Ontology-Based Business Processes Gap Analysis

Abdelgaffar Hamed Ahmed Ali
Department of Computer Information Systems, King Faisal University, Al-hasa, Saudi Arabia

***Abstract*—Business processes are subject to change for quality reasons (i.e., efficiency). However, the gap analysis process is a preliminary and essential step in discovering the gap between the to-be and as-is business processes. It usually resorts to a nonstandard and manual analysis process, making it unpredictable and complex. This paper proposes a standard method based on ontology principles and the business process design methodology (DEMO). The ontology unifies the shared vocabulary among worlds of source and target business process to enable this sort of interoperability. Building an essential model is a core concept behind DEMO that provides an ontological view independent of realization and implementation issues and enables understanding of the enterprises' behavior. Moreover, this paper provides heuristics for detecting gaps, based on the premise that producing similar institutional facts reflects similar behavior between the to-be and as-is business processes. Since the domains of the source and target are the same, it is also possible to compare the inputs of corresponding actions. The paper proposes a UML activity model for modeling business processes, enriched with DEMO concepts, to provide a foundational and informative ontology for reasoning about gaps. The expected outcome is a contribution to the broader community of business process management, ERP, and strategic planning, enabling more informed decision-making.**

***Keywords*—Business process; gap analysis; ontology for business processes**

I. Introduction

Enterprises use business processes to produce products and services for their stakeholders. These business processes are subject to change due to quality reasons such as adding efficiency or general business change requirements. Therefore, changing these business processes is a critical success factor for enterprises. There are hundreds or even thousands of business processes (BPs) within small and medium-sized enterprises (SMEs) and large organizations, ranging from simple tasks such as enrolling students in courses to complex ones like procurement and recruitment. These processes evolve over time to meet quality demands—becoming more cost-effective, responsive, and standardized. Introducing ERP to an organization is an example of this major change usually required to achieve some quality, such as effectiveness and efficiency, reducing costs by removing waste and redundancy. Therefore, it replaces legacy systems and business processes with standard, best practices, and new value-added business processes. The documentation of these business processes became of great value for enterprises to understand, analyze, monitor (i.e., bottleneck), re-engineer these processes, and generally seek high quality by proper management.

However, there is typically a gap between the legacy process (as-is process) and the new ERP processes (to-be process) that must be identified as a critical step before transformation occurs. This is because developers and strategists need to make informed decisions. Moreover, the issue becomes more pronounced when integrating at least two systems.

The main challenge lies in ensuring that the to-be process aligns with the organization's goals. Current practices are inefficient because they rely on manual inspections of specifications, models (dependent on experts' experience and knowledge), or artifacts to identify discrepancies. Additionally, there is no standardized process to serve as a baseline for evaluating differences and determining whether to replace or integrate a system.

On the other hand, existing literature has primarily focused on analyzing business processes in repositories for reuse purposes, identifying redundancies and variations [1] . More importantly, prior studies [2] address compliance between business processes, where one serves as an ideal reference model and the other represents current practices. While this is a prominent research area, it assumes the existence of business process instances in event logs. These efforts have led to various metrics and methods. However, the core question—whether process A (to-be) should replace process B (as-is) and why—remains unaddressed (gap analysis).

This paper tackles this question from a semantic-based perspective. Although some existing methods propose behavior-based or semi-semantic-based approaches, their focus has been either partial or limited to manipulating business process models at the implementation level.

This work introduces DEMO, a methodology that applies ontological discipline to enterprise engineering and design, independent of implementation and realization. DEMO enables a semantic and formal understanding of what enterprises actually do when performing business activities.

Ontology, as a discipline, addresses interoperability issues among information systems and agents. It establishes principles for enabling interoperability, such as explicit specifications of shared concepts in a common vocabulary. This ensures a unified understanding among agents, facilitating communication both within and outside organizations—for example, in e-commerce systems, supply chain exchanges, and other domains.

This work argues that integrating DEMO concepts into a business process model will enable reasoning about gaps in business processes, making automated semantic gap analysis possible. Furthermore, this research aims to provide a framework for automating business process gap analysis and related evaluations using ontological principles. The expected value lies in reducing the costs associated with manual alternatives—methods that are inefficient and do not scale well, particularly when dealing with large volumes of business processes.

The paper is organized as follows: Section I provides an introduction and background. Section II discusses the context of this work and explains the business process. Section III presents the ontology principles. Section IV explains DEMO concepts and its philosophy for designing business processes, while Section V reviews related literature. Section VI outlines the proposed methodology, which is evaluated using a case study discussed in Section VII. Finally, Section VIII offers interpretations and comments, and Section IX concludes the article.

II. Business Process Design and Reengineering

Business processes are the heart of organizations because it's the machinery providing the services or products. It is meaningful work performed end-to-end to create customer value across an enterprise [3]. They are usually tasks performed in order, either due to space or time to produce a specific outcome. Procurement, Recruitment, processing of purchase orders, Making visa approval, Getting a new passport, replenishing stock, product development etc., are all concrete examples of business processes. Practically it is observed that it is subject to change or redesign or generally re-engineering for several reasons, such as business, organizational, and technical, as well as the major aim to add some quality attributes (i.e., speed, economy, better service). For instance, a business can merge or acquire other business (s), leading to business and organizational structure change. For a few decades, the government had witnessed major changes to their citizen-provided services that necessarily involved reengineering business processes to add some quality attributes. Therefore, we can basically classify it into two reasons: functional (merge case) and non-functional aspects (government case). However, it turns out that, changing business processes is a critical, costed task and has a high failure rate. On the other hand, Business Process Management (BPM) is a discipline concerned with documenting, designing and redesigning, monitoring, and instrumenting business processes. Deming and Hammer have established the principles of BPM [3].

Business processes have been studied for about decades ago, and a famous key redesign attempt was proposed by Hammer [4]. The key concept Hammer came up with was the result; it is a primary or intrinsic element where business processes are secondary, which tries to achieve it even when changed or reformed to add some qualities. However, big organizations with hierarchy management layers have many people doing different tasks that usually involve activities across departments or units as well as organizational boundaries. Understanding and making sense of what is going on is where the concept of the business process comes in. It is worth bringing in Searle's theory [5] here which builds on speech act theory, to understand in some depth what the business process is actually doing. Seral argues that businesses are changing social reality by performing speech acts that have a memory (records), called institutional facts, which have meaning only under some context, i.e., background and framing rules. For example, the acceptance or rejection of this article is an institutional fact that is a result of a set of speech acts (actions) performed under some framing rules; authors follow the regulations of academic publishing as well as reviewers and editor. Therefore, Searle distinguishes between brute facts that exist independent of humans and institutional facts that depend on human society. For example, this article can be seen by students (primary or probably high secondary schools) as any essay, so from Searle's perspective is a brute fact, while only under the background of research as well adhering to framing rules like scientific methods, publishing, etc., will be considered institutional facts. Further, a single speech act might be a result of performing several business processes.

The modeling of business processes is a key engineering activity required before making any sort of analysis, process redesign, and general management. In literature, there are different schools or methods for modeling business processes: BPMN [6], Petri net [7], Object role, and Event-driven [8]; but among the common and familiar ones are a UML activity model and DEMO, which are the interest of this work. DEMO has a breakthrough approach for designing and modeling business processes where it supports richer concepts for business processes that adopts ontology principles. On the other hand, although the UML Activity model is not like DEMO originating from the technological world (software developers), it attracts business process modelers and becomes familiar to modelers and business analysts.

III. Ontology Principles

An ontology in philosophy studies the existence, reality, and being. The commonly cited definition is the specification of conceptualization [9]. The main concern of ontology in the computing discipline is the interoperability problem where at least two different agents or systems; for example, two different information systems, want to interoperate. In this case, the heterogeneity of these two agents makes queries or assertions between them impossible. It is because there is no shared and standard meaning for the vocabulary used in the communications. For instance, the types of messages, the content, and what it means. Therefore, the need for standard semantics of the messages communicated, their content, and schemas is obvious to enable interoperability; it is the concern of ontology. For example, a big interoperability case can be observed in the medical field, such as in SNOMD .Healthcare systems use SNOMD to record medical treatment incidents that enable information about patients to follow between hospitals, practitioners, and funding agencies [10]. Another example can be observed in tax systems where tens of thousands of taxpayers interoperate with government agencies using an ontology specified using ontology language for e-businesses [11].

Ontology is a sort of conceptual model that needs to be developed using ontology representation language [10]. Although there are standard languages developed initially to support ontology representation that stemmed from knowledge-based systems like Common Logic, and OWL FULL/Lite [12], which is standardized by W3C, the use of software engineering languages such as UML [13] and MOF [14] as well as information systems modeling languages (i.e. ER) have attracted the ontology community because of its visualization feature and definitive engineering object they can specify. Therefore, we have different competing languages with different capabilities but share the principle of being originated from set theory and predicate calculus. However, a conceptual model will represent the individuals, relationships, and messages with their different classes and define and unify schemas. This description is like an agreement about the semantics and interpretation of things in some world of interoperability [10]. On the other hand, a reasoner is an important element of the architecture of ontology management tools or Ontology Server that enables drawing conclusions from premises using mathematical logic and theorem prover disciplines.

Gruber in his famous paper [9] come up with the main principles of ontology, and the one that best fits the problem of this research is Ontological commitment which is about plausible re-using of ontology. It refers to how far we must alter the application's world to commit to the ontology [10]. For instance, it is well known that businesses implementing enterprise computing solutions like SAP, Peoplesoft, or Oracle Financials must significantly alter their business processes in order to get the most out of the software [10]. However, Colomb argues that ontological commitment would be high if the ontology supporting conceptualization of a world is used outside the scope. Therefore, the problem of ontology comes in here because an organization has its particular set of institutional facts (conceptualization) created by different speech acts (Searle’s institutional facts theory) that mostly is different from the ones canned in software packages such as ERP or the implemented platform.

IV. DEMO

DEMO is a business process design methodology that focuses on enterprise ontology theory which has studied and formalized what actually business is doing independent of realization and implementation issues. The enterprise ontology builds on a set of principles. This work only considers the ontological model, operation axiom, and transaction axiom, which Dietz [15] explores the big picture of it.

Dietz argues that to understand the current and future enterprises with the given complexity, an ontological model (white box approach) is needed as a conceptual model. However, it focuses on the essential model that uncovers the hidden essence of an enterprise from its actual appearance. The operation axiom is a fundamental theory behind DEMO builds on that goal by abstracting the organization operations into two kinds: production acts (P-acts) and coordination acts (C-acts), where both are performed by the subjects representing actor roles. It defines Actors as an elementary set of authority and responsibility. While the transaction axiom groups a set of elementary c-acts into a transaction concept, it also defines three main phases that each transaction should follow: the Order phase, the Execution phase, and the Result phase.

Informally, DEMO is a business process design language that stems from ontology principles and other related disciplines to allow a compact and deep design of business processes. It has rich concepts and features. A fundamental feature of DEMO is its Essential Model concept, which is designed independently of an enterprise's implementation and realization concerns. First, it states that actors in an enterprise are roles performing two basic kinds of acts: production acts and coordination acts. Second, Actors perform two kinds of acts: production acts and coordination acts. They contribute to achieving the enterprise's purpose or mission by performing production acts. While they enter and comply with mutual commitments about production acts by performing coordination acts. The second axiom, the Transaction Axiom, states that production and coordination acts occur in consistent socioeconomic patterns called transactions.

A. *Actors*

By playing different critical roles, people of an enterprise are considered the intrinsic element in DEMO. A subject who plays some role is called an actor in DEMO. For example, in this context, the actors are the Authors, Reviewers, and Editors. As explained in the following subsections, those actors perform basically two actions: P-acts and C-acts. However, DEMO identifies an actor cycle where actors as autonomous objects constantly loop through to perform tasks or agendas. An actor is performing actions, C-acts, for the reason of C-fact that who commits to respond to within a limited time. Each type of agenda has a set of rules called action rules to deal with it.

As a consequence, actors enter into a network of assignments and commitments where each actor, through response to agenda, triggers assignments of work to others (agenda) in a chain until reaching a terminal point. Therefore, an enterprise is a system of actors who perform two kinds of acts: production acts and coordination acts to respond to the agenda in the form of C-facts. Dietz calls this principle the operation axioms.

B. *Production Acts*

"By performing production acts (P-acts for short), the subjects contribute to bringing about the goods and/or services that are delivered to the environment" [15].

This quote by Dietz shows a production act is a primary action that supports the ontological model. It is a fundamental action for an enterprise that stems from a fact called production fact (P-fact) that is considered a definitive result. For example, the facts resulting from the judgment of accepting paper, shipping an order to a specific customer address, and deciding to admit postgraduate students are the results of mainly production acts. Production acts are of two types: martial (i.e., storing, transporting physically) and immaterial. For example, the delivery of goods of order is material, while acceptance of a paper is immaterial. However, this corresponds to Searle's theory concept of changing the social state. So, production acts not like other actions; it makes a social change of state; in our example: a paper is accepted, a student is admitted, and an order is received that is a different state than the previous ones and with new consequences.

C. *Coordination Acts*

"By performing coordination acts (C-acts for short) subjects enter into and comply with commitments towards each other regarding the performance of production acts," [15] . Also, Dietz says in this quote, P-acts occur because some P-fact usually triggers a coordination act that a performer actor does and is directed to another actor called the addressee. Searle's theory [5] interprets this as reporting social attitude, for example, request, promise, assertion, etc. For instance, the request made by this author to the journal is a coordination act, as well as the request from the editor for reviewers to review a paper. Facts created by C-acts are called C-facts, such as in our example, Reference No. of a paper, time of assigning a paper to reviewers, response or feedback item from reviewers etc. On the other hand, a set of C-acts with their C-facts are needed for the existence of a P-act; for example, the C-acts shown are for publishing an article, the P-act in this case. Therefore, C-acts usually do not exist as an independent entity but are related to a production act more accurately production facts. This view of this can be seen in Fig. 1, which shows two worlds: C-word and P-world, where actors change the state of both. This state is incremental, so at a given time, a set of C-facts and P-facts have been created, representing the state of that time. Therefore, the accumulative state represents the history of an enterprise. Searle's theory has more elaboration concept for this point, which calls them both institutional facts, the record and memory of speech acts that occurred.

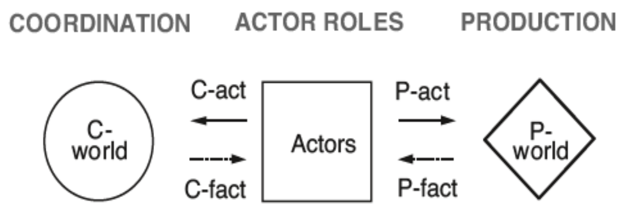

Fig. 1. Graphical representation of the operation axiom from (Dietz, 2006).

D. Transaction

The set of related C-acts contributing to one P-act constitutes a transaction. A business process might have one or more transactions. As shown in Fig. 2, DEMO recognizes universal patterns consisting of request, promise, state, and accept, which also define a transaction. Each transaction, in this case, has two actor roles: consumer and producer, aiming to achieve a specific result. For example, in Fig. 2, this pattern states that the cause of producing a new or original thing, the production result, ontological, is because a consumer starts requesting it from a producer. In this case, and for any C-acts, there is a commitment. Therefore, performing actions through a transaction entails taking turns in entering into and complying with commitments. For instance, the state "result requested" is created because of a customer making a request, more importantly, commits to that to demonstrate responsibility. A producer promises the result requested through the state "result promised".

As Dietz argues, often, it is the case that promised C-acts are performed tacitly in practice. After this, the request undergoes processing by a producer to produce the result (ontological action), which creates the state "result produced" as well as stating the result (hand over in material kind or communicate in immaterial kind) to be checked by a consumer, therefore, creating the two states respectively: "result stated" and "result accepted." Similarly, the acceptance of C-acts is usually performed tacitly. For example, my request as an author(consumer) to the Journal Editorial Board (producer) creates the state "result requested" while getting a notification from the journal system as a representative of the main Editorial board a claim of promising to process the request and so will create the state "result promised." After this, the journal makes a notification that states the result (result stated), which will be checked by the author (result accepted). It is easy to observe that the promise and acceptance actions are performed tacitly, which means there is an assumption that the Journal Body is complying with the promised result since no assertion came for them, saying the opposite[15]. Furthermore, DEMO identifies three phases that usually a transaction is subject to it: The order phase (O-phase), the execution phase (E-phase), and the result phase (R-phase). It typically entails a conversation in which a set of coordination acts communicated between two actor roles to produce a clearly defined outcome regarding a P-act/fact. However, in the O- phase, the initiator and the executor try to come to terms with the transaction's desired outcome: the production fact that the executor will produce and the intended time of creation. Then, the executor creates this production fact during the execution phase.

During the result phase, the initiator and the executor try to agree on the actual production fact that has been produced and the moment of its delivery (both of which may differ from what was initially requested). During these phases, instances of transaction type will be created that correspond to the type of production fact, which is the result.

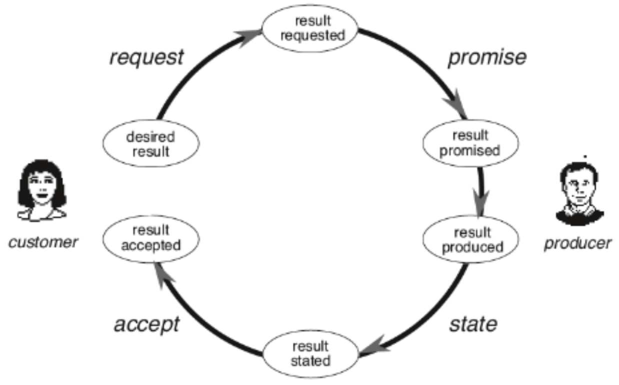

Fig. 2. Transaction pattern (Dietz, 2006).

V. Related Literature

The literature related to this work is Business Process Management (BPM). On the one hand, the Process mining approach is a growing field in BPM that extract the business process model by starting from event logs; mostly business processes have footprints (the performance of actions or events with information like timestamp and owner or customer and etc.) recorded in simple form like spreadsheets to complex one like ERP and databases or workflows repository. This fact enables discovery of a model, where we can perform an enhancement or conformance checking for these models [2]. However, this work is in line with this conformance-checking goal of Process mining approach. But the model proposed in this work is not sensitive to semantic heterogeneity problems, which enables comparing diverse process models. Also, the proposed approach does not assume existence of a repository of instances or the events log for business processes (footprint) to function, although it is possible to base the legacy model on the events log, which will add some accuracy as well as can solve the problem of lack of documentation for the Business process (a BPM principle). However, this work supports the situation of analyzing the business processes before operationalization.

On the other hand, there is a school of research [16] that addresses this problem based on the similarity of model nodes using approaches such as NLP Antunes et al. [17] and edit distance [18] .These approaches perform analysis on a repository of models; that act as knowledge-based for BPM to serve different purposes such as reusing part of existing models in the modeling process, merging models (i.e. company acquisition), and conformance checking [16]. This school of research, although share some features with the proposed approach (using the metamodel for matching), it represents a different direction to the problem under the general umbrella of Business process analytics.

Moreover, other approaches under this school consider the grammar of the label that calls for part of speech tagging and parsing, which is not part of the proposed approach [16]. These are more information retrieval methods. This diversity can be better described as the difference between the qualitative approach (the proposed one) and the quantitative approach.

VI. Gap Analysis Methodology

This section has been organized as a set of principles that constitutes the main constructs of the method. They are: 1) Modeling business process using Activity model injected with DEMO concepts (Principle A), 2) Building the Domain ontology of BPs (Principle B), 3) Gaps reasoning process (Principle C).

It is clear now that using business process design languages such as UML activity model or BPMN is the first step towards the goal of this work. The author chooses the UML activity diagram for its commonality and for reducing the learning curve for modeling business processes. However, many studies in the literature have shown a synergistic relationship between BPMN and UML activity [19]. It turns out clearly that, from the discussion in section 4, DEMO as a design language for business processes is more elaborative than the activity model, so integrating DEMO concepts into UML enables describing the essential model of an organization. It helps in understanding the behavior of an enterprise independent of the context and implementation, and technology issues. Therefore, this will be called principle A, which aims to develop a DEMO profile for annotating the UML Activity model with DEMO concepts discussed in section 4. In principle B, a domain ontology for the organization is needed to unify and standardize the vocabulary used because we have two worlds: newly implemented and legacy business processes. Finally, principle C develops on that by providing new semantic-based methods for the gap analysis. Before discussing these principles, an interpretation of what we have obtained from the literature so far paved the way for understanding the model of solution in this section.

The gap analysis is defined by Monk & Wagner [20] and Kendall & Kendall [21] as the process of identifying the differences between the current system and the desired future state or the functionalities covered by new business processes.

A. How is P-Fact Created, and What makes it Different?

A couple of C-acts contribute to creating a P-act, which naturally involves a decision or judgment (called ontological action) using or applying the enterprise's rules to produce a P-fact. For example, visa approval (a P-act) comes into existence through a series of coordinated C-acts; these are like verifying eligibility, validating documents, assigning an employee, etc., along with their corresponding C-facts such as passport and return flight tickets. Based on DEMO methodology, these C-acts originally belong to the O-phase in a transaction that precedes the E-phase. As explained, the order phase concerns requests and promises between communicators. Therefore, O-phase starts with the "request" act and ends with the state "promised". The E-phase starts with the P-act and ends with the state that the P-fact is created [15].

Obviously, the behavior of producing certain P-fact can be expressed necessarily by a set of related C-acts and C-Facts that are part of the O-phase. Conversely, a set of related C-acts should necessarily exist for a P-act to exist. They are preconditions for the corresponding P-Act. Identifying these C-acts enables observing the differences or comparing any arbitrary two P-acts. Therefore, there are three scenarios: matched, similar to some extent, and not matched or related. However, according to the context of this work, the assumption is that the comparisons will be between processes from the same domain. The basic assumption of ERP is to standardize a domain of business processes. It is the case that an enterprise is interested in queries like whether a new process B, for example, credit control, can replace existing or legacy process A. Of course, A and B here belong to the same domain. Therefore, the fundamental question, which is the mainstream interest for enterprises, is how much A differs from B and What changes are required in this case. One of the major failures reported in the literature is the change needed to comply with the standard best-practiced processes. The sort of difference B will make appears on the set of c-acts. They are going to produce either simialr c-facts or different based on certain improvements have been adopted, for example, following new standards, protocols, and technology.

B. How are Two Business Processes Different?

To identify differences between business processes, we must first understand their fundamental operations. For generalization across organizations - independent of implementation details and technological layers - an ontological perspective offers standardized, context-independent interpretations. This approach enables consistent comparison and difference identification through a unified conceptual vocabulary. DEMO is a rich design language that provides this view. According to DEMO, a business process consists of one or more transactions, and so do transaction types. Each transaction centers on a result called a P-fact that must have instances of its type when the transaction executes. In this E-phase which comes after O-Phase, a request is submitted, and P-facts will only exist if one or more C-acts have been performed that might also produce C-facts.

The stable result in a business process from a DEMO perspective is the P-act [22]. For example, let's look at the visa approval process. The visa document with a unique visa number, in essence, is about the P-fact resulting from P-act -visa approval process. Also, let's think about the process of getting this article published. An *approved article* is the main stable action that ultimately results in an approval letter with a DOI or unique reference number for publication. In both cases, there are a couple of intermediate C-acts that have been performed, such as eligibility check and send-to- reviewer respectively. On the other hand, in this context, Searle's institutional facts theory [5] provides an elaborate interpretation that is P-fact is considered a kind of institutional fact. Seral shows that speech acts under some context count as an institutional fact, which has some social reality impacts. Informally, speech acts theory argues that speech can be expressed as rules of an organization in a formal context. Therefore, P-acts are examples of speech acts under some context that change social reality and produce institutional facts. For example, the visa approval document, MSc certificate, and Check finical document are all examples of institutional facts resulting after a set of speech acts have been performed under some context. The context is framing rules or constraints; for example, what makes a blank paper with some figures in US dollars written, is not a formal cheque document or financial claim [1]. The change in social reality is observing what happens to the situation before and after getting, for example, a MSc. certificate or visa approval which is different. Hence, what enterprises are actually doing is performing different sorts of speech acts (building blocks of BPs) that create institutional facts. DEMO calls these institutional facts, P-facts that have a subordinate the set of C-facts of its realization. Therefore, P-act is a stable result in business activities that concerns the creation of institutional facts that are necessarily realized by C-acts involving speech acts.

In principle, two P-acts can be different because they have a different set of c-facts. However, they might agree on P-fact itself but have the same set of c-facts with varying sequences of execution. This situation provides an interpretation of what added quality means for a business process or generally the sort of change happening between two processes where there are different scenarios of semi-matching between them with a reality that their P-acts agree only on a subset of c-facts. This analysis provides insights independent of implementation and realization.

This background is enough now to realize the principles of the proposed approach to the problem.

C. *DEMO Profile for Activity Model (Principle A)*

A profile is a system of subclasses that provides a powerful extension mechanism to some metamodel (in this case UML). It allows the original metamodel to acquire new syntax or semantics and other features that are explained in OMG [14]. The profile is needed for the reason of lacking corresponding DEMO concepts in the UML activity model.

Using the UML Activity Diagram modeler can specify workflow steps, such as in Fig. 3, a simple example of the Visa Approval process. It consists of several activities needed to add value to visa applicants. These are, ApplyForVisa, Assign_To_EMP, and Verify Documents, as well as one structured activity called Visa Processing that involves the nested activities: Eligibility Check, Approve, and Issue_Visa. In a workflow, execution progresses sequentially. Upon receiving a token and all required inputs, an activity immediately triggers the next activity in the chain. This control transfer continues along the sequence until an exit point or the final flow node is reached. Object nodes, such as 'AppForm' and 'Visa', represent data or objects that are produced or consumed by activities and also participate in the flow of execution. An activity can comprise multiple nodes and edges, as illustrated in Fig. 4, where each node signifies a distinct step in the execution.

This profile aims to enrich UML activity diagrams with DEMO concepts, which are not natively supported by standard UML activities. The initial step in developing this profile involves pinpointing the core elements within the activity model that require extension to incorporate DEMO's linguistic constructs. DEMO, as explained, adds standard ontological concepts that lead to a better understanding of what a business is doing. The central concept is the P-act (result) that produces P-fact(s) but with the support and coordination of a couple of C-acts that subjects have initiated. Activity in the UML Activity model is a central concept that represents one way of modeling behavior which is a description of potential events that could happen in real-time OMG [13]. It involves one or more actions and can be orchestrated using forks, joins, decisions, merges, conditions, and loop nodes. An action can call an activity as well. Therefore, an activity can be used to model business activities and computation procedures.

Fig. 4 shows the metaclass Activity has been extended to model P-act as well as Action (an activity can have one or more actions) that is extended to model C-act, so C-act and P-act are stereotypes (a kind of change needed for the metamodel).On the other hand, both P-fact and C-fact are kinds of classes, so an extension of the metaclass class of UML is needed, as shown in Fig. 3; a metaclass class is extended to add both concepts. Further, ObjectNode is extended to model C_facts, which is an abstract activity node that usually represents an output of an activity that participates in the workflow. To be modeled is necessary for the context of this work, although it is optional in the convenience of using a UML activity diagram. In addition, P_act and C_act need identity, so an attribute is added to the stereotype to allow to specify the uniqueness (this concept is not included in the original DEMO but is necessary for this work).

On the other hand, since StructuredActivity is associated with an Activity class as a whole-part relationship (aggregate association) in the original UML metamodel [13], it can inherit the same property of its parent. StrcuturedActivity is an activity group that involves nodes and edges as subordinate objects. It allows nesting actions to form a hierarchy. It could be an alternative structure that models P_fact/P_act, but in this work, we only considered the first option (class extension) because it is simple and more convenient.

Furthermore, a UML Package concept can be used to model a business process. In contrast, the transaction concept can be mapped to ActivityPartiton(swimlane), which groups a set of ActivityNode and edges. A swimlane represents some role or corresponds to a business unit, showing a separate view and responsibility boundary. This is because the UML activity diagram does not have a transaction concept. Activities may describe procedural computation, forming hierarchies of activities invoking other activities corresponding to a business process.

D. *Building the Domain Ontology for Business Processes (Principle B)*

This principle is to develop an ontology for the domain of business processes. This kind of ontology is known as Endurant ontology DOLCE [24,10], the ontology of data objects that are independent of time. A potential interoperability issue arises from the variations in meaning and interpretation of data and messages (schemas) used in business process communication, a phenomenon termed semantic heterogeneity.

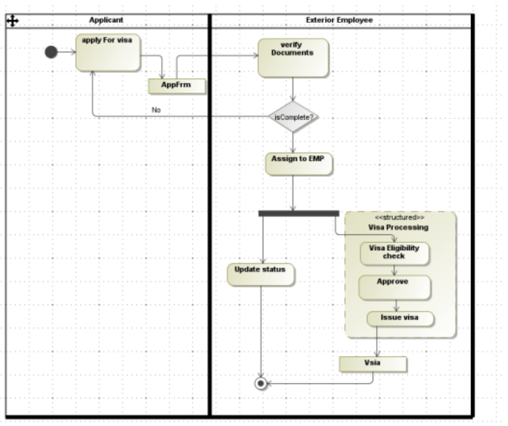

Fig. 3. Example of visa approval process using activity diagram.

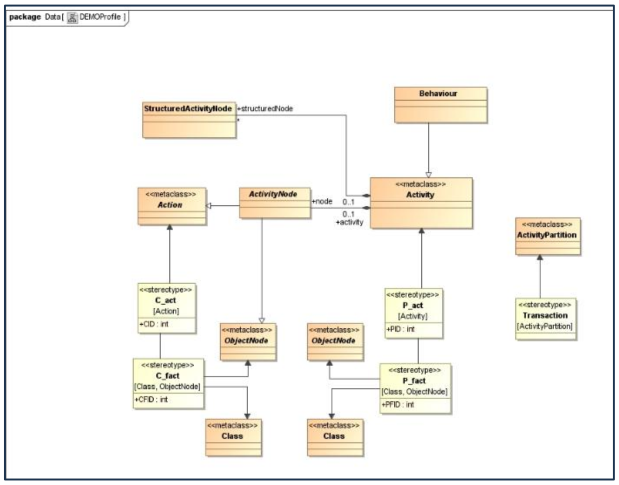

Fig. 4. DEMO profile for activity model.

In our physical world, this problem is evident; for example, the same word in the language has two different meanings in two communities or the same subject refers to it by two different words. Also, in electric power systems, a refrigerator works in one country but not in another because it is designed to use a US system of 110 volts and 60 cycles per second for current. In contrast, the other country uses 240 volts and 50 cps. The problem appears when the new business process wants to replace a legacy business process. Therefore, we do expect a semantic heterogeneity problem between the two business processes. In this context, it is the meaning and interpretation of the P-facts, C-facts, and their corresponding speech acts. For example, suppose there is a service to check the format of a submitted journal that is based on the Harvard standard of citation. In that case, it is unlikely to replace a service in another journal that uses IEEE or APA standards as well as the system of journal citation. Also, if an application uses the ISI standard of ranking journals, the JCR, it will unlikely replace Scopus standard SJR.

Similarly, a service purchasing items from Amazon is unlikely to be able to replace the purchased items on eBay. However, it is obvious that standardization is needed in all these cases before a hand, and this is where an ontology concept comes in. Therefore, a language is needed to describe the ontology according to ontology principles. The UML design language as one candidate has been chosen for its familiarity and visualization feature mentioned because OWL, for example, does not have a graphical representation. Therefore, the business process needs to unify the meaning of words or vocabulary used for interactions or communications using a design language like a class diagram, OWL, DL and others. This specification also explicitly includes the structure of complex objects usually hidden in a single system's conceptual model [10]. Colomb argues that ontology is a kind of conceptual model but exists outside the domain. It is, therefore, a standardizing of the meaning of P-facts/C-facts which is necessary to perform the gap analysis task. As a consequence of this principle, the business process designer needs to specify an ontology using like UML class diagram or OWL. This class model should specify the institutional facts of the domain of some business application. However, many CASE tools are available that support modeling and transformation between modeling languages.

OWL, standardized by the W3C, is a rich ontology representation language that allows us to specify individuals in a triple store format (Subject-Predicate-Object). An RDF triple, which consists of a predicate connecting a subject to an object, forms the basis of this representation. An individual can possess properties, which define direct relations from a domain class to a range class. By default, instances of properties have the most general domain and range, owl:Thing. OWL defines two primary property types: object properties, which describe relationships between individuals (e.g., participatesIn, enrollsIn), and data properties, which describe attributes of individuals (e.g., age, weight). The domain of a property can be restricted using cardinality constraints, such as owl:FunctionalProperty, which asserts that a property has at most one value for each instance. For example, the following OWL syntax declares hasFather (an object property) as functional:

```
<owl:ObjectProperty rdf:ID="hasfather">
<rdf:type rdf:resource="&owl;FunctionalProperty" />
<rdfs:domain rdf:resource="#Son" />
 <rdfs:range rdf:resource="#Person" />
</owl:ObjectProperty>
```

Similarly, OWL allows us to restrict the range of a property using the concept of surjectivity (i.e., every instance of the range must participate). For example, in a postal system, if we want to express that a postal code belongs to a city, and each city has one single postal code, we can model this using such constraints.

Moreover, we can use SPARQL [25] to query an ontology represented in OWL, which has also been standardized by the W3C and is supported by tools such as Protégé (cite). For instance, we can query whether a property is functional:

```
SELECT ?property
WHERE { ?property rdf:type owl:FunctionalProperty .
FILTER (? ?domain = sc:Son)) }
```

A property in this example is a variable that will be bound to specific values based on pattern matching. This allows reasoning about the ontology which is abbreviated by namespace sc (schema of some ontology). We specify conditions in the WHERE clause to be satisfied, in this case specifies whether an RDF graph explicitly defines a property as functional. The FILTER clause adds further restrictions, such as requiring that a property must have the domain son. For instance, the property has Father specified in the OWL ontology above will be returned.

Additionally, NOT EXISTS can be used with FILTER to assert certain constraints. Moreover, a query can be specified for each part of RDF instances using rich built-in predicates and operations, such as intersection, union, and others.

In the following, a demonstration of a case used throughout the paper is presented as part of a postal system ontology. Fig. 5 describes the structure of some institutional facts created by a set of corresponding c-acts, which will be specified later in the section. The main production fact is manifest (see Fig. 5), which consists of a set of properties and c-facts required to fulfill the postal system's primary activity: sending a mailpiece from a sender with a specific address to a receiver with a specific address. Addresses, in this case, belong to a superclass named NAaddress, which has the properties city (range: Clist), postal code (range: Pcodelist), district (range: string), and street name (range: string). The mailpiece, referred to as mailRequest on the left side, has properties including ID, city, postal code, and ship type, which ranges over a specific list called ShipMethod. A customer who initiates this request pays an amount (ranging over RateSchedule) and obtains a stamp by referencing postage. The payment is declared through a postage invoice, which contains a set of properties identifying the date, amount paid, and shipping type.

1) Sample of Mailpieces (invoice)

a) Address Information

- Sender's Address: Name, street address, city, state, ZIP code
- Recipient's Address: Name, street address, city, state, ZIP code

b) Postage

- Stamp (Metered Info): Evidence of payment for postage
- Postage Amount: Value of postage paid

2) Sample of Manifest

- Container Details:
  - Container type (sack, tray, pallet, etc.)
  - Container number or identifier
  - Weight of the container
- Mailpiece Details:
  - Total number of mailpieces
  - mailpiece type (letters, flats, parcels, etc.)
  - Total weight of the mailpieces
- Origin and Destination:
  - Originating postal facility
  - Destination postal facility
- Date and Time:
  - Manifest creation date and time
  - Departure time (if applicable)
- Personnel Information:
  - Name and signature of the postal worker preparing the manifest

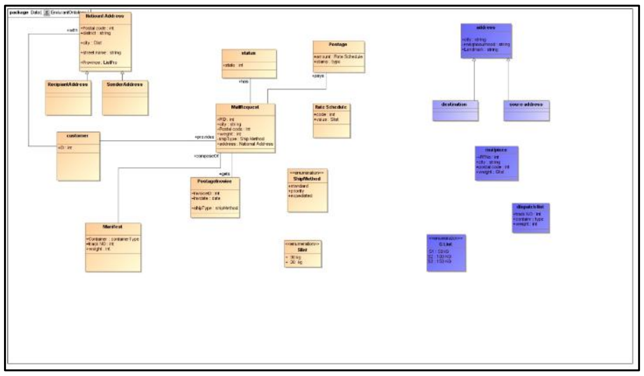

Fig. 5. Postal system endurant domain ontology.

VII. Gaps Detection

The gap analysis aims to discover the alignment of new business processes with business objectives and how far the current practiced business process is from this newly adopted one (which involves some change). The software packages, with their new embedded business processes, define new practices and standards (stemming from research and long experience of Gaint enterprises) for a given domain of business, such as accounting, purchasing, recruitment, etc. It turns out that the identification of noncompliance and its processing is a complete process following the principles of quality management [26].

This section aims to build on the principles established so far to standardize and formalize the gap analysis process that establishes the base for automation.

It is obvious that now we need to focus on P-act/ c-facts; that would be the starting point in observing the differences even between two BPs from different domains. ERP or other Enterprise packages is to replace a business process from the same business domain. For instance, an accounting business process is expected to replace another accounting business process but not purchasing, for example. However, how do we know if two P-acts have identical matches or semi-matching?

Colomb [22] argues that P-act is the stable result which means the c-acts are variable part. Therefore, the difference arises in the set of C-acts with their C-facts that realize the stable result P-act. The assumption supported by principle B; says two c-facts are identical because they belong to the same class or type based on a unified ontology that specifies the vocabulary being used and provides standard meaning. However, the kinds of P-facts and C- facts and how their creation is performed will ultimately make the fundamental difference. This suggests that we need to do a deep analysis of C-acts. So, the fundamental question becomes when and how two corresponding C-acts are different. The domain ontology reduces this problem into a matching function that asserts where an individual (c-fact) belongs to some existing class. Furthermore, this mechanism can be extended to implement like the Substitution principle that makes whenever an instance of the superclass is valid, the instance of the subclass is valid [27]. Therefore, a superclass with a stronger postcondition can substitute a subclass with a weaker precondition.

This work argues that the practical consequence of this approach is that the production of the same facts signifies the same behavior assuming the same domain.

However, given two similar P-facts, there might be some functional or non-functional (the fundamental assumption of change, such as adding efficiency or economy to business process) differences, but that must be reflected in the C-acts with their C-facts in some way, such as extra inputs or/and different sequence of performance of c-acts. Based on DEMO view when two processes or P-acts have identical production facts, they have already make a response to the same first request in O-phase but probably with some different executions commitment , eventually they will be having an exact result stated in R-phase. In principle, these reports a similar behavior, although they may have different scenarios of execution in E-Phase.

A. *Mapping UML Activity into Ontology Individuals*

The UML activity model [23] represents the business process at a high business level. We need to map it into ontology representation using one ontology representation language in order to make the reasoning. UML activity diagrams represent both static structures, such as action inputs and outputs, and dynamic processes. From an ontological perspective, for each endurant entity, represented by a class model, there exists a perdurant entity that brings it into existence. This implies that data and processes should be consistent, with each data element resulting from a specific action. We utilize activity models to represent these perdurant entities. Crucially, the domain ontology metamodel (Principle B) serves as the primary source of these facts. They are the set of related P-facts and C-facts of the organization worlds: P-World and C-world. We can do this mapping as definitive statements in OWL(or any other ontology languages), such as asserting that there is an individual P-fact, visa reference number in Fig. 3 process : P-act(visa- No, date) or asserting fact such as an invoice: C-fact(invoice-No, date, amount). These facts are usually augmented with the specification of transactions in the process model, getting a visa, for example.

The output of this mapping process is a set of facts (as will be shown in the case study) of metamodel level 1 because it models objects that are instances of metamodel level 2 [14]. From OWL prospective, a c-fact is an individual that does not belong to any class but has a set of properties. The properties of the individual are kind of Datatype property. Concrete objects, or specific instances, such as a visa application for Mr. David, are considered level zero individuals. A concept fact (c-fact) can be defined by a combination of properties with literal ranges. For example, in the case study presented in Fig. 6, a mailpiece, produced during the E-phase of the postal system's main mail delivery process, is a c-fact. This mailpiece represents an essential communication document (c-act) for delivering packages, letters, and other items. Properties of this individual c-fact, such as the sender's address, have a domain of 'Customer' and a range of a literal (e.g., string). Similarly, properties like city, state, and zipcode are also literals. Other properties, such as stamp and postage amount, have numeric ranges. Conversely, meter info is likely an object property with a range that is a class with a defined structure. The container type property of the Manifest follows a similar pattern. Because OWL allows the representation of meta-levels [23] within a single model, unlike UML, OWL and RDFS are commonly used for ontology representations.

The problem of converting UML models into OWL has been explored extensively, yielding results across various methods: MDA (Model-Driven Architecture), ontology profile-based approaches, and hybrid techniques. The choice is about cost-benefit analysis approach which has been elaborated with concrete examples by a fruitful OMG ODM project [28], for sake of simplicity will not be considered here.

The view of processes is needed because it consumes the Endurant facts which is a sequence of c-acts corresponding to a specific P-fact. Because OWL does not directly recognize P-acts and their related c-acts, the workaround is to add a meta property, called 'type,' for each act to classify it into one of a set that can be constrainted to c-act,p-act,c-fact,and p-fact . In fact, all c-acts and P-acts can be modeled as OWL classes that can have a set of OWL or RDF properties. Alternatively, OWL-S can be used since it supports service modeling, such as atomic and composite services. OWL-S has a rich structure capable of modeling inputs and outputs. However, we use OWL for its simplicity and comonality.

Fig. 6 illustrates a business process involving concept acts (c-acts). To produce the manifest (main perdurant fact, or p-fact) through the ontological action 'generate manifest,' a mail item is received by the action 'receive mail.' Note that some actions, such as the first two in process A, are manual steps. Consequently, a sequence of concept acts (c-acts) must be executed.A mail package will be gauged,, then a formal request will be created, create request, a service fee will be calculated by calc service fee, and sorting of packages will be thorough Sorting action. These c-acts have a sequence (incoming edge and outgoing edge), input(s) and output sometimes. However, process B in the right side of Fig. 6 is similar to that but involves some differences (discussed later in details) which represent the to-be system or target.

To specify a general method of mapping acts in Activity model to OWL, we can make some abstraction. We do model a property called next for keeping track of the sequence in c-act individual.

B. *Mapping Target Ontology Into Source Institutional Facts (Principle C)*

The domain ontology is all about intuitional facts. These institutional facts as discussed are created by speech acts under some context which represents framing rule. However, the situation now is we have got two different worlds of institutions:

to-be system and as-system so the question how do we know the institutional facts of to-be system is similar or matching the as-system's institutional facts which as argued in this research as fundamental principle. This distills down into finding a base where the automation machinery going to present later can use it to decide such as on the gap between source and target. Consequently, this step is essential for the following stages, which is about mapping and comparing processes. This aim to establish correspondences between the c-facts from the two different worlds; will refer to them to-be system as the target world, based on intuition that we need to move towards the new system and as-is system will be called the source world, based on the perspective of the main production act.

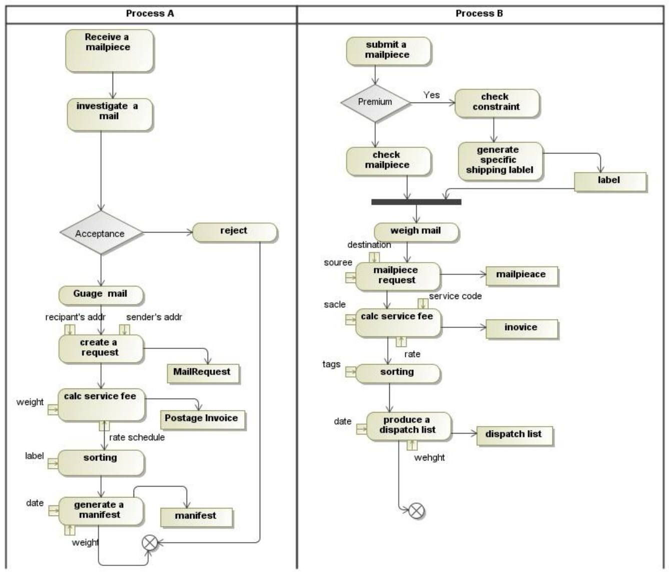

Fig. 6. Postal system main business process (Perdurant ontology.)

How far or near the target from source is the principle of ontological commitment (re-use) which can be low or high as dicussed. The commitment will be low when an ontology used in its usual scope. However, the inputs to the mapping process consist of the two ontology worlds, along with the documentation of the target world (e.g., data dictionaries and BPMN models) .The output is a specification of alignments, mapping target institutional facts to source institutional facts, which depends on the level of ontological commitment. In extreme cases, this may result in extending the ontology with new concepts or creating specializations (subclasses).

More importantly, as demonstrated in the case study, generating the manifest—the main production act—requires a set of C-acts to be performed. These acts lead to the fulfillment of a commitment or promise to create a mailpiece. A mailpiece consists of several attributes: sender, recipient, stamp, and postage amount. This perspective provides a conceptual link that helps track or connect these elements.

A domain expert may observe a similar structure for the mailpiece, though with some variations—for example, differences in naming (e.g., "service amount" instead of "postage," or "meter info" instead of "stamp") or the presence of new concepts not originally defined in the system (e.g., "date," "log info," etc.). These variations are often captured in a data dictionary, which defines the business vocabulary. While precise terminology is ideal, a degree of flexibility is acceptable at this stage, with a greater focus placed on identifying key roles and entities.

Business analysts typically consult both the data dictionary and documentation of the business process—such as BPMN diagrams or activity models—when performing alignment activities. This approach is a common and effective practice within enterprises, as it allows experts to focus on the primary roles and major institutional facts, which serve as abstractions for complex systems. In principle, this alignment process can be automated or semi-automated as in the literature.

It is common for business analysts or ontology engineers to identify relationships or mappings between concepts, whether at the schema level or instance level. This challenge is well-known in the literature and is often referred to as semantic matching, semantic mapping, or ontology merging [29]. Several approaches have been developed to identify relationships such as equivalence, subsumption, and others. Tools like LogMap [30] and the Alignment API [31] are widely used for this purpose. According to this activity untimely will end up with a table similar to Table I (based on the case study) in which the target concepts mapped into their corresponding source c-facts P-facts. The ontology can be built either based on source world or target world since mapping has been performed.

TABLE I. MAPPING TARGET INSTITUTIONAL FACTS INTO SOURCE INSTITUTIONAL FACTS

Source	Target	Comments
Mail request	Mailpiece	Similar
manifest	Mail list	Some differences more attributes added
Invoice	Invoice	Similar
Guagement	weights	Different scales

C. *Generating Implication Rules Based on Corresponding Actions(D)*

We have now a unified ontology has been annotated with the target concepts after mapping as explained in the previous section. This section will deal with the base for matching and discovering the gap between two business processes.

As Colomb [22] argued, the stable result is the production act, meaning that all different c-acts and their associated c-facts will not change the reality of p-fact. Additionally, an institutional fact can be understood as a record of a speech act. Furthermore, as stated in the quote, "the state of the P-world at a specific point in time is defined by the set of P-facts created up to that moment, while the state of the C-world at a specific point in time is defined by the set of C-facts created up to that moment." In simpler terms, the creation of a fact of a particular type represents a state transition within one .of these two worlds.

This implies that we can trace the primary production acts by examining the pre-c-facts generated during the E-phase and the post-c-facts generated during the R-phase. Consequently, when two distinct c-acts result in the same c-fact instances, it indicates that they exhibit similar behavior.

Therefore one can observe that dependencies usually exist in the creation of c-facts, which often follow a logical sequence. Fig. 7 illustrates that the main production fact, the Manifest, requires the creation of two necessary c-facts: Mailpiece (c-fact1) and Invoice (c-fact2). Additionally, each c-fact can be associated with a set of c-acts that were performed prior to its creation (referred to as the c-world state), which contribute to its existence. This observation suggests that we can use these facts as a basis for tracing and matching subordinate elements between two worlds.

For instance, both Mailpiece and Manifest have sets of c-acts that contribute to their existence. Let us refer to these sets as Set M (for Mailpiece) and Set F (for Manifest). In this context, Set F is a proper set, while Set M is a subset because the Manifest encompasses multiple Mailpieces. This implies that the Manifest represents the whole, while the Mailpiece is a part of that whole. Consequently, there may be many parts (Mailpieces) that belong to the same whole (Manifest). Therefore, in this case, we need to identify the corresponding whole in the to-be world and construct similar sets of c-acts.

Since institutional facts from the to-be world are already mapped to corresponding institutional facts in the as-is world (principle c), it is possible to define a mapping function or make a relationship between them (i.e., they contribute to the creation of the same fact). Once these sets are constructed, a more specific matching between corresponding sets of c-acts can be established. For example, in Fig. 7, the Mailpiece has its corresponding MailRequest, and actions such as Gauge Mail and Weigh Mail are also corresponding actions.

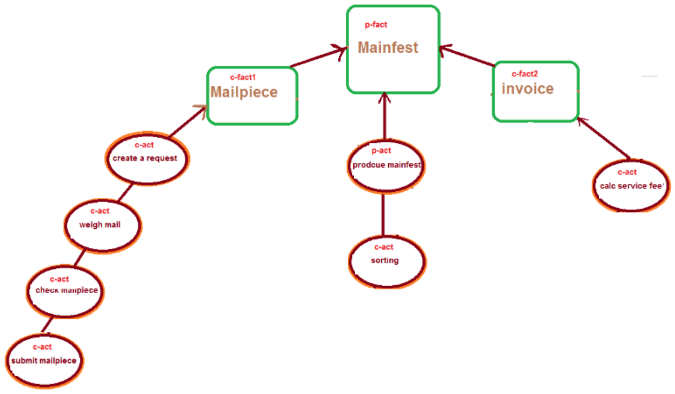

Fig. 7. Example of dependences among c-acts that contribute to the Production fact Mainfest.

More importantly, I argue that, based on this perspective, we can conceptualize an implication rule where the left side (a set of c-acts from Process A) implies the right side (a set of c-acts from Process B), provided that the major institutional facts (i.e., c-facts) are similar.

Furthermore, we can closely examine the direct relationship between the left-side and right-side sets by analyzing the inputs to c-acts and matching them with the standard domain ontology through querying or assertion. This process ensures that all inputs originate from the same ontology. In this context, we will have a set of properties derived from different classes. These classes belong to the left-side and right-side sets, which are either similar or represent different versions of the same entity—the institutional fact.

Now our ultimate goal is to determine when a BPs fail to be replaced by other BPs. The failure is because of different reasons but we argue that it can be commonly studied under incompatible classes or individuals in which properties are conflicting. Therefore, we need to look at the specific problem of when two classes are incompatible because at least one property in first class conflicts with the corresponding class's property. Hence a reporting of mapping failure with evidences. However, identifying these discrepancies is essential where a business can leverage them to adopt potential and necessary change (gap analysis principles).

Now let us take concrete feedback from our case study, Fig. 7 models part of a main business process in postal system that have the original BPs or as-is system, call it Process A and the to-be system, call it process B.

Process A:

1) Recive a mail
2) Check mail
3) Decide Acceptance - input – mailpiece info
4) If accepted then
5) Guage mail - input : mailpiece output : weight
6) Create mail request : output mailpiece request
7) Calcuate service fee
8) Make invoice : output invoice
9) Sort mailpieces
10) Generate manifest : output manifest

Process B:

It is similar to A but it has additional subprocess premium service that is not considered by A. Assume for simplicity the following differences.

1) Recive a mail
2) Check mail
3) Decide Acceptance - input – mailpiece info
4) If accepted then
5) Guage mail - input : mailpiece output : weight
6) Create mail request : output mailpiece request
7) Calcuate service fee
8) Make invoice : output invoice
9) Sort mailpieces
10) Generate manifest : output manifest

From step 2, for example we have a subprocess running in parallel to deal with premium service:

1) If premium service then
2) Check constraints
3) Calculate service fee
4) Confirm payment : output receipt
5) Generate shipping label : Output new label
6) Priortize handling : output special manifest

Let us imagine also three major differences in institutional facts between A and B described by Fig. 6, blue classes at right side): National address (NAaddress) that is introduced as a new government regulation in Process B. It follows a different format, including fields such as landmark, city, and neighborhood, postage amount is specified in the local currency, measurement units (guagment): There is a difference in measurement systems; Process B uses local weight standards with a different scale.

These cases can be summarized in the following (Process A):

1) In mailpiece the type of postage amount is Rayal currency.

2) Also in mailpiece as well as mainfest the addresses are formed from the structure of { a-building No (4-digits),b-street name (15-character), c-district(limited set of values : all local district for a city),city (limited set of values: all local country cities), etc}.

3) Guagement is a set of 50 kg, 100kg, 150kg, etc.

A prior knowledge is that the to-be system or process B has US dollar currency in any financial transaction also does not support the national address's structure and 15kg, 30 kg weights scale (i.e. large volume of business is in this scale) because the to-be system has only Mutiple of 50kg.The data dictionary of these systems could be a good source for investing such requirements or information.

It is required now to generate the implication according to the principle of finding the corresponding of institutional facts. Since we already have the specification for business processes as part of ontology in an ontology language like OWL (as described in …), this step is going to extend that to incorporate this implication generation step. The mapping of institutional facts in the source to target institutional facts will act as an input to this process (i.e. Table I). It can start with finding the main production fact and their subordinates or c-facts. Then mapping this main production fact to its corresponding fact from source.

In the case study the main manifest and manifest are similar concepts and represent the same real-world entity. Having different names or synonyms for the same concept can be automated as in the literature using corpus or wordnet and dictionaries methods [32]. It can classify concepts into the same class when they are belonging to these relationships: is-kind-of or is-a (always hold) and part- of a whole.

1) Main production fact rule

1. Mainfest $\longrightarrow$ postage invoice, mail request

2. Main Maninfest $\longrightarrow$ inovice, mailpiece, label

Based on Table I and the principle of left side implies right side then it follows that:

postage invoice, mail request $\longrightarrow$ novice, mailpiece, label

Also from Table I:

2) Branching

1.1 Post invoice $\longrightarrow$ invoice

1.2 Mail request $\longrightarrow$ mailpiece

1.3 Since label has no corresponding concept in table 1 it means new entity needs to be added to the ontology of new business process world. There are two interpretations in this case a) new requirement does not exist in the source so far b)or more refinement for existing concept(s).

Then we need to find out what are the speech acts (c-acts) have contributed to the production of these c-facts from both side of implication which will inherit this implication also.

3) Based on B will get the following implication of acts as consequence :

For 1.1:

calc service fee $\longrightarrow$ calc service fee

Also,

For 1.2: create request $\longrightarrow$ mailpiece request

Therefore, we conclude that the inputs to these c-acts are also equivalent

Weight, rate schedule $\longrightarrow$ scale, rate

By querying or asserting the developed ontology in principle B) above we will discover that they are not different concepts.

Make a request $\longrightarrow$ Create a request

Sender, Receiver $\longrightarrow$ source, destination

We need now to identify their corresponding classes in order to discover their incompatibility and properties in conflict. Since OWL and SPRQL has standard ontology to represent properties with its different rich characteristic functions, we can develop standard methods.

To identify incompatible classes, we can approximate the problem using the concept of subsumption. In mathematics, we say that class A subsumes class B if every element of B is also an element of A. In other words, B is contained within A, and we can say that A represents B. One can think of the relationship between the to-be system and the as-is system using the substitution principle: if A is a superclass of subclass B, then an instance of A can substitute an instance of B. However, we need to investigate the conditions under which this substitution is valid or invalid. If we can determine that business process A (the to-be process) subsumes business process B (the as-is process), then we might conclude that A can replace B. To reach such a conclusion, a set of operations—such as intersection, set difference, and others—must be applied.

But how do we know when substitution is not possible? For example, in the case of a more constrained subclass, substitution may break. According to set theory, if two sets differ in their elements—either by having disjoint elements or partial overlap with at least one exclusive element—then they are not equivalent.

However, in this context, we need more precision. We require a rigorous definition of what it means for an element to be "different." One way to realize this is by examining conflicting properties. These conflicts help determine incompatibility. Therefore, we address this question in the following section.

There are of course many reasons for discrepancies that are difficult to count but it can be generalized under common classification theme such as in Table II, then for each class we provide a treatment.

TABLE II. AN EXAMPLE OF DISCREPANCIES AMONG PROPERTIES BASED ON THE CASE STUDY

S	Concept in A	Type	Is_essential?	Concept in B	Type	Difference
1	amount	Property	No	Postage amount	property	naming
2	NA address	Set of Properties for a class	Yes	address	class with different set of properties but it has some common items	Structure
3	gaugement	Set of individuals for a property	Yes	wight	properties	Range – different scale
4	tracking	class	NO	New class with properties	Does not exist	Not esists (to Model new class object)

Table II demonstrates the concept in process A (source) and the corresponding concept in process B (target), type of concept from ontology prospective (class, property, etc.) with their differences stated. Moreover, a column is added to adopt metaproperty is_essential that discriminates or defines the essential properties for each class. An essential property is one that must exist in each instance. This can be based on the theory of BWW (Bob). The purpose of proposing it here is that the final decision of dissimilarity can rely on it which allows this task to be automated.

D. *Building Standard Queries and Assertions (Principle E)*

In order to reason about discrepancies, we need to standardize and formalize testing of a gap in the form of assertions and quires. The basic assumption is to use SPRQL since we ended up with ontology specified using RDF-based language (OWL). An alternative is use the built-in machinery of consistency check [33] but there is no more control especially if customised quires or assertion are required.

Referring to Table II, one can infer that some properties have been converted into class types, such as the NA address in the new process (No 2). Additionally, the range of one property has changed, resulting in a subsumption relationship, as seen with 'gaugement' and whight; the initial range is a subset of the new range, indicating a change in the property range's scale. Furthermore, a new property has been introduced in the new system, which was absent in the legacy system, as illustrated in case 4(tracking).Therefore, the following queries demonstrate how to reason about these cases based on the source and target ontologies given.

1) Range discrepancy query: This is typically for like case 3 in Table II. The first obvious case occurs when the value of a property in target class does not belong to the range class of the source(i.e. range of source is mailpiece while the range class of target is manifest).Second, the range of the source property is more specific than the target property(target range for instance is a set of red and yellow while source is bule only).

The discovery of the first case is straightforward because we can use the SPRQL not exist in the filter clause to assert that an individual has property's range of the source (i.e. postage amount is not riyal). The second case can be obtained by different ways; one way is to use OneOf OWL construct that allows to specify, for example, a property having specific range (enumeration data type).Therefore, we can use SPRQL to disprove that the range set of the source is not either subset or proper set of the target. For instance, the base to discover the incompatibility in case 3 in the Table II:

```
# Check subset condition, A ⊆ B: every member of Set A is also in Set B
Select? x
FILTER NOT EXISTS {
?x rdf:type :RangeOfClassA.
FILTER NOT EXISTS {?x rdf:type :RangeOFClassB }
}
```

The Not-exists clause will return false always except when one tuple appears in the result showing that one member of set A is not part of set B.

```
# Check proper subset condition, B ⊇ A: there exists a member in Set B not in Set A
FILTER EXISTS {
?x rdf:type : RangeClassB .
FILTER NOT EXISTS {?x rdf:type : RangeClassB }
}
}
```

Notice that this is the inverse of the first query so the Not-exists clause returns true only when there is a member in B does not belong to A.

2) Structure discrepancies query: It is typically like case No 2 in the table where a property needs to be replaced by a class(NA address) which is a recurring problem. Since based on DEMO their corresponding classes are from the same P-act then some overlapping of properties might occurs. However, there are different types of structure differences that might happen. Mostly these will recur in the whole ontology and the advantage of this is that a bench of quires will be re-used, therefore reducing the cost of the development. In the following these different types of structure discrepancies will be sketched.

Type1: Class range vs. property range

The following query will return instances of properties that at most one of its range is a class. The postage amount in target could ranges over specific class (standard list) while the source amount has range integer.

```
SELECT ?Property1 ? Property2
WHERE {
 ?property1 rdfs:range ?range1 .
 ? Prpoerty2 rdfs:range ? range2
 FILTER NOT EXISTS (( range1? Owl:datatype ?) And (range2 owl:Class })
 }
}
```

Type 2: Ranges are classes but one subsumes the other

Using proper set and subset check as discussed above allows us to make a test for which is a subset of another, but before that an initial test is required to map corresponding properties instead of comparing a source property with all target properties. For example, Address and NA Address, in such a situation Hamming distance can be used which computes distance between two strings. Properties can be encoded using bitmaps such as 4bits for each character (we need determine the size based on the dynamic range of characters exist). The Hamming distance function computes how many number of characters are in differences. In this case, it will result in less distance between NA address and Address than among other properties. Also, the case could be two different names of properties used but with the same semantic meaning. For example dispatch list and manifest .A well-established method of Wordnet [32] can classify these concepts into the same class which is an is-kind relationship. Wordnet-based methods can also detect is and part-of relationships.

Type 3: Cardinality discrepancies

More restriction could be specified on ontology of source and target where properties have specific cardinalities (min, max) therefore must present in testing. For example:

- A Manifest must include at least one Mailpiece (a manifest cannot be empty).
- A Mailpiece can be included in at most one Manifest (a mailpiece cannot be listed on multiple manifests).

Min/Max cardinality test: Remember this test on TBox or the terminoglical level of the ontology but not instance level.Therefore we need to teat ComposeOf property if it specifies max or min cardinality .Usually OWL allows one to make these constrains by defining a subclass of the restriction class (OWL built in).In the following a query ComposeOf for this case will be checked for if it has min cardinality constraint. Constraint and card value are variables will be instintiated when the where condition satisfied. The where condition binds a restriction variable with instance if it finds rdf type owl:Resitirction class which has property ComposeOf.

```
SELECT ?constraint ?cardValue
WHERE {
 ?restriction rfd:type owl:Restriction ;
  owl:onProperty : ComposeOf;
  ?constraint ? cardValue .

 FILTER (?constraint =minCardinality)
}
```

Based on that queries and assertions we will be able to verify if any major differences exist for essential properties of the source ontology and accordingly, we can reach to the final decision of compatibility or not because of the exitance or not existence of conflicting essential properties.

We could combine or package these quires to be executed in a sequence using Nested substructure where select ... is going to be nested or chained. Therefore, we can get one final and single answer for a couple of quires and assertions. Moreover, the implication rule principle can be automated and linked with this these queries using XSLT which can transform the implication rule into a direct call to APIs that will perform the necessary tests as explained above.

TABLE III. COMPASSION AMONG THE METHODS USED FOR GAP ANALYSIS

Criteria	Manual inspection	Process Mining	Proposed method
Automation support	No	Yes	Yes
Semantic heterogeneity	Yes	Yes	No
BP Instances required	No	Yes	No
Error rate	High	low	low
Reliability	Low	High	High
Performance	Low	High	High
Scalability	No	Yes	Yes

VIII. DISCUSSION AND INTERPRETATIONS

Enterprises often adopt new and innovative business processes under the assumption that they will lead to breakthrough results. However, if such changes are implemented without sufficient preliminary analysis, the risk of failure significantly increases. For example, a recently established company in the region specializing in paper manufacturing and recycling—with a capital exceeding one million dollars—faced failure during an attempt to upgrade its systems and reengineer its business processes.

Traditional approaches in such cases are typically ad hoc, suffer from semantic heterogeneity, and lack scalability due to inherent complexity. Table III shows a theoretical comparison based on expert reasoning of manual inspection, process mining, and the proposed method across several key criteria: semantic heterogeneity, instances requirement, automation support, errors, reliability and scalability. As demonstrated, manual inspection is unreliable, has a high error rate, and lacks automated support. Although process mining increases automation and reliability, it does not address semantic heterogeneity and is still dependent on the availability of process instances. By removing semantic heterogeneity and lowering reliance on process execution logs, the suggested approach outperforms both process mining and other methods while maintaining high reliability and performance. This comparison serves to highlight the anticipated benefits of the suggested approach, even though empirical validation is still a future objective.

Since business processes (BPs) are fundamentally about institutional facts (i.e., production facts), the model proposed in this article offers a way to mitigate such risks. It does so by unifying the institutional vocabulary used by businesses to describe their expected services and products.

This unified vocabulary enables standardized gap analysis, supported by DEMO, which provides a formal language for expressing the essential elements of a business process—abstracted from implementation and realization details. Such abstraction is a powerful tool for managing complexity.

Furthermore, comparing the ontologies of to-be and as-is business processes is feasible because both originate from the same business domain. Various scenarios can arise, such as one process being more constrained than the other. These discrepancies can often be grouped under common classes, allowing the development of a general method for systematic analysis and resolution.

One related outcome of this work is that it facilitates documenting gaps so they can be understood at a high level, as argued by Jeston [34]. This is the fact that models are transformed into a knowledge-based system; therefore, it not only supports reasoning about gap but also acts as an informative repository that can be reused for different analysis goals, which is a principle aligned with the BPM objectives. For example, top managers, executives, and strategist are interested in answering inquiries about different business processes for various reasons, such as benchmarks to determine enhancements for as-is processes that can justify the investment [35].

The major cost is developing an ontology manually for a domain of business processes or institutional facts. Also annotating the model with DEMO concepts and mapping target institutional facts into source institutional facts. However, some capabilities of the ontology toolset can be utilized to some extent, such as ontology learning, consistency check and the model's mappings facility of QVT to reduce this cost. Moreover, conducting large-scale evaluations in different domains with different scenarios will highlight more classes of discrepancies. These are elements of future work.

Regarding the reengineering process itself, standardization enabled by DEMO helps to define the kind and type of change required as usual practices of adopting new ERP (commits to ERP ontology). Therefore, the problem would shift to focusing on which institutional facts (C-fact and P-fact) need to be changed as well as its set of actions (C-act and P-acts). Moreover, DEMO provides an ontology to talk about processes gap and their classification.

IX. Conclusion

This study examines the challenges of gap analysis problem when replacing legacy business process (as-is) with new business process (to-be). Business processes evolve to incorporate qualities such as economy, productivity, and efficiency, necessitating a thorough analysis to ensure alignment with organizational objectives and strategy. Gap analysis plays a critical role in answering key questions, such as whether a new process can replace an existing one and, if not, identifying the reasons. This work proposed a structured method to identify gaps among business processes. It consists of Four principles: 1) Developing DEMO profile (principle A) 2) Building domain ontology for BPs (principle B) 3) Mapping target institutional facts into source ontology (principle c) 4) Generating Implication rules based on corresponding actions 5) Reasoning using standard discrepancy quires(principle E). Because business processes are about the production of institutional facts, semantic heterogeneity prohibits comparing and analyzing two different processes (main source of failure); building domain ontology(consists of both endurant and perdurant) that unifies terms, concepts, messages and interpretation is an essential process in the proposed approach (principle B). This suggests mapping the to-be system's institutional facts into their corresponding source's institutional facts (principle c). Also, DEMO has richer concepts for designing business processes that focus on the Essential model of an enterprise. It handles the complexity through the identification of the main production act, p-act, which is the stable result. Therefore, a set of c-acts could be identified and compared that is required for the existence of the P-fact (i.e. manifest) b; business activities independent of realization and implementation issues. Therefore, a UML Activity as a famous design language has been profiled to support DEMO concepts (profile A). Integrating DEMO concepts into UML activity Diagram puts forwards and facilitates the analytics of business processes. This suggests that we reason about gaps using such as SPRQL (Principle E). However, this study contributes to the state-of-the-art of BPM and ERP community by providing a facility to compare different business processes semantically, either as legacy or new processes, providing a great opportunity for business analysts, architects, and strategists to make critical (multi-million dollars) decisions.

References

[1] A. Koschmider, M. Fellmann, A. Schoknecht, and A. Oberweis, "Analysis of process model reuse: Where are we now, where should we go from here?," Decis. Support Syst., vol. 66, pp. 9–19, 2014.

[2] W. M. P. van der Aalst, Process Mining: Data Science in Action, 2nd ed. Springer, 2016.

[3] M. Hammer, "What is business process management?," in Handbook on Business Process Management 1: Introduction, Methods, and Information Systems, Springer, 2015, pp. 3–16. doi: 10.1007/978-3-642-45100-3_1.

[4] M. Hammer, "Reengineering work: Don't automate, obliterate," Harvard Business Review, vol. 68, no. 4, 1990.

[5] J. R. Searle, The Construction of Social Reality. New York: The Free Press, 1995.

[6] OMG, Business Process Modeling Notation, Version 1.2, 2009, OMG Document Number: formal/2009-01-03

[7] G. Rozenberg, J. E. Models, and P. N. I. B., “Elementary net systems,” Springer, 1998. [Online]. Available: https://link.springer.com/content/pdf/10.1007/3-540-65306-6_14.pdf.

[8] A.-W. Scheer, Business Process Engineering: Reference Models for Industrial Enterprises. Springer, 2012. [Online]. Available: https://books.google.com.

[9] T. R. Gruber, “Toward principles for the design of ontologies used for knowledge sharing,” Int. J. Hum.-Comput. Stud., vol. 43, no. 5–6, 1995. doi: 10.1006/ijhc.1995.1081.

[10] R. M. Colomb, Ontology and the Semantic Web, vol. 156. IOS Press, 2007.

[11] Universal Business Language Version 2.1. [Online]. Available: http://docs.oasis-open.org/ubl/UBL-2.1.html

[12] W3C, OWL Web Ontology Language Reference, W3C Recommendation, Feb. 10, 2004. [Online]. Available: http://www.w3.org/TR/owl-ref/.

[13] OMG, An OMG Unified Modeling Language (OMG UML) Publication, 2009. [Online]. Available: https://www.omg.org/spec/UML/20161101/PrimitiveTypes.xmi.

[14] OMG, *Meta Object Facility (MOF) Core Specification*, Version 2.0, OMG Document Number: formal/2006-01-01, Jan. 2006. [Online]. Available: https://www.omg.org/spec/MOF/2.0/.

[15] J. L. G. Dietz, Enterprise Ontology: Theory and Methodology. Springer, 2006. doi: 10.1007/3-540-33149-2.

[16] A. Schoknecht, T. Thaler, P. Fettke, A. Oberweis, and R. Laue, “Similarity of business process models - A state-of-the-art analysis,” ACM Comput. Surv., vol. 50, no. 4, 2017. doi: 10.1145/3092694.

[17] G. Antunes, C. Pereira, L. F. Pires, and M. van Sinderen, “Using ontologies for enterprise architecture model analysis,” *Inf. Softw. Technol.*, vol. 54, no. 1, pp. 4–14, Jan. 2015. [Online]. Available: https://doi.org/10.1016/j.infsof.2011.06.005.

[18] R. Dijkman, M. Dumas, and L. García-Bañuelos, “Graph matching algorithms for business process model similarity search,” *Data Knowl. Eng.*, vol. 70, no. 6, pp. 597–625, Jun. 2011. [Online]. Available: https://doi.org/10.1016/j.datak.2011.02.003.

[19] M. A. Cibrán, “Translating BPMN models into UML activities,” Lecture Notes in Business Information Processing, vol. 17, pp. 236–247, 2009. doi: 10.1007/978-3-642-00328-8_23.

[20] E. Monk and B. Wagner, *Concepts in Enterprise Resource Planning*, 4th ed. Boston, MA, USA: Cengage Learning, 2013.

[21] K. E. Kendall and J. E. Kendall, *Systems Analysis and Design*, 10th ed. Boston, MA, USA: Pearson, 2019.

[22] R. Colomb, Module 09: Business Process Modeling, lecture notes, Enterprise Information Systems MCM2623, University of Technology Malaysia, Feb. 20, 2008.

[23] OMG, OMG UML Superstructure, Version 2.1.2, 2007, OMG Document Number: formal/2007-11-02.

[24] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari, “WonderWeb Deliverable D18: Ontology Library (Final),” IST Project 2001-33052 WonderWeb, 2003. [Online]. Available: https://www.loa.istc.cnr.it/old/DOLCE.html.

[25] E. Prud'hommeaux and A. Seaborne, “SPARQL Query Language for RDF,” W3C Recommendation, Jan. 2008. [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/.

[26] M. J. Epstein, J.-F. Manzoni, and A. Davila, *Performance Measurement and Management Control: Behavioral Implications and Human Actions*. Bingley, U.K.: Emerald Group Publishing, 2014.

[27] B. Liskov and J. Wing, “A behavioral notion of subtyping,” *ACM Trans. Program. Lang. Syst.*, vol. 16, no. 6, pp. 1811–1841, Nov. 1994. doi: 10.1145/197320.197383.

[28] R. Colomb et al., “The object management group ontology definition metamodel,” in Ontologies for Software Engineering and Software Technology. Springer, 2006. doi: 10.1007/3-540-34518-3_8.

[29] A. Thiéblin, M. Chekol, and M. Giese, "Ontology Alignment with Deep Learning: A Survey," *Semantic Web*, vol. 11, no. 6, pp. 1011-1044, 2020.

[30] J. Jiménez-Ruiz, B. Parsia, and U. Sattler, "LogMap: Logic-based ontology alignment," in *Proc. 10th Int. Semantic Web Conf. (ISWC)*, 2011, pp. 274-289.

[31] J. David, H. Laforest, and J. Euzenat, "The Alignment API 4.0," in *Proc. 8th Int. Semantic Web Conf. (ISWC)*, 2011, pp. 182-197.

[32] G. A. Miller, "WordNet: a lexical database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39-41, 1995.

[33] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen, “From SHIQ and RDF to OWL: The making of a web ontology language,” *J. Web Semantics*, vol. 1, no. 1, pp. 7–26, Jan. 2003. [Online]. Available: https://doi.org/10.1016/j.websem.2003.04.001

[34] J. Jeston, Business Process Management: Practical Guidelines to Successful Implementations. New York, NY, USA: Routledge, 2018.

[35] P. Harmon, Business Process Change: A Business Process Management Guide for Managers and Process Professionals. 4th ed. United States: Elsevier, 2019.