition*, and *RoutingStepProcess* are omitted since they are not directly related to the multiplicity of *Parts*, shown in Figure 11a. Next, since each *PartConfigurationConj* can be assigned to a unique *Acquisition*, the simplified one-to-many multiplicity between *PartConfiguration* and *Acquisition* is held, as shown in Figure 11b. *PartConfigurationDto* can be omitted into *PartDto* because each *PartConfiguration* directly corresponds to a specific *Part* and is provided. As a result, the model in Figure 11c is created. Finally, after simplifying the one-to-many multiplicity between *Part* and *Acquisition*, the one between *Acquisition* and *RoutingStep*, and the one between *RoutingStep* and *RoutingStepPart* and merging the *RoutingStepPart* into *PartDto*, the class diagram of Figure 11d appears. Thus, the essentiality, interpretability, and stability of the proposed class model are verified.

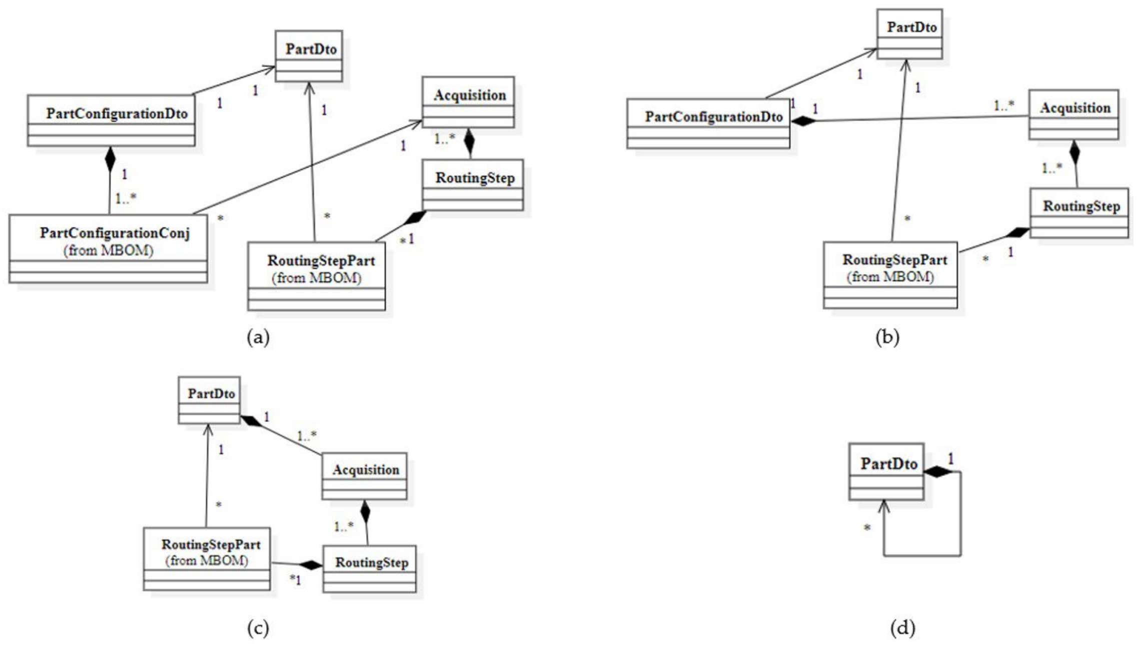

Figure 11. The relaxation process of MBOM domain modeling.

5.3. Phase 3: dynamic behavioral modeling

The class for identifying a product configuration version must have a status value to indicate whether it has been released or is still being edited. The activity and state diagrams in Figure 12 show the sequence of events and perspective of a single object. The life cycle of a product configuration is finished at finalization. When the

MBOM needs to be changed, a new product configuration is established based on the current part configuration and each change and finalization relates to a single object. The model may appear simple, but it can fully describe the history of all product configurations.

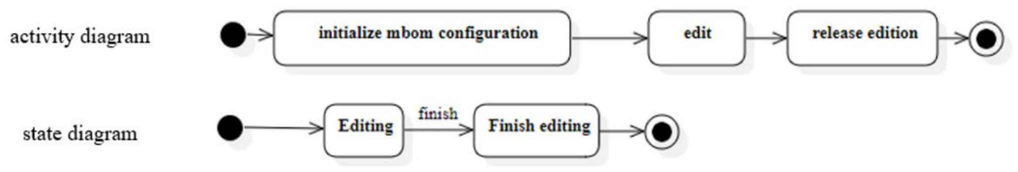

Figure 12. The activity diagram and state diagram of a part configuration in the MBOM domain modeling example.

Figure 13's activity and state diagrams might be misleading to users as they contain the phrase "engineering change." However, the activity diagram is not complete as it suggests that the edition is continually updated and released after each engineering change. This diagram was created with the intention of representing the object's entire life cycle, but it is not appropriate.

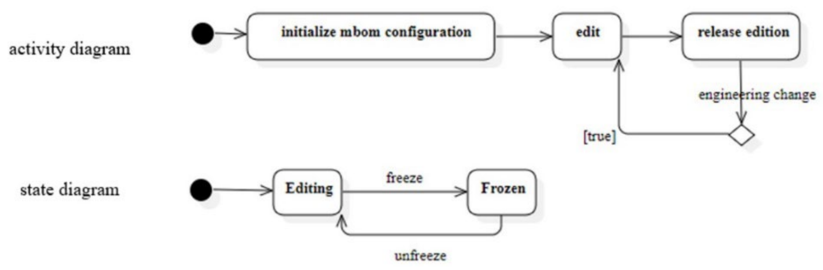

Figure 13. The activity diagram and state diagram of a part configuration in the MBOM domain modeling example (an inappropriate version).

The "finish editing" in Figure 12 becomes "frozen" in Figure 13, but "finished" and "frozen" have different meanings. Frozen is temporary and can be unfrozen at any time, making it difficult to define the version of the configuration. A state diagram describes the state changes of an object, but if the configuration can change continuously, it is impossible to manage the version. Therefore, the version cannot be used as an attribute of the part configuration in the class diagram, and the incompleteness of the model is revealed.

6. Discussion

The proposed method is compared with related studies in terms of subject, objective, perspective, and view, as shown in Table 6. IPA aims to discuss the convergence of AI, automation, and customer data. The process-oriented method based on BPM and through BPMN as the main view is the main challenge of automation due to the diversity of processes and a large amount of data. Therefore, emerging researches introduce AI methods, such as machine learning and NLP chatbot technology (Chakraborti et al., 2020). However, the development bottleneck of NLP chatbots has shifted from system development to the establishment of an in-depth domain knowledge base (Chao et al., 2021). How much investment in AI and other digital technologies can be directly converted into substantial progress in the digital transformation of enterprises remains to be confirmed (Kane et al., 2015).

Table 6. The proposed method compared with related studies.

	Proposed method	**Park & Van Der Aalst (2021)**	**Chakraborti et al. (2020)**
Subject	DTO	DTO	IPA
Objective	Clarify the domain knowledge ontology of the enterprise	Building an organizational digital twin	The convergence of AI, automation, and the customer data
Perspective	Object-essence-oriented	Action-oriented process mining	Process-oriented
View	Basic UML diagrams: class diagram, activity diagram, and activity diagram	Digital Twin Interface Model (DT-IM)	Business process model and notation

Park & Van Der Aalst (2021) proposed an action-oriented process mining method to construct a DTO, which mainly relies on the system architecture. The main challenge in implementing a DTO is technical debt, which should not outweigh the generative possibilities of the digital twin's construction (Parmar et al., 2020). Therefore, while this research refers to the concept of DTO, clarifying the enterprise's domain knowledge ontology is the primary task. The objective is similar to that of DTO, while the OOO focuses on object cognition of the domain knowledge ontology instead of exhaustive modeling. The proposed method emphasizes the "static structure" of the business concept, with "interpretability" and "stability" used to verify dynamic behaviors. In short, the OOO spirit is to "see the same (model) and realize from each own specialized perspective.

The proposed method is more suitable for large-scale and long-established enterprises that face challenges with digital transformation due to accumulated "knowledge debt" and "architecture debt." The OOO characteristics and UML-based CAS model provide clear guidelines, starting from single-point implementation and gradually constructing a series. The methodology is universal and can construct a small-scale knowledge ontology by clarifying essential objects and their multiplicity. The deepening of OOO characteristics requires interdisciplinary participation and proactive IT personnel. Implementing from a small-scale single point makes it difficult to show comprehensive results in the early stage, which is the main challenge of this method.

7. Conclusions

This study uses OOO to assess business activities and identify discrepancies between knowledge and reality. It has important implications for domain knowledge modeling and computer science and data science design patterns. The proposed CAS model applied in the MBOM domain modeling case supports knowledge integration and reflects enterprise reality. This section discusses theoretical contributions, managerial implications, novelty and transdisciplinarity, and limitations and future research.

7.1. Research contributions

This study proposes three contributions for designing and analyzing knowledge-integrated information systems for smart manufacturing services. Firstly, it suggests four OOO characteristics for domain knowledge ontology: essentiality, interpretability, stability, and incompleteness. Secondly, it proposes a UML-based CAS model with simple annotations as a communication tool. This keeps communication straightforward and allows for the full expression of the diverse nature of enterprise reality with a few fundamental principles. Thirdly, the proposed model construction workflow is confirmed through a case study of MBOM domain modeling, implementing the three-phase CAS model workflow as a communicating basis for BITA in digital transformation. The study demonstrates the remarkable capacity of the CAS model workflow and the interaction between the four OOO characteristics and the CAS model for knowledge integration.

7.2. Managerial implications

Clarifying enterprise domain knowledge and improving communication among employees are essential for digital transformation and achieving BITA (Holotiuk & Beimborn, 2017). To integrate heterogeneous sources, a careful plan is needed for the data extraction, transformation, and analysis process (Blazquez & Domenech, 2018). This requires defining a semantic model of data, specifying domain knowledge, and defining links between different types of semantic knowledge (Munir & Sheraz Anjum, 2018). The challenges to achieving BITA include a common language, role repositioning, and customer-oriented value chain recognition (Hsu et al., 2018; Tijan et al., 2021).

7.3. Research novelty and transdisciplinarity

By integrating the four OOO characteristics and the CAS model, this study proposes a novel communication language for describing enterprise domain knowledge that applies to managing the BITA issue in the digital transformation agenda. The study showcases transdisciplinary integration across philosophy, computer science, knowledge engineering, and management, resulting in a novel approach.

7.4. Limitations and future research

The OOO concepts and CAS model proposed here tackle the problem of integrating enterprise domain knowledge during digital transformation. While other knowledge architectures and models exist in literature, their adaptability to all enterprises cannot be guaranteed. The lack of criteria to evaluate the BITA degree is a limitation of this study, and future research should focus on driving transformation and devising strategies using the OOO and CAS models.

8. Acknowledgements

This research is partially supported by the National Science and Technology Council (Taiwan) individual research grants (Grant numbers: MOST-110-2221-E-007-113-MY3 and MOST-111-2221-E-007-050-MY3).

9. References

Al Tal, S., Al Salaimeh, S., & Hajiyev, N. (2020). Information technology in business process reengineering. *Información Tecnológica*, *29*(7), 3653-3657. Retrieved in 2022, November 19, from https://www.researchgate.net/publication/342283185_Information_Technology_In_Business_Process_Reengineering

Baiyere, A., Salmela, H., & Tapanainen, T. (2020). Digital transformation and the new logics of business process management. *European Journal of Information Systems*, *29*(3), 238-259.

Bao, J., Guo, D., Li, J., & Zhang, J. (2019). The modelling and operations for the digital twin in the context of manufacturing. *Enterprise Information Systems*, *13*(4), 534-556.

Blazquez, D., & Domenech, J. (2018). Big Data sources and methods for social and economic analyses. *Technological Forecasting and Social Change*, *130*, 99-113. http://dx.doi.org/10.1016/j.techfore.2017.07.027.

Buriánek, J. (2021). *Comparison of selected enterprise architecture modeling techniques from the perspective of IT services*. Vienna: CEUR-WS.

Chakraborti, T., Isahagian, V., Khalaf, R., Khazaeni, Y., Muthusamy, V., Rizk, Y., & Unuvar, M. (2020). From robotic process automation to intelligent process automation: emerging trends. In *Business Process Management: Blockchain and Robotic Process Automation Forum: BPM 2020 Blockchain and RPA Forum*. Seville.

Chao, M.-H., Trappey, A. J., & Wu, C.-T. (2021). Emerging technologies of natural language-enabled chatbots: a review and trend forecast using intelligent ontology extraction and patent analytics. *Complexity*, *2021*, 1-26.

Chen, J., Wang, L., & Qu, G. (2021). Explicating the business model from a knowledge-based view: nature, structure, imitability and competitive advantage erosion. *Journal of Knowledge Management*, *25*(1), 23-47.

Cupek, R., Ziebinski, A., Drewniak, M., & Fojcik, M. (2019). Knowledge integration via the fusion of the data models used in automotive production systems. *Enterprise Information Systems*, *13*(7-8), 1094-1119.

Delgado, J. M. D., & Oyedele, L. (2021). Digital Twins for the built environment: learning from conceptual and process models in manufacturing. *Advanced Engineering Informatics*, *49*, 101332.

Dinata, H. (2020). Business process reengineering: the role of information technology as a determinant of success for improving performance. *Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi*, *5*(1), 25-31. Retrieved in 2022, November 19, from http://repository.ubaya.ac.id/id/eprint/37796

Feio, I. C. L., & Santos, V. D. (2022). A strategic model and framework for intelligent process automation. In *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*.

Ferro, F. (2019). Object-oriented ontology's view of relations: a phenomenological critique. *Open Philosophy*, *2*(1), 566-581.

Gong, Y., & Janssen, M. (2019). The value of and myths about enterprise architecture. *International Journal of Information Management*, *46*, 1-9.

Haleem, A., Javaid, M., Singh, R. P., Rab, S., & Suman, R. (2021). Hyperautomation for the enhancement of automation in industries. *Sensors International*, *2*, 100124.

Hammer, M., & Champy, J. (2009). *Reengineering the corporation: manifesto for business revolution, A*. Michigan: Zondervan.

Harman, G. (2018). *Object-oriented ontology: a new theory of everything*. London: Penguin UK.

Holotiuk, F., & Beimborn, D. (2017). Critical success factors of digital business strategy. In *Proceedings der 13. Intern. Tagung Wirtschaftsinformatik*. St. Gallen

Hsu, C.-C., Tsaih, R.-H., & Yen, D. C. (2018). The evolving role of IT departments in digital transformation. *Sustainability*, *10*(10), 3706.

Jonathan, G. M., Hailemariam, K. S., & Debay, W. L. (2019). Business-IT alignment in the banking sector: a case from a developing country. In *The 13th Mediterranean Conference on Information Systems (MCIS)*. Naples, Italy.

Kane, G. C., Palmer, D., Phillips, A. N., Kiron, D., & Buckley, N. (2015, July 14). Strategy, not technology, drives digital transformation. *MIT Sloan Management Review*.

Kirchmer, M., & Franz, P. (2020). Process reference models: accelerator for digital transformation. In *International Symposium on Business Modeling and Software Design*. Bulgaria.

Kobryn, C. (1998, November). Modeling enterprise software architectures using UML. In *Proceedings of the Second International Enterprise Distributed Object Computing* (Cat. No.98EX244). IEEE.

Kotusev, S. (2021). *A comparison of the top four enterprise architecture frameworks.* Retrieved in 2022, November 19, from https://www.bcs.org/articles-opinion-and-research/a-comparison-of-the-top-four-enterprise-architecture-frameworks/

Li, B., Schultz, C., Teizer, J., Golovina, O., & Melzner, J. (2022). Towards a unifying domain model of construction safety, health and well-being: SafeConDM. *Advanced Engineering Informatics*, *51*, 101487.

Liu, S.-F., Fan, Y.-J., Luh, D.-B., & Teng, P.-S. (2022). Organizational Culture: the key to improving service management in Industry 4.0. *Applied Sciences*, *12*(1), 437.

Malaurent, J., & Karanasios, S. (2020). Learning from workaround practices: the challenge of enterprise system implementations in multinational corporations. *Information Systems Journal*, *30*(4), 639-663.

May, M. C., Kiefer, L., Kuhnle, A., & Lanza, G. (2022). Ontology-based production simulation with OntologySim. *Applied Sciences*, *12*(3), 1608.

McCormack, D. P. (2018). *Atmospheric things*. Duke University Press.

Mongale, A., Kekwaletswe, R. M., & Mogoale, P. D. (2021). A framework for the alignment of information technology and business in South African Development Banks. *International Journal of Innovative Science and Research Technology*, *6*(10), 92-98.

Munir, K., & Sheraz Anjum, M. (2018). The use of ontologies for effective knowledge modelling and information retrieval. *Applied Computing and Informatics*, *14*(2), 116-126. http://dx.doi.org/10.1016/j.aci.2017.07.003.

Ozkaya, M., & Erata, F. (2020). A survey on the practical use of UML for different software architecture viewpoints. *Information and Software Technology*, *121*, 106275.

Panetta, K. (2019). Hyperautomation, blockchain, AI security, distributed cloud and autonomous things drive disruption and create opportunities in this year's strategic technology trends. Retrieved in 2022, November 19, from https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2020

Pang, T. Y., Pelaez Restrepo, J. D., Cheng, C.-T., Yasin, A., Lim, H., & Miletic, M. (2021). Developing a digital twin and digital thread framework for an 'Industry 4.0'Shipyard. *Applied Sciences*, *11*(3), 1097.

Park, G., & Van Der Aalst, W. M. (2021). Realizing a digital twin of an organization using action-oriented process mining. In *2021 3rd International Conference on Process Mining (ICPM)*. The Netherlands.

Parmar, R., Leiponen, A., & Thomas, L. D. (2020). Building an organizational digital twin. *Business Horizons*, *63*(6), 725-736.

Ramesh, N. (2019). *Digital transformation: how to beat the high failure rate* (Doctoral dissertation). Oklahoma State University

Recht, S., & Wiberg, N. (2018). Quantified manufacturing, little data and the new æsthetic. *CSPA Quarterly*, (21), 8-19.

Sahal, R., Alsamhi, S. H., Breslin, J. G., Brown, K. N., & Ali, M. I. (2021). Digital twins collaboration for automatic erratic operational data detection in industry 4.0. *Applied Sciences*, *11*(7), 3186.

Schallmo, D. R., & Williams, C. A. (2018). History of digital transformation. In D. R. A. Schallmo & C. A. Williams (Eds.), *Digital transformation now!* (pp. 3-8). Springer.

Schmidt, J., Peruzzini, M., Paganin, L. B. Z., Sato, R. S., & Borsato, M. (2020). Expert system based on ontological model to support the detailed design of agricultural machinery: a case of hydraulic hoses. *Product: Management & Development*, *18*(1), 53-69.

Schneider, S., & Kokshagina, O. (2021). Digital transformation: what we have learned (thus far) and what is next. *Creativity and Innovation Management*, *30*(2), 384-411.

Sebastian, I., Ross, J., Beath, C., Mocker, M., Moloney, K., & Fonstad, N. (2017). *How big old companies navigate digital transformation*. New York: Routledge.

Siau, K., & Loo, P.-P. (2006). Identifying difficulties in learning UML. *Information Systems Management*, *23*(3), 43-51.

Tijan, E., Jović, M., Aksentijević, S., & Pucihar, A. (2021). Digital transformation in the maritime transport sector. *Technological Forecasting and Social Change*, *170*, 120879.

Venkatesan, D., & Sridhar, S. (2019). A rationale for the choice of enterprise architecture method and software technology in a software driven enterprise. *International Journal of Business Information Systems*, *32*(3), 272-311.

Venkatraman, S., & Venkatraman, R. (2019). Process innovation and improvement using business object-oriented process modelling (BOOPM) framework. *Applied System Innovation*, *2*(3), 23.

Wessel, L., Baiyere, A., Ologeanu-Taddei, R., Cha, J., & Blegind-Jensen, T. (2021). Unpacking the difference between digital transformation and IT-enabled organizational transformation. *Journal of the Association for Information Systems*, *22*(1), 102-129.

Wilson Júnior, E., Farias, K., & Silva, B. (2021). A survey on the use of UML in the Brazilian industry. In *Brazilian Symposium on Software Engineering* (pp. 275-284). Association for Computing Machinery.

Wohlrab, R., Horkoff, J., Kasauli, R., Maro, S., Steghöfer, J.-P., & Knauss, E. (2020). Modeling and analysis of boundary objects and methodological islands in large-scale systems development. In *International Conference on Conceptual Modeling*. Vienna.

Wolf, M., Semm, A., & Erfurth, C. (2018). Digital transformation in companies–challenges and success factors. In *International Conference on Innovations for Community Services*. Delft, The Netherlands.

Yoran, G. (2018). Applied metaphysics–objects in object-oriented ontology and object-oriented programming. *Interface Critique*, (1), 120-133.

Yu, H., Ogbeyemi, A., Lin, W., He, J., Sun, W., & Zhang, W. (2023). A semantic model for enterprise application integration in the era of data explosion and globalisation. *Enterprise Information Systems*, *17*(4), 1989495.