International Journal of Systems and Service-Oriented Engineering
Volume 12 • Issue 1

CODVerif:
A Continuous Verification of Service-Oriented Architecture Data

Malik Khalfallah, Paris Est Créteil University, France*
Parisa Ghodous, Lyon 1 University, France

ABSTRACT

CODVerif is an approach that aims to verify the data being inserted in a data store continuously. CODVerif leverages the combination of ontology and workflow technologies in order to define workflows that are specific to the domain of "monitoring data insertion." These domain-specific workflows are constrained on two dimensions: (1) They use a set of workflow elements that are specific to the "monitoring data insertion" domain. (2) The logic that these workflows support is predefined by relying on a set of common data insertion scenarios. Nevertheless, CODVerif is flexible enough to allow users to define continuous data verification workflows with higher complexity logic thanks to workflow operators that can be applied on the "monitoring data insertion domain"-specific workflows. To illustrate the applicability of CODVerif, the authors deploy it in a customer relationship management (CRM) application and show how CODVerif is used to support users to verify the data they populate in the CRM. They have also evaluated the CODVerif approach.

KEYWORDS

CRM, Data Standards, Naming Conventions, Populating Data Workflows, Salesforce, Service-Oriented Architecture

INTRODUCTION

Service-oriented architecture applications (SOAs) define a set of services from a service provider. Service consumers could invoke these services by providing the right inputs to obtain the expected outputs. For many applications, the input/output data does not need to be defined rigorously (e.g., class attributes representing informal descriptions). However, for many other applications, it is necessary to define SOA input/output data very rigorously (e.g., an attribute that aims to capture a URL (Uniform Resource Location) needs to be well-defined), otherwise, problems would appear at run-time when the data will be used. The rigor comes from the specificity of the application that the SOA supports. This application happens for certain cyber-physical systems (CPSs) applications where the input should uphold a standardized specification strictly, and users cannot deviate from the constraints defined by that standard when defining inputs for SOA services. A typical example is the STK SOA data definition standard (Osorio et al., 2006) and its application to CPSs (https://bit.ly/3K0JOJS). In STK, data must be defined rigorously following the constraints defined by the standard. Otherwise, the data would create problems at run-time when CPS on-board services are invoked. Moreover, besides the standard, SOA data might operate the CPS; thus, the data must uphold a certain naming

DOI: 10.4018/IJSSOE.315582 *Corresponding Author

convention (Paniagua et al., 2019). Therefore, as a good practice, additional rules might be applied even though the standard left the door open and did not put constraints on certain data aspects.

Managing the definition of SOA services' input and output data could be done in a data store (Fischer et al., 2018). Nevertheless, because of the complexity of this data and the number of constraints associated with its definition, it could be challenging for users to ensure the coherency of the SOA data they define. Furthermore, there are two approaches to address the problem of enforcing SOA data coherency in a data store, namely: (i) *a-posteriori SOA data verification approach* and (ii) *continuous SOA data verification approach (CODVerif)*.

In the a-posteriori verification approach, users define their SOA input/output data but check the data's coherency at the end of the data definition phase. The advantage of this approach is that users would not be constrained by a lot of rules related to data coherency when inserting data. Furthermore, this approach is simple and can even rely on third party software to perform the verification once the SOA data is deemed ready (For STK standard, third-party software (https://bit.ly/3NyToG0) can check STK SOA data coherency). Nevertheless, the drawbacks of the a-posteriori verification approach are two-fold:

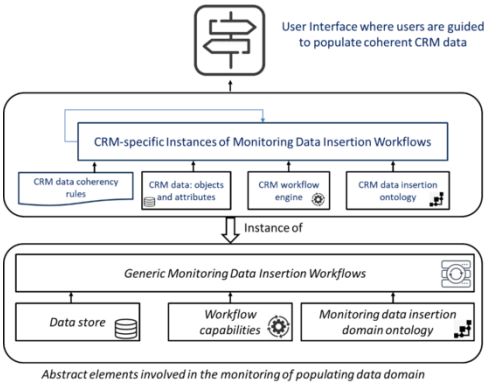
- Users will figure out the errors when the data definition phase is almost done (van der Aalst & Pesic, 2006). This means that users could affect the delivery of their data because they would repeat defining data to resolve the problems raised.
- Users would face many errors in many objects in the data store. These problems often need to be addressed sequentially and manually.

With the CODVerif approach, each time users populate a piece of data, one or many Monitoring Data Insertion processes or Workflows (MDIWs) are launched. An MDIW checks whether the piece of information populated is coherent, following the rules imposed by the standard or even by naming conventions if they would be applicable. If the MDIW detects that a certain object has not been well-populated in the data store, it will raise the warnings or create actions that will remain pending until the problems raised are resolved. The advantage of this approach is that users can continuously address the problems related to data coherency. Furthermore, users will not let the problems accumulate until the end of the data definition phase.

CODVerif Overview

In this paper, we propose the CODVerif framework to enforce SOA data coherency when this data is managed in data stores. The general architecture of CODVerif is depicted in Figure 1. In the bottom part of the CODVerif architecture, we have a data store and a workflow engine. The store for SOA services data could go from an OpenOffice Calc sheet to an object-oriented or relational database defining objects and their attributes. The workflow engine could run workflows modeled with business process management notation (BPMN) (ObjectManagementGroup, 2022) or other notations. Above these two components, CODVerif provides the capability of defining monitoring data insertion workflows (MDIWs). These workflow models are specific to the "monitoring data insertion" domain. Furthermore, these workflow models use specific elements defined in an ontology (Yu, 2014). We call this domain-specific ontology Monitoring Data Insertion Workflow ontology (MDIWO, which is available here https://bit.ly/3JXDaDU). MDIWs models capture the specific activities and events that must be considered ensuring that we satisfy the rules imposed by the target data definition standard or the rules imposed by an adopted naming convention for populating data in the data store.

Figure 1. General architecture of CODVerif



With the configuration of CODVerif that we described until now, users will probably be overwhelmed if we let them define MDIWs that aim to verify their SOA data. Knowing that defining workflow models can be challenging (Dumas et al., 2013) and as we are targeting a specific domain, that is "monitoring data insertion" domain, CODVerif addresses this challenge by developing a set of generic monitoring data insertion workflows (GMDIWs). In summary, CODVerif architecture defines GMDIWs by relying on MDIWO to enforce data coherency rules imposed by SOA standards and the potential naming conventions.

The CODVerif architecture described until now and depicted in the bottom part of Figure 1 is abstract. Nevertheless, CODVerif users can use this abstract layer to create an instance of the CODVerif architecture for a specific use-case of the "monitoring data insertion" domain. Examples of use cases could be the verification of STK data in a data store or customer data definition in customer management systems (CRM) (Shaalan, 2020).

If we consider CRM data as depicted in the top part of Figure 1, the instantiation of the CODVerif architecture for this specific use-case consists in using the set of GMDIWs to attach to relevant objects or attributes in the CRM data store one or multiple instances of GMDIWs (hereafter, workflow instances). Each workflow instance aims to enforce a certain data coherency rule imposed by the standard or the naming convention.

Running Example

We have initially prepared the study in this paper to address STK standard data for CPSs SOA-based applications. However, this standard is related to CPS and needs a complex background before elaborating on an easy-to-follow example. Thus, to illustrate our CODVerif approach, we will apply it to the definition of customers' data in a CRM called Salesforce (www.salesforce.com). This use case is relevant because the definition of CRM data is often constrained by complex naming conventions (Acker et al., 2011). The same approach could be applied to STK-based data and other SOA data standards.

This paper is organized as follows. Next, we propose the conceptual framework that supports the CODVerif architecture of Figure 1. In this section, we detail the different components of CODVerif by providing their formal definitions and by proving some important properties of CODVerif. Then, we identify the different generic monitoring data insertion scenarios we have identified. We will detail the important aspects of the scenarios that have been identified. This work describes the CODVerif prototype and its deployment in a specific use case that comprises monitoring the insertion of (SOA) data in a CRM system. This use case is also used to evaluate CODVerif by involving CRM experts. Finally, we present the related work and then conclude the paper.

CODVERIF CONCEPTUAL FRAMEWORK

CODVerif Specific Workflow Domain

Different studies have been conducted to extend workflow modeling to make it domain-specific. For example, Yousfi et al. (2016) extended BPMN to make it specific to the Internet of Things (IoT) domain. Besides BPMN core elements, such extensions add their constructs. Users will have the possibility to use the core elements of BPMN and the extension elements. CODVerif aims to make workflow modeling specific to the "monitoring of data insertion" domain by introducing constraints in the two following dimensions:

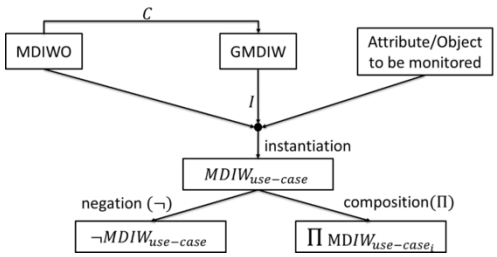
- Constrain the elements that can be used in workflow models by defining a domain ontology (MDIWO) specific to the "monitoring of data insertion" domain,
- Constrain the logic that workflow models accept by defining generic monitoring data insertion scenarios.

With these two constraints, we propose the following workflow operations to instantiate and extend CODVerif:

- Combine the monitoring data insertion scenarios (GMDIW) with the domain ontology (MDIWO) via the *instantiation* operation to create concrete monitoring data insertion workflows,
- Automated creation of MDIW that aims to keep the data store content coherent even after the removal of data (the inverse of populating data) via the *negation* operation,
- Create complex MDIWs via the *composition* operation.

We depict the cODVerif core elements cited above in Figure 2. In the next subsequent sections, we elaborate on these CODVerif core elements.

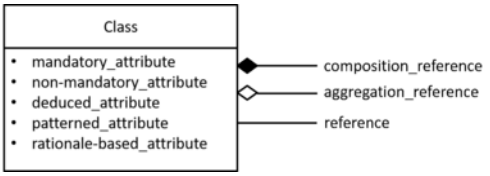
Figure 2. CODVerif Core Elements



Monitoring Data Insertion Workflow Ontology (MDIWO)

The ontology helps us narrow the scope of workflow models we are targeting and thus simplifies the development of operations on these models. Indeed, as mentioned previously, in this paper, we are only focusing on workflows that aim to guide users in inserting data in data stores. Accordingly, we build an ontology that defines workflow core concepts related to "monitoring data insertion" domain. In this domain, we focus on populating attributes and references that objects in a data store could have. Figure 3 depicts the different pieces of information that need to be populated for a generic class in a unified modeling language (UML) model.

Figure 3. Types of attributes and references covered by the MDPWO



The ontology MDIWO leverages message events related to populating data in a data store. MDIWO captures the definition, removing the update message events of references and attributes with their different types, as depicted in Figure 3. MDIWO leverages populating data activities that act on references and attributes with their different types. Furthermore, MDIWO defines disjoint relationships between concepts to facilitate identifying a concept and its inverse. A typical example of a disjoint relationship between two message events is reference_defined_Receive_MESSAGE and reference_cleared_Receive_MESSAGE, which are opposites.

MDIWO supports CODVerif in specifying workflow models related to different scenarios related to populating data. Furthermore, MDIWO aims to support the automated generation of workflow models that undo what populating data workflows did. Finally, MDIWO aims to support the instantiation of monitoring data insertion workflows for specific use cases. Thus, MDIWO defines four concepts:

Concepts Related to Activities that Populate or Clear Attributes and References

Populating data activities could concern:

- Creating references between objects,
- Populating mandatory attributes,
- Defining or reviewing the functions that calculate deduced attributes,
- Populating patterned attributes,
- Populating rationale-based attributes,
- The inverse of the above activities when applicable.

Concepts Related to Exceptions of Unfound Objects for References and Values Rejected for Attributes

When populating data, exceptional situations could occur. Exceptions that could occur concern objects that users look for but would not find when creating references between objects. Furthermore, exceptions could concern values that are rejected when populating attributes. In MDIWO, exceptions do not have opposite classes.

Concepts Related to Event Messages Which are Triggered When Data is Updated

Although workflow message events are used to establish communication between pools in BPMN (Dumas et al., 2013), in this paper, we use message events to create modular and decoupled workflow fragments. Workflow fragments could be combined to support complex scenarios of populating data in a data store. Furthermore, message events can notify users about actions to be performed before the populating data phase ends. Thus, MDIWO creates a hierarchy of send and receive message events to support different scenarios of populating data. The hierarchy of all send/receive messages related to populating data in a data store are defined in MDIWO.

Concepts Related to Conditions Associated with Workflow Flows

Conditions could be associated with flows when defining XOR and OR gateways. Therefore, MDIWO defines a hierarchy of concepts to capture conditions related to populating data in a data store and their inverse.

Every ontology can define concepts and individuals (Yu, 2014). Individuals are realizations of the concepts for a specific use case (e.g., populating data for a CRM or populating data for STK). At this stage, we do not need to define any individual in MDIWO because individuals will be specific to the use case to be addressed.

Operations on MDIW

MDIWO allows us to model MDIWs to address the common scenarios of populating data. Successfully defining operations on MDIWs will allow us to address more complex scenarios and specific use cases in the "monitoring of data insertion" domain. The next sections will detail three operations of MDIWs. These operations are:

- **Instantiation** of MDIWs to address specific use cases (e.g., STK-related monitoring data insertion workflows, CRM-related monitoring data insertion workflows).
- **Negation** ($\neg$) of MDIWs to handle the scenarios of undoing what will be done by GMDIWs.
- **Composition** (Π) of MDIWs to attach multiple data monitoring workflows to a single object or attribute.

Instantiation of MDIW

It is important to differentiate between the instantiation of an MDIW and the creation of a case (van der Aalst, 2005) of an MDIW. The instantiation of MDIWs consists in starting from an MDIW modeled with the concepts of the MDIWO. Then, we replace each concept in the MDIW model with the individual from the MDIWO to address a specific scenario of populating data for the use case being addressed.

According to the observation above, the instantiation starts by defining individuals in the MDIWO. The result would be $MDIWO_{use-case} = C \cup I_{user-case}$. Once the $MDIWO_{use-case}$ has been defined, users can create as many instances of monitoring populating data workflows as necessary to handle the rules related to their use case.

Definition: Workflow instance

A workflow instance is derived from a GMDIW where we *replace* each element used by the GMDIW and defined in $\mathrm{MDIWO}.C$ by the appropriate individual coming from $MDIWO_{use-case}.I$. The *replace* action is denoted as |. Notice that a workflow instance is different from a workflow case (Dumas et al., 2013) because a workflow instance could lead to creating multiple workflow cases.

Lemma

Applying the **instantiation operation** to a well-formed MDIW modeled with BPMN generates a well-defined workflow instance $MDIW_{use-case}$.

Proof

Using the formal notation of BPMN models defined by Ye et al. (2008), where O is the set of all elements in BPMN and ϵ^S is the set of all start events, we prove by contradiction that if GMDIW is well-formed and has the instance $MDIW = GMDIW_{o \in MDIWO \mid o_{individual} \in o}$ then $MDIW$ is well-formed. To be well-formed, $MDIW$ has to satisfy all conditions for well-formed BPMN processes. We show

that $MDIW$ will satisfy the first condition of well-formed BPMN models. The same approach can be used to prove the other properties defined in Yousfi et al. (2016).

- $\forall s \in \epsilon^{S} \cup dom\left(Exc\right),\ \bullet s = \varnothing \wedge \left|s \bullet\right| = 1$

Assume that $\exists s_{individual} \in s :\ \bullet s_{individual} \neq \varnothing \vee \left|s_{individual} \bullet\right| \neq 1$. This $s_{individual}$ cannot exist because any $s_{individual}$ always has the same properties as its class s and thus $s_{individual}$ has several predecessors and successors equivalent to those of its class s.

Thanks to this lemma, CODVerif ensures that if users start from a well-formed GMDIW and instantiate it, they will always obtain a well-formed workflow instance.

Negation of MDIW

MDIWs are defined to ensure that populating data leads to coherent data in the store. As it is necessary to consider data coherency rules when populating data, it is also necessary to consider them when removing data. The objective is to keep data coherent in the data store even though some data entries are removed. Instead of redefining MDIWs specific to the data removal scenarios, we propose defining the negation operation on MDIWs. With the negation operation, if users define a MDIW, then they will be able to generate automatically the inverse of that MDIW. The inverse of a MDIW aims to ensure that the global data coherency in the data store is maintained after the removal of certain pieces of data. The definition of the negation operation is possible thanks to our ontology-based approach that limits the scope of MDIWs. If we had not limited the scope, then it would have been too complex to define the negation operation.

The negation operation on a MDIW model is applied sequentially to elements constituting the workflow from the start events until the end events. The result of each step of the negation operation depends on two aspects:

- The element encountered and its definition in the MDIWO,
- The context in which the concept is encountered is being used in the MDIW model.

MDIWO already captures the negation of certain concepts via the *disjoint* relationship. Nevertheless, this is insufficient, as the negation of certain MDIWO concepts could also affect their successor in MDIW. To comprehensively define the negation operation on MDIW, Table 1 summarizes the rules driving the negation operation. To formally capture the negation operation rules, we rely on the formal representation of workflows defined by Ye et al. (2008).

Table 1. Results of the negation operation on MDIW elements

Negation of MDIW constructs	Description
$\forall a \in O,\ \left(\neg a\right)\bullet = \neg\left(a\right)\bullet$	The successor of the negation of an object a is the negation of the successor of the object a
$\neg\left(XOR\left(a_{1_{c_1}}, a_{2_{c_2}}\right)\right) = XOR\left(\neg\left(a_2\right)_{c_1}, \neg\left(a_1\right)_{c_2}\right)$	The negation of an XOR-split gateway is an XOR-split gateway with output flows switched. Notice that following the GMDIW presented, the XOR gateways always have only two output flows. This is one benefit of CODVerif where the domain scope is predefined.
$\forall a \in O,\ \neg a = a \rightarrow \left(\neg a\right)\bullet\ = a\bullet$	If there is no impact of the negation operation on an element O of an MDIW, then the successor of that element remains unchanged.
$\neg\left(AND - JOIN\right) = OR - JOIN$	The negation of a parallel-join gateway is equal to an OR-join gateway.
$\neg\left(AND - SPLIT\right) = AND - SPLIT$	There is no impact of the negation on the parallel-split gateway.
$\forall a_1, a_2, \ldots a_n \in O:\ \neg\left(OR - JOIN\left(a_1, a_2, \ldots a_n\right)\right) = OR - JOIN\left(\neg a_1, \neg a_2, \ldots \neg a_n\right)$	The negation of an OR-JOIN is the OR-JOIN of the negation of its predecessors.
$\forall a \in O : OR - JOIN\left(a\right) = a$	The OR-JOIN with one single predecessor is constituted of that predecessor only.
Clear_non-mandatory_attribute_ACTIVITY•= non-mandatory_ attribute_cleared_Send_MESSAGE	The successor of any individual whose class is clear_non-mandatory_attribute_ACTIVITY $\in MDIWO$ is an individual of the class non-mandatory_attribute_cleared_Send_ MESSAGE
$\forall a \in \epsilon, \neg a = \varnothing \rightarrow \neg a\bullet = \varnothing$	If the negation of a message event in a MDIW is not defined, then all successors are not defined in $\neg MDIW$.
Mandatory_attribute_to_be_cleared_Receive_MESSAGE•= mandatory_attribute_to_be_populated_Send_MESSAGE	The successor of any individual whose class is mandatory_attribute_to_be_cleared_ Receive_MESSAGE $\in MDIWO$ is an individual of the class mandatory_attribute_ to_be_populated_Send_MESSAGE
populate_patterned_attribute_ACTIVITY $\in$ patterned_attribute_to_ be_populated_Receive_MESSAGE• $\rightarrow$ patterned_attribute_to_be_ cleared_Receive_MESSAGE•= clear_patterned_attribute_ACTIVITY	The negation of an activity that populates patterned attributes is an activity that clears that patterned attribute.
Clear_patterned_attribute_ACTIVITY•= patterned_attribute_cleared_ Send_MESSAGE	The successor of any individual whose class is clear_patterned_attribute_ACTIVITY $\in MDIW$ is an individual of the class patterned_attribute_cleared_Send_MESSAGE

Table 1 continued on next page

Table 1 continued

Negation of MDIW constructs	Description
composite_object_to_be_removed_Receive_MESSAGE•= remove_composite_object_ACTIVITY	The successor of any individual whose class is composite_object_to_be_removed_Receive_MESSAGE $\in MDIWO$ is an individual of the class remove_composite_object_ACTIVITY
remove_composite_object_ACTIVITY•= composite_object_removed_Send_MESSAGE	The successor of any individual whose class is remove_composite_object_ACTIVITY $\in MDIWO$ is an individual of the class composite_object_removed_Send_MESSAGE
clear_rationale-based_attribute_ACTIVITY•= rationale-based_attribute_cleared_Send_MESSAGE	The successor of any individual whose class is clear_rationale-based_attribute_ACTIVITY $\in MDIWO$ is an individual of the class rationale-based_attribute_cleared_Send_MESSAGE

Composition of MDIW

It is possible to associate an object in a data store with multiple MDIWs to check multiple coherency rules for that particular object. Hence, we need to define the composition of different MDIWs. Indeed, the composition of MDIWs consists of applying multiple MDIW on the same attribute that will be populated. The composition of MDIWs is a first-order logic formula involving different MDIWs. The composition operator of MDIWs should satisfy the following requirements:

1. It should be possible to compose an MDIW and its negation $\neg MDIW$,
2. It should be possible to apply the negation on a composite MDIW $Composite - MDIW,$ and the resultant $\neg Composite - MDIW$ should be applicable.

To compose MDIWs, we rely on first-order logic operators because CODVerif aims to enforce multiple coherency rules simultaneously. Hence, as far as logic operators are concerned, the logic operator that satisfies a couple of requirements above is the implication $\rightarrow$ operator. Indeed:

1. If $MDIW \rightarrow \neg$MDIW : this means if MDIW terminates successfully, then its negation can be triggered if the undo of MDIW occurs.
2. If $Composite - MDIW = MDIW_1 \rightarrow MDIW_2 \rightarrow \ldots MDIW_n$ then $\neg Composite - MDIW = \neg MDIW_n \rightarrow \ldots \rightarrow \neg MDIW_1$: this means that negating a composite MDIW, leads to ensuring that clearing the attribute value maintains the data coherent.

To compose MDIWs, the "Or" logic operator $\vee$ and "he" "And" logic operator $\wedge$ are"not"adapted. For $\vee$, the reason is obvious because CODVerif aims to impose multiple coherency rules on the same object and not a subset of coherency rules. For $\wedge$ operator, the reason is that we cannot compose an MDIW and its inverse $\neg MDPW$ via the $\wedge$ logic operator. In this case, we would have

$MDIW \wedge \neg MDIW = null$. Moreover, the negation of a composition leads to a disjunction of MDIWs ($\vee_i MDPWs$) which is not applicable as already clarified for the $\vee$ operator.

Notice that DecSerflow (van der Aalst & Pesic, 2006) proposes to compose workflows using temporal logic operators. Nevertheless, CODVerif does not aim -at least- for the moment to address time-related coherency rules.

Proposition

Applying the **composition operation** to well-defined $MDIW_i$ modeled with BPMN generates a well-defined workflow $\Pi MDIW_i$ modeled with BPMN.

Proof

The proof is by contraction: a composition of well-formed MDIWs is:
$C - MDIW = MDIW_1 \rightarrow MDIW_2 \rightarrow \ldots MDIW_n$. Suppose that this composition $C - MDIW$ is not well formed. This means that one of the component workflows $MDIW_i$ is not well formed. More specifically,
$not - well - formed\left(MDIW_1\right) \vee not - well - formed\left(MDIW_2\right) \ldots \vee not - well - formed\left(MDIW_n\right)$.
Nevertheless, we said initially that our composition involves well-formed MDIWs, which leads to a contradiction.

GENERIC MONITORING DATA INSERTION WORKFLOWS (GMDIW)

Different scenarios exist for populating data in a data store. It is necessary to identify these scenarios to develop the appropriate GMDIWs that will monitor and guide users when populating data.

To develop the scenarios of populating data, we have relied on two inputs:

1. The experience gained when populating data for different purposes while considering naming conventions,
2. The study of the data coherency rules is defined by different SOA standards, including STK.

We have identified 8 GMDIWs related to data insertion. These GMDIWs do not aim to define a comprehensive list of data insertion scenarios, but we believe they offer important insights. In Table 2, we present the rationale behind each GMDIW along with its negation $\neg GMDIW$ which can be calculated automatically thanks to the negation operation defined above.

Table 2. List of Generic Monitoring Data Insertion Workflows (GMDIW)

GMDIW Label	GMDIW Description	¬GMDIW
Scenario 1: Populating relationships between objects	**When users create a new object in the data store, this GMDIW aims to guide users to ensure that this new object will be linked to another object because of the mandatory association between both objects. The object to be referenced might exist or not, and this scenario addresses both cases.**	$\neg\ MDIW_{scenario1}$ **aims to guide users to undo the relationships cleanly.**
Scenario 2: Populating values of deduced attributes	**In the data store, the value of certain attributes (dependent attributes) could be a function of the values of other attributes (independent attributes). This GMDIW aims to guide users to ensure that whenever dependent attributes are populated when the independent attributes have been populated. As the generated values of the dependent attributes could be rejected, it is necessary to raise tasks to guide users to update the independent attributes.**	$\neg\ MDIW_{scenario2}$ **aims to guide users to update the dependent attributes when their corresponding independent attributes will be updated.**
Scenario 3: Populating values of non-mandatory attributes	**In a data store, the value of certain attributes might be optional. However, if a value is defined for those attributes, they might need to be unique. This GMDIW aims to guide users to ensure that whenever a value is assigned to such attributes, it will be required to ensure that it is unique.**	$\neg\ MDIW_{scenario3}$ **is triggered when the attribute value is cleared, and it raises a message event to show this.**
Scenario 4: Populating values of mandatory attributes	**In a data store, the value of certain attributes might be mandatory but also it might need to satisfy certain conditions. This GMDIW aims to guide users to ensure that values assigned to such attributes are unique and that the values assigned to uphold the constraints defined if they exist.**	$\neg\ MDIW_{scenario4}$ **is triggered whenever a value of a mandatory attribute is cleared, warning users it needs to be populated.**

Table 2 continued on next page

Table 2 continued

GMDIW Label	GMDIW Description	¬GMDIW
Scenario 5: Populating values of patterned attributes	**In a data store, values assigned to two or more attributes might need to have relationships with each other. The list of relationships that we consider is the following:** **Attribute a '' value in object x_1 has to be *inline* with attribute a 's value in all other objects x_i in the data store (objects of the same type),** **Attribute a 's value in object x_1 has to be *different* from attribute b 's value'in all other objects x_i in the data store (objects of the same type),** **Attribute a 's value in'object x_1 has to be *inline* with attribute a 's value in a'l other objects y_i in the data store (objects of different types),** **Attribute a 's value in obje't x_1 has to be *different* from attribute b 's value in all ot'er objects y_i in the data store (objects of different types).** **The *inline* and *different* are abstract relationships. The corresponding checking activities are defined in MDIWO and can be instantiated to capture concrete use-case relationships. This GMDIW aims to guide users to ensure that we build the values assigned to certain attributes following the above relationships, if one or more are applicable.**	$\neg\ MDIW_{scenario5}$ **is triggered when a patterned attribute is requested to be cleared.**
Scenario 6: Populating aggregation relationships	**It is possible to create a global object in a data store that aggregates a certain number of components. Nevertheless, when adding an aggregate object, it could be necessary to add objects referenced by this component to the same global object.**	$\neg\ MDIW_{scenario6}$ **is triggered when the aggregate object is removed and ensures that the inner objects are removed.**
Scenario 7: Populating composition relationships	**In a data store, populating hierarchical objects could be performed differently. In fact, there is no "correct" manner to be followed but to maintain the coherency of data; it is necessary to follow the same steps for all hierarchical objects of the same type. As far as hierarchical objects are concerned, two approaches could be used:** **A raw approach to capture the hierarchy of objects.** **A structured approach to capture the hierarchy of objects** **Both approaches capture the same information, but one must populate the data store to keep the content of similar objects coherent.**	$\neg\ MDIW_{scenario7}$ **comprises a workflow that appropriately removes the link between the composite objects and components.**
Scenario 8: Populating values of rationale-based attributes	**In a data store, values of certain attributes might accept different inputs, and all of them might be correct. Nevertheless, the inputs might need a certain rationale that might have been already by the same attributes in other objects.**	$\neg\ MDIW_{scenario8}$ **is triggered when a patterned rationale-based attribute is requested to be cleared.**

Lemma

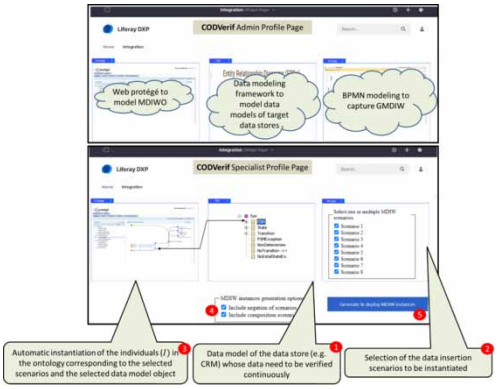
Applying the **negation operation** to a well-defined GMDIW modeled with BPMN generates a well-defined workflow $\neg GMDIW$ modeled with BPMN.

Proof

The proof can be presented by exhaustion through applying the negation operation on each GMDIW in Table 2 and showing that the result is a well-formed BPMN model.

CODVERIF PROTOTYPE

Figure 4. CODVerif liferay-based portal with the two profiles interfaces



We have developed a three-profile liferay-based portal (Yuan, 2012) to implement the CODVerif framework, as illustrated in Figure 4. The first profile is the CODVerif-Admin (top part of Figure 4). Users having this profile aim to perform the following activities required for CODVerif:

- Define and extend the CODVerif ontology (MDIWO).
- Define the data model corresponding to a specific domain whose data needs to be verified (e.g., CRM or STK).
- Define generic monitoring data insertion workflows (GMDIW).

The second profile is called CODVerif-Specialist (bottom part of Figure 4). Users having this profile aim to perform the following activities:

1. Select the object, the attribute, or the reference to be monitored from the target data model.
2. Associate the selected attribute, reference, or object in the data model with one or multiple GMDIW scenarios to enforce data coherency rules or the imposed naming conventions.
3. Generate the individuals in the MDIWO ontology.
4. Determine whether generating the negations and the composition of the selected scenarios is necessary.
5. Generate MDIW instances and deploy them.

The third profile of CODVerif is users populating data stores that are guided by CODVerif MDIW instances.

CODVerif-Admin profile uses a portlet that accesses Web-Protégé (Tudorache et al., 2008) to model the ontology. Additionally, they use Bizagi (Bizagi, 2022) via a portlet to model the different GMDIW, presented earlier, using BPMN (ObjectManagementGroup, 2022). In practice, Table 3 provides one demonstration per scenario to illustrate how MDIW instances deployed in Salesforce CRM help users identify errors in Salesforce data when populating data.

Table 3. Illustrations of MDIWs with salesforce where the description of each example is attached to the demonstration

Scenario	$MDIW_{Salesforce_{sc_i}}$ short demo
Scenario 1: Populating relationships between objects	https://bit.ly/3Nyii8D
Scenario 2: Populating values of deduced attributes	https://bit.ly/3qRLQEl
Scenario 3: Populating values of non-mandatory attributes	https://bit.ly/3Lv3PIy
Scenario 5: Populating values of patterned attributes	https://bit.ly/3qRLLk1
Scenario 6: Populating aggregation relationships	https://bit.ly/3iT3rHy
Scenario 8: Populating values of rationale-based attributes	https://bit.ly/3JPn5Ad

CODVERIF EVALUATION

Evaluation Design

To evaluate CODVerif, we have defined three levels of naming convention rules:

- *Simple*: users can implement naming convention rules easily. For example, a rule could be not to include white spaces in phone number attributes in the CRM.
- *Medium*: an example of such a CRM rule is that each account name should have a score as a suffix representing the importance of that account.
- *Complex*: knowing that in a CRM an account could have multiple contacts (Shaalan, 2020), contact names might need to be attached to a suffix number that could identify two important pieces of information which are internal to the company using the CRM:
  - The identifier of the role played by the contact from the account point of view,
  - The account number to which the contact belongs.

For example if we assume that we have 9 accounts in our CRM and each account could have at most 16 contacts playing different roles for that account, then a contact whose name ends with the suffix 105 will bring the two following pieces of information:

- ID of the role of the contact in the account = $\left(\left(105+1\right)mod16\ =10\right.$
- ID of the account to which the contact belongs = $Int\left(\left(1105-1000\right)/16\right)+1\ =7$

Thus, a complex rule is that contacts shall end with a number capturing the two pieces of information above.

The hypothesis that is evaluated is that it is beneficial for CRM users and users who populate general data to be guided to uphold the naming convention rules and produce high-quality data. For this reason, we need some criteria that will reflect this issue. The criteria that will be used for evaluating CODVerif should examine whether we address the aforementioned challenges. Therefore, we define three criteria for the qualitative challenges addressed by CODVerif and one criterion for the relevance of the CODVerif approach. We have four metrics (M1, M2, M3, and M4).

- M1: the number of simple rules that have been upheld.
- M2: the number of medium complexity rules that have been upheld.
- M3: the number of complex rules that have been upheld.
- M4: the V-Aiken provided by involved users regarding the questionnaire.

Evaluation Case

Our test sample comprises three groups of university master students who have joined a CRM course and who are to accomplish different exercises on various CRM features. The exercises required populating data in the CRM. To avoid introducing any bias when evaluating CODVerif, we have put no focus on following the naming convention. The subject was presented normally. The CRM exercises assigned to the students have avoided batch data insertion. We believe it is easier for users to follow the naming convention rules when they populate data in batch mode. The three groups, which do not know each other, were split into:

- The first group populated CRM data while they were asked to follow the naming convention in the exercise. However, CODVerfi workflow instances were not deployed. Therefore, we call this group *Group 1*.
- The second group of students, following another lecture about business process management, generated workflow instances using CODVerif prototype and deployed them into Salesforce to enforce the provided naming convention. We have assigned the generation of CODVerif workflow instances to this group to keep the two other groups unaware of the existence of these workflow instances. We call this group *Group 2*.
- The third group used Salesforce, and the deployed workflow instances to accomplish data insertion exercises. We call this group *Group 3*.

Evaluation Results

Group 1 and *Group 3* are composed of 15 master's students. Based on the analysis of the results that these students have provided, the evaluation metrics defined earlier were calculated and presented in Table 4.

Table 4. Evaluation of the implementation of the naming convention rules by Group 1 and Group 3

	The proportion of the total number of rules not satisfied for *Group 1* (no CODverif)	The proportion of the total number of rules not satisfied for *Group 3* (with CODVerif)
M1: simple naming convention and data integrity rules	**16.4%**	**4.9%**
M2:Medium complexity naming convention and data integrity rules	**45.1%**	**15.3%**
M3:Complex naming convention and data integrity rules	**60%**	**16.1%**

We assessed the statistical significance of our findings with hypothesis testing (Shi & Tao, 2009). Knowing that hypothesis testing aims to test the viability of the null hypothesis in the light of experimental data. In our case, all three hypotheses that were tested concern the parameters M1, M2, M3 detailed in Table 4, which are proportions. For this reason, a two-sample Z-test of proportion (Zou et al., 2003) was used to decide whether the null hypothesis should be rejected for a significance level. The p-value of the test has a central role in the decision regarding the rejection of the null hypothesis. A p-value is a measure of how much evidence we have against the null hypothesis (the smaller the p-value, the more evidence we have against it). In the following, for each finding, we also report the test results regarding its significance.

A first finding regarding simple naming convention rules (M1) are upheld easily by both *Group 1* and *Group 3,* even though *Group 3* performs better than *Group 1*. This finding could be explained by the pending notifications that oblige users to take the appropriate actions to eliminate them. The null hypothesis (H0) in this case assumes that M1_Group1 $\leq$ M1_Group3, while the alternative hypothesis (H1) is that M1_Group1 >M1_Group3. The results of the two-sample Z-test of proportion for metric M1 are depicted in the first column of Table 5. As seen in Table 5, the null hypothesis should be rejected (p-value < 0.01), and thus the alternative H1 is true. In other words, the observed difference between the two groups regarding the percentage of simple naming convention rules that have been upheld is statistically significant at the 0.01 level.

Table 5. Z-tests for the three hypotheses

Z-test	M1	M2	M3
Level of significance	**0.01**	**0.01**	**0.01**
Group 1			
Number of rules not considered	**47**	**88**	**63**
Sample size	**285**	**195**	**105**
Group 3			
Number of rules not considered	**14**	**30**	**17**
Sample size	**285**	**195**	**105**
Intermediate calculation			

Table 5 continued on next page

Table 5 continued

Z-test	M1	M2	M3
Level of significance	**0.01**	**0.01**	**0.01**
***Group 1* proportion**	**0.164**	**0.451**	**0.6**
***Group 3* proportion**	**0.049**	**0.153**	**0.161**
Difference between two proportions	**0.115**	**0.298**	**0.438**
Z-test statistic	**4.47**	**6.39**	**6.53**
***p* -Value**	$p < 0.00001$	$p < 0.00001$	$p < 0.00001$
Null hypothesis (reject/not reject)	**Reject H0**	**Reject H0**	**Reject H0**

The interpretation of this finding is as follows: for *Group 1*, students have not deemed it important to follow every simple naming convention rule, while *Group 3* students were driven by the CODVerif workflow instances, which allowed them to uphold more rules compared to their matching pairs in *Group 1*.

For M3, the same conclusion as M1 can be drawn, but here we see that the proportion of naming convention rules that were not upheld by *Group 1* is higher. Interpreting this is that complex rules are more difficult to implement. Nonetheless, the proportion of students who relied on CODVerif and did not uphold certain naming convention rules has also increased. Despite students observing CODVerif notifications pending regarding the naming of their data, they lacked time to complete the exercise.

CODVerif Utilization Evaluation

We have implemented a procedure to assess the usefulness of our CODVerif approach for data. To do so, we count on nine master's students in the CRM discipline (*Group 2*). These students have experience in defining CRM data for different purposes. Among these purposes is the export of CRM data for SOA-based applications. Additionally, these students have experience defining Salesforce workflows to automate tasks (Keel, 2016).

Over four weeks, we organized weekly sessions with the students of Group 2 to evaluate the CODVerif approach. First, we provided the students with information about our CODVerif approach, its goals, and the problem it aims to address. After that, we presented the generic scenarios described in Section 3 to the students so that they could assess their relevance. Next, we asked the students to implement the scenarios described and populated the CRM with customers' data to evaluate the relevance of CODVerif. When they had finished the definition of the different scenarios and the assessment of the errors left, we asked them to answer a series of questions (denoted A1–A7 in Table 6) about the validity and applicability of CODVerif and the results obtained in relation with their customers' data. Finally, the students of *Group 2* were required to assess each element with a value in the set {0, 1, 2, 3, and 4} where 4 represents the most positive possible feedback.

Table 6. Group 2 evaluation with V-Aiken

Element to assess	Average of the nine students assessment score	V-Aiken (M4)
A1. How far do you consider the problem populating data for SOA addressed in this paper worth researching?	**4**	**1**
A2. How far do you consider the usefulness of replacing a fine-grained naming convention with a set of instances of GMDIWs to ensure the coherency of data populated?	**3.88**	**0.97**
A3. How far do you consider the usefulness of performing the checking during the populating phase rather than after exporting the data?	**3.77**	**0.94**
A4. How far do you consider the complementarity between third-party verification software and the instances GMDPWs?	**2.77**	**0.69**
A5. How far do you consider the decrease of non-conformity errors raised during the usage of the data delivered?	**3.11**	**0.77**
A6. How far do you consider the simplicity of exchanging verification rules captured in the form of MDPW between partners in the extended enterprise (Figl et al., 2018)?	**2.44**	**0.61**
A7. How far do you consider the comprehensiveness of the set of GMDIWs that have been identified?	**3**	**0.75**

To summarize the global value given by the *Group 2* students for each assessed element, we used the **V-Aiken** statistic, a commonly used approach to summarize research relevance ratings obtained from experts (Lara et al., 2020). The following equation defines the formula used to calculate this statistic: $V = \sum_i S_i / n\left(c-1\right)$, where S_i represents the sum of the values a student provides to each assessed element, n is the number of students (9 in this case), and c is the number or categories to rate (5 in this case).

Figure 5. Group 2 evaluation V-Aiken plot

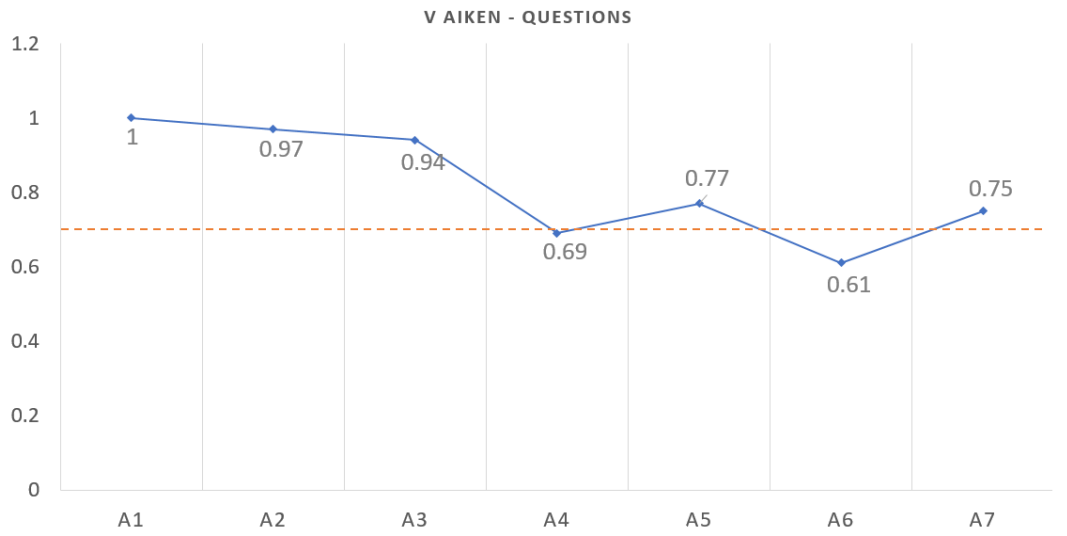


Figure 5 summarizes all these statistical findings graphically. Knowing that the statistical significance limit is 0.7 (obtained in the V-Aiken right-tail probability table (Lara et al., 2020) and considering the results obtained (Table 6), we can make the following conclusions:

- According to *Group 2* students, the problem addressed in this paper is worth researching; the method is appropriate, and the research questions are well-defined (all V-Aiken values are above the threshold of 0.7)
- The V-Aiken of A4 and A6 are below the threshold of 0.7. For A4, we can explain the score because third-party verification software performs more checks that could go beyond the standard and the naming convention rules. For A6, the score can be explained by the concentration of experts on their data, and they have not seen the usefulness of exchanging verification rules with other partners. Furthermore, each partner probably has their naming convention, and thus there is no need to exchange MDIW.

RELATED WORK

A data store that aims to define data for SOA applications (Hobsch & Schmidhuber, 2022) has been discussed in the literature (Eisenmann et al., 2015). However, this data store does not perform continuous verification of data. Instead, it highlights the attributes that contain an error for simple verification rules. Therefore, users can rely on the consistency check for complex verification rules to identify potential errors. However, with the large number of objects that could be managed in this kind of data store (hundreds of thousands of objects (Kazmirchuk, 2017)), the consistency check report could quickly become difficult to analyze and use.

The problem of defining data that upholds data coherency rules imposed by standards, naming conventions, or both is not limited to SOA applications, but has a larger scope. Wang et al. (2012) addressed the problem of defining data that upholds a naming convention for vehicular communication. They have shown the necessity of having coherent data with a naming convention. The observation made by Wang et al. (2012) is valid, but they have not provided a solution on how to impose this naming convention when populating the data store that contains the data of their vehicles. Paniagua et al. (2019) aimed to show how to use a naming convention to build a system of systems structure that relies on SOA. They defined the relevant characteristics of a naming convention: (i) Descriptive, (ii) Structured, (iii) Topology Informative, (iv) Security reinforce, (v) Useful for discovery, (vi) Unambiguous, (vii) Versatile. and (viii) Meaningful. The authors have clearly shown the difficulty of enforcing the definition of data that satisfies all the characteristics imposed by a naming convention. Paniagua et al. (2019) also addressed the naming of services in an SOA but not the naming of inputs and outputs of services in an SOA. Harding and Bayliss (2022) linked drug databases and naming convention policies to enforce interoperability between vendors, physicians, and pharmacists. Baijens et al. (2020) showed the importance of naming conventions for analyzing clinical trial data, which are managed in big data platforms. They noticed that equivalent concepts were named differently, which complexified the clinical trial data analysis. First, this shows the importance of upholding the naming convention rules when populating data stores. Second, this also shows the limits of the a-posteriori data verification, which led to complexified data analysis models.

Examples of naming conventions we can cite are Dhavle and Rupp (205) and Dumas et al. (2013). All cited works above define the rules on how data must be named. However, these rules are defined in a separate document from the data themselves. Except if manual checking or a-posteriori checking is performed, no automated mechanism guides or warns users who populate the data store to uphold the rules imposed by the naming convention.

Leopold et al. (2013) developed a technique to detect violations of naming convention rules when defining business processes. Authors have relied on natural language processing (NLP) techniques to check the names used when different languages can be applied. Although this contribution is

automated, it belongs to the category of a-posteriori verification approaches because data populated in the data store can be checked when the data definition phase ends.

We have also illustrated the importance of having a data store with data upholding the standard's rules or the naming convention in Alshreef et al. (2017). In this paper, the authors have defined access rights based on the name given to the data. They called their framework NC-RBAC (Naming Convention RBAC). The study developed in Alshreef et al. (2017) illustrates the importance of having data that uphold the naming convention and the standard rules. Besides the ambiguity and the wrong usage that can be made with data named wrongly, the access control based on the naming convention better motivates the CODVerif approach to impose the naming convention. In the same cyber security domain, we can cite the vulnerability databases (CNNVD.org.cn and NVD.nist.gov), which contain a variety of vulnerabilities, including different attributes such as the name and vulnerability priority. The data in these databases follow a naming standard, making vulnerabilities from completely different vulnerability databases available in the same standard. This standard facilitates sharing of vulnerability information (Jia et al., 2018).

O'Donovan et al. (2019) has defined basic naming conventions to differentiate between cyber-physical interfaces in a fog and cloud architecture. Similarly, García-Holgado and García-Peñalvo (2019) developed a model-driven approach to model technological ecosystems. First, they proposed a meta-model specific to technological ecosystems, and then they defined how it can be used through different phases to generate Platform-Specific Model (PSM) for developing technological learning ecosystems based on Open Source software. The interesting aspect of their work is how they raised the importance of the naming convention when building the meta-model for their domain. Indeed, they have proposed the naming convention associated with their model-driven approach. Nevertheless, like the other model-building and populating frameworks discussed above, their naming convention is just a recommendation, and nothing obliges users to follow its rules. This would generate models that would be difficult to read and interpret.

Jin et al. (2018) developed an approach to extract microservices and propose them to users by using their functionality instead of other criteria, including whether the definition of the services has followed a naming convention. However, as pointed by Jin et al., the limitation of relying on the naming convention to extract microservices consists in the absence of evidence that proves that the naming of a microservice, its inputs, and its outputs mean the same business capability requested by users. This risk is because of the lack of rigor when developing the naming convention rules and the lack of obliging developers to follow the naming convention rigorously. CODVerif aims to lessen that risk.

In software engineering, Mateo Navarro et al. (2016) proposed the S-DAVER framework to support run-time data verification when the data is populated in software application forms. S-DAVER allows developers to define verification rules on attributes. Additionally, S-DAVER keeps these verification rules separated from the software business logic code; thus, the rules can be updated if needed without affecting the features of the software embedding S-DAVER. Nonetheless, we can observe that two problems exist that are addressed by CODVerif while S-DAVER does not cover them. First, S-DAVER could help define simple verification rules. However, for more complex data insertion scenarios where it is necessary to guide users step by step to insert the data correctly, in this case, S-DAVER cannot address such scenarios. Second, S-DAVER checks the data only when the user finishes the insertion. Therefore, s-DAVER cannot implement a complex rule, as we showed earlier. Thus, with S-DAVER, users will notice an error but cannot manage it.

CONCLUSION

Populating SOA data in a data store could be constrained by standard and naming convention coherency rules to ensure that the SOA services will be invoked appropriately. This paper proposes CODVerif, a conceptual framework that helps impose data coherency rules continuously when populating SOA data in data stores. First, CODVerif delimits its application via an ontology related to the "monitoring

data insertion" domain. Then CODVerif shows how data coherency rules can be captured and enforced by defining monitoring data insertion workflows (MDIWs). CODVerif defines *instantiation* and *composition* operations on MDIWs to allow users to capture complex data coherency rules and enforce them continuously when populating data. Thanks to the delimitation imposed by the monitoring data insertion workflow ontology, CODVerif also defines the *negation* operation on MDIWs. This operation generates $MDIWs$ that aim to "keep" the data coherent when the inverse of populating data actions are carried out. With our experience, we have identified generic data insertion scenarios that need to be monitored (GMDIWs) to enforce data coherency rules. We successfully defined MDIWs for these scenarios and applied them to real-life examples in the CRM Salesforce domain. Students who applied CODVerif on this use-case have given positive feedback about CODVerif and how it maintains data coherently in the CRM by considering the provided naming convention. Even though the evaluation provided positive feedback about CODVerif and has shown that the problem of continuous verification of SOA data is worth researching, we still plan to apply CODVerif in real-life projects, and this application would probably allow us to extend the definition of the core components of CODVerif and make it more comprehensive.

REFERENCES

Acker, O., Gröne, F., Blockus, A., & Bange, C. (2011). In-memory analytics—Strategies for real-time CRM. *Journal of Database Marketing & Customer Strategy Management*, *18*(2), 129–136. doi:10.1057/dbm.2011.11

Alshreef, A., Li, L., & Rajeh, W. (2017). Naming convention scheme for role based access control in cloud based ERP platforms. In K. Kim & N. Joukov (Eds.), *Information Science and Applications 2017* (Vol. 424, pp. 84–93). Springer Singapore., doi:10.1007/978-981-10-4154-9_11

Baijens, J., Helms, R., & Iren, D. (2020). Applying Scrum in data science projects. *2020 IEEE 22nd Conference on Business Informatics (CBI), 1*, 30–38. doi:10.1109/CBI49978.2020.00011

Bizagi. (2022). *Bizagi web page*. https://www.bizagi.com/en

Dhavle, A. A., & Rupp, M. T. (2015). Towards creating the perfect electronic prescription. *Journal of the American Medical Informatics Association*, *22*(e1), e7–e12. doi:10.1136/amiajnl-2014-002738 PMID:25038197

Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2013). *Fundamentals of business process management*. Springer. doi:10.1007/978-3-642-33143-5

Eisenmann, H., Cazenave, C., & Noblet, T. (2015). RangeDB, the product to meet the challenges of nowadays System Database. In *The 9th ESA Workshop on Simulation for European Space Programmes*. ESA.

Figl, K., Mendling, J., Tokdemir, G., & Vanthienen, J. (2018). What we know and what we do not know about DMN. *Enterprise Modelling and Information Systems Architectures*, *13*(2), 1–16. doi:10.18417/emisa.13.2

Fischer, P. M., Deshmukh, M., Maiwald, V., Quantius, D., Gomez, A. M., & Gerndt, A. (2018). Conceptual data model: A foundation for successful concurrent engineering. *Concurrent Engineering*, *26*(1), 55–76. doi:10.1177/1063293X17734592

García-Holgado, A., & García-Peñalvo, F. J. (2019). Validation of the learning ecosystem metamodel using transformation rules. *Future Generation Computer Systems*, *91*, 300–310. doi:10.1016/j.future.2018.09.011

Harding, L., & Bayliss, L. (2022). *Salesforce platform governance method: A guide to governing changes, development, and enhancements on the Salesforce platform*. Apress. doi:10.1007/978-1-4842-7404-0

Hobsch, M., & Schmidhuber, M. (2022). Software and systems. In F. Sellmaier, T. Uhlig, & M. Schmidhuber (Eds.), *Spacecraft operations* (pp. 213–241). Springer International Publishing. doi:10.1007/978-3-030-88593-9_12

Jia, Y., Qi, Y., Shang, H., Jiang, R., & Li, A. (2018). A practical approach to constructing a knowledge graph for cybersecurity. *Engineering*, *4*(1), 53–60. doi:10.1016/j.eng.2018.01.004

Jin, W., Liu, T., Zheng, Q., Cui, D., & Cai, Y. (2018). Functionality-oriented microservice extraction based on execution trace clustering. *2018 IEEE International Conference on Web Services (ICWS)*, 211–218. doi:10.1109/ICWS.2018.00034

Kazmirchuk, P. (2017). Managing telemetry definitions on the fly. In *The 11th ESA Workshop on Simulation for European Space Programmes*. ESA.

Keel, J. (2016). *Salesforce.com lightning process builder and visual workflow: A practical guide to model-driven development on the Force.com platform*. Apress. doi:10.1007/978-1-4842-1691-0

Lara, J. A., De Sojo, A. A., Aljawarneh, S., Schumaker, R. P., & Al-Shargabi, B. (2020). Developing big data projects in open university engineering courses: Lessons learned. *IEEE Access: Practical Innovations, Open Solutions*, *8*, 22988–23001. doi:10.1109/ACCESS.2020.2968969

Leopold, H., Eid-Sabbagh, R.-H., Mendling, J., Azevedo, L. G., & Baião, F. A. (2013). Detection of naming convention violations in process models for different languages. *Decision Support Systems*, *56*(C), 310–325. doi:10.1016/j.dss.2013.06.014

Mateo Navarro, P. L., Ruiz, D. S., & Pérez, G. M. (2016). A lightweight framework for dynamic GUI data verification based on scripts. *Software Testing, Verification & Reliability*, *26*(2), 95–118. doi:10.1002/stvr.1579

O'Donovan, P., Gallagher, C., Leahy, K., & O'Sullivan, D. T. J. (2019). A comparison of fog and cloud computing cyber-physical interfaces for Industry 4.0 real-time embedded machine learning engineering applications. *Computers in Industry*, *110*(C), 12–35. doi:10.1016/j.compind.2019.04.016

ObjectManagementGroup. (2022). *Object management group business process model and notation*. https://www.bpmn.org/

Osorio, R. V., Lemos, J. P., Beech, T. W., Julian, G. G., & Chaumon, J.-P. (2006). SCOS-2000 Release 4.0: Multi-mission/Multi-Domain Capabilities in ESA SCOS-2000 MCS Kernel. *2006 IEEE Aerospace Conference*, 1–17. doi:10.1109/AERO.2006.1656141

Paniagua, C., Eliasson, J., Hegedus, C., & Delsing, J. (2019). System of Systems integration via a structured naming convention. *2019 IEEE 17th International Conference on Industrial Informatics (INDIN), 1*, 132–139. doi:10.1109/INDIN41052.2019.8972250

Shaalan, S. (2020). *Salesforce for Beginners*. Packt Pub.

Shi, N.-Z., & Tao, J. (2009). *Statistical hypothesis testing: Theory and methods*. World Scientific Publishing Company.

Tudorache, T., Vendetti, J., & Noy, N. (2008). Web-Protege: A lightweight OWL ontology editor for the web. *Fifth OWLED Workshop on OWL: Experiences and Directions.*

van der Aalst, W. M. P., & Pesic, M. (2006). DecSerFlow: Towards a truly declarative service flow language. *Proceedings of the Third International Conference on Web Services and Formal Methods*, 1–23. doi:10.1007/11841197_1

van der Aalst, W. M. P., Weske, M., & Grünbauer, D. (2005). Case handling: A new paradigm for business process support. *Data & Knowledge Engineering*, *53*(2), 129–162. doi:10.1016/j.datak.2004.07.003

Wang, L., Wakikawa, R., Kuntz, R., Vuyyuru, R., & Zhang, L. (2012). Data naming in Vehicle-to-Vehicle communications. *2012 Proceedings IEEE INFOCOM Workshops*, 328–333. doi:10.1109/INFCOMW.2012.6193515

Ye, J., Sun, S., Wen, L., & Song, W. (2008). Transformation of BPMN to YAWL. *2008 International Conference on Computer Science and Software Engineering*, *2*, 354–359. doi:10.1109/CSSE.2008.980

Yousfi, A., Bauer, C., Saidi, R., & Dey, A. K. (2016). uBPMN: A BPMN extension for modeling ubiquitous business processes. *Information and Software Technology*, *74*, 55–68. doi:10.1016/j.infsof.2016.02.002

Yu, L. (2014). *A developer's guide to the semantic web.* Springer. https://link.springer.com/book/10.1007/978-3-662-43796-4

Yuan, J. X. (2012). *Liferay portal systems development*. Packt Publishing.

Zou, K. H., Fielding, J. R., Silverman, S. G., & Tempany, C. M. C. (2003). Hypothesis testing I: Proportions. *Radiology*, *226*(3), 609–613. doi:10.1148/radiol.2263011500 PMID:12601204

Malik Khalfallah received a PhD in computer science Lyon 1 University in 2014. He is currently a research engineer. His current research interests include services mediation, interoperability of business processes in complex product development environments.

Parisa Ghodous is currently full professor in computer science department of University of Lyon I. She is member of LIRIS UMR 5205 (Laboratory of Computer Graphics, Images and Information Systems). Her research expertise is in the following areas: Interoperability, Web semantic, Web services, Collaborative modeling, Product data exchange and modeling and Standards, She is in editorial boards of CERA, ICAE, and IJAM journals and in the committees of many relevant international associations such as concurrent engineering, ISPE, Interoperability.