Refining Core Ontologies Through Executable Transformation

Ivars Blums[1,*,†], Hans Weigand[2,†]

[1]SIA ODO, Riga, Latvia
[2]University of Tilburg, The Netherlands

Abstract

Formal ontologies can remedy the ambiguity of narrative financial standards, yet their static nature creates a gap between specification and implementation. This paper presents a framework that transforms static UFO/OntoUML models into interactive, executable artifacts, serving as a validation tool for modelers, a communication bridge for stakeholders, and an executable specification for developers. This framework uses a technology-neutral Event Specification Table (EST) to generate executable logic from ontological patterns, demonstrated with the IFRS 15 standard. Unlike operational platforms, the resulting "living model" is a validation instrument designed to test and communicate the behavioral semantics of complex standards, closing the loop between theory and practice for a more robust development process.

Keywords

UFO, UFO-L, IFRS, COFRIS

1. Introduction

Social and financial standards, issued as narrative text, invite inconsistent interpretation due to inherent vagueness; formal ontologization offers a principled remedy. However, ontological models often remain static artifacts, creating a gap between formal specification, stakeholder understanding, and software implementation. This gap contributes to inconsistent applications and vulnerabilities like ambiguous or fraudulent records that exploit interpretive leeway.

The converged revenue recognition standards, IFRS 15 and ASC 606 [1], are a prime example of this challenge. Though principles-based by design, official Post-Implementation Reviews (PIRs) confirmed persistent 'application matters' [2], stemming not from textual ambiguities but from deep conceptual hurdles leading to widespread user misunderstanding. Difficulties in qualifying assets, identifying distinct performance obligations, and navigating principal-versus-agent assessments highlight a fundamental problem: a complex, principles-based standard in narrative form is difficult to internalize and apply consistently [3,4].

This paper argues that the solution is a formal, executable conceptual model. We introduce a framework that transforms a UFO-based core ontology into an interactive artifact. This 'living model' serves as a learning and validation testbed, allowing stakeholders to resolve misunderstandings by simulating scenarios and observing the direct, unambiguous consequences of the standard's logic. Unlike ERP systems or low-code platforms, the generated artifact is not an operational application but a model-driven validation and communication instrument, focused on exposing and testing the semantics of standards.

It provides (1) a validation testbed for modelers and standard-setters, (2) a communication bridge for stakeholders, and (3) a specification for implementers. The process of making the ontology executable also serves as a powerful catalyst for refining the core model itself. To guide our research, we pose the following research question:

ER2025: Companion Proceedings of the 44th International Conference on Conceptual Modeling: Industrial Track, ER Forum, 8th SCME, Doctoral Consortium, Tutorials, Project Exhibitions, Posters and Demos, October 20-23, 2025, Poitiers, France

*Corresponding author.
†These authors contributed equally.
ivars@odo.lv (I. Blums); h.weigand@uvt.nl (H. Weigand)
0000-0003-3405-0754 (I. Blums); 0000-0002-6035-9045 (H. Weigand)

RQ: How can a static, type-level OntoUML-based core ontology be systematically transformed into an interactive, executable artifact that makes its behavioral logic explicit and testable, thereby serving as an instrument for its own validation and refinement?

This paper follows a Design Science Research (DSR) approach, with the main contribution being an artifact: a transformation pipeline from OntoUML to SQL via the Event Specification Table (EST). The DSR cycle is reflected in the paper's structure: Section 2 introduces the ontological foundations for executable semantics. Section 3 (design) presents the financial reporting elements shaping the transformation requirements. Section 4 (artifact development) specifies the EST and transformation algorithms. Section 5 (evaluation) demonstrates the artifact's usability through illustrative contract lifecycle scenarios. Sections 6 and 7 interpret results and position the work, and Section 8 concludes.

2. Foundations: A UFO-B Toolkit for Executable Models

Our work is grounded in the Unified Foundational Ontology (UFO) [5], a formal theory of reality whose distinctions are made available for conceptual modeling through the OntoUML language [6]. While UFO provides a rich typology, our framework critically relies on primitives from UFO-B, the foundational ontology for event-driven systems [7], and the concept of *mutual activation partnership* [8]. Figure 1 presents a selection of key UFO-B concepts relevant to this paper.

The toolkit is composed of the following core concepts:

Social Agents («roleMixin») represent the actors in our domain (e.g., Enterprise, Person) capable of bearing intentions and participating in social relations.

Social Objects («relator») reify relationships between Social Agents, turning them into stateful "things" with their own identity and lifecycle. In our model, Contracts and Performance Obligations are Social Objects that act as the primary bearers of economic dispositions and qualities, a critical step towards making commitments executable.

Dispositions («mode») are intrinsic properties of objects representing a potential for change under certain conditions. Dispositions like Transfer Commitment and the reciprocal Right to Consideration inhere in Social Objects, formalizing contractual commitments as testable potentialities. Our algorithm operationalizes dispositions by modeling them as potential changes to **Qualities** («quality»), which are measurable properties of an object, like quantity or value.

Conditions («situation») are snapshots of reality that enable or activate a Disposition, representing the "if" part of "if...then" logic. In our model, situations are implemented as queries over the event log that act as formal guards.

Mutual Activation Partnership (MAP) **Social Relator** reifies a formal dependency between two or more Dispositions, stating they can only manifest their effects together. This concept is fundamental to modeling the reciprocity established by a Contract between the dispositions of Agents, ensuring that the right to receive value is formally linked to the obligation to provide it.

Institutional Events («event») are perdurants that reify actual changes in the institutional world, such as the creation, termination, or reclassification of a social object. As the sole mechanism of change and bearers of history, events are recorded as immutable facts in an event log. An event's primary function is to manifest a pre-existing Disposition, providing the causal link between a potentiality and its actualization.

Together, these primitives—Social Objects bearing Dispositions, activated by Conditions to manifest as Events—form a rigorous toolkit for building a conceptual model whose behavioral logic can be systematically transformed into an executable artifact.

3. COFRIS Ontology: Executable Model of Economic Exchange

Previous work [9-12] established the Contract as a static Social Object binding Social Agents and their Commitments, but the dynamic process of contract execution remained largely unmodeled. To bridge this gap and address persistent application challenges in standards like IFRS 15, we present a

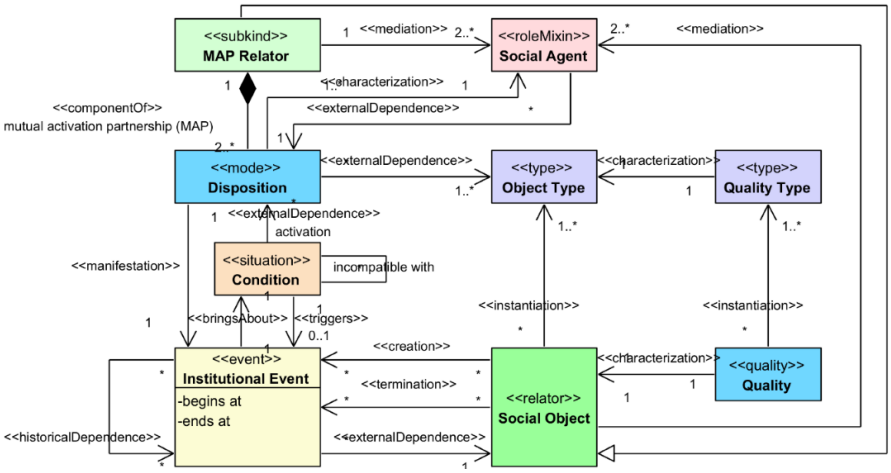

Figure 1: A UFO-B Toolkit for Modeling Institutional Systems. This diagram refines concepts from UFO-B [7], extended with our reification of Mutual Activation Partnership [8] as a MAP Relator.

refined version of COFRIS. Its core ontological commitment is to treat legal positions (e.g., obligations, rights) as dispositions and their contractual relations as Mutual Activation Partnerships (MAPs). This reframing enables an executable, event-driven lifecycle model initiated by an *Inception* event (see Figure 2), which creates the *Contract* as the smallest atomic unit of exchange of Performance Obligations in our executable model. To model this reciprocity, we adopt an *independent view* [13] of Party *A* and *B* playing *Transferor* or *Transferee* roles. The correlative counterparts (e.g., Performance Obligation-Right, Right-Obligation to Consideration, Transfer Commitment-Receipt Claim, Transfer-Receipt Event) are modeled as concept meta-properties, allowing consistent generation of *enterprise-specific views*.

At the heart of the Contract is a set of **Performance Obligation** MAP reciprocity relators, each representing a party's specific exchange of value. Each Performance Obligation relator creates a formal link between two core, interdependent dispositions («mode»s):

Transfer Commitment represents the performing party's one or more promises—dispositions specifying the Resource and Timing. When manifested by a Transfer event, its economic effect is characterized by simultaneous changes of equal amount in a Stock account (e.g., Inventory) and an Outflow account (e.g., Expenses).

Right to Consideration represents the performing party's reciprocal right to economic benefit. It is a disposition to receive a specific Amount of value, earned upon satisfaction of the corresponding Performance Obligation. When manifested by a Conditional Asset Recognition event, its economic effect is characterized by simultaneous changes of equal amount in a Stock Account (creating a Conditional Asset) and an Inflow Account (recognizing Revenue).

This relator pattern captures a crucial ontological commitment: value is a distinct relationship between bundled commitments and their corresponding value portion, allowing the model to track the allocation of the contract's total value, a central requirement of IFRS 15. The IFRS 15 Contract specializes this by binding the Seller and Customer roles and incorporating the Transaction Price.

Economic Events execute Contracts by manifesting the relevant dispositions, with Transferor, Transferee, and Economic Item - Resource and Claim participation.

Economic Item is a category that generalizes Performance Obligations and **Property Items** - in rem resources of the Market Participants.

A fundamental invariant of any valid Contract is the principle of reciprocal and indivisible exchange, captured by two formal constraints that act as preconditions upon Inception.

Axiom 1: Value Balance Constraint ensures equal total consideration of Performance Obligations and Rights.

$$\sum_{i=1}^{n} \text{Amount}(\text{Right to Consideration}_i) = \sum_{j=1}^{m} \text{Amount}(\text{Obligation to Consideration}_j) \tag{1}$$

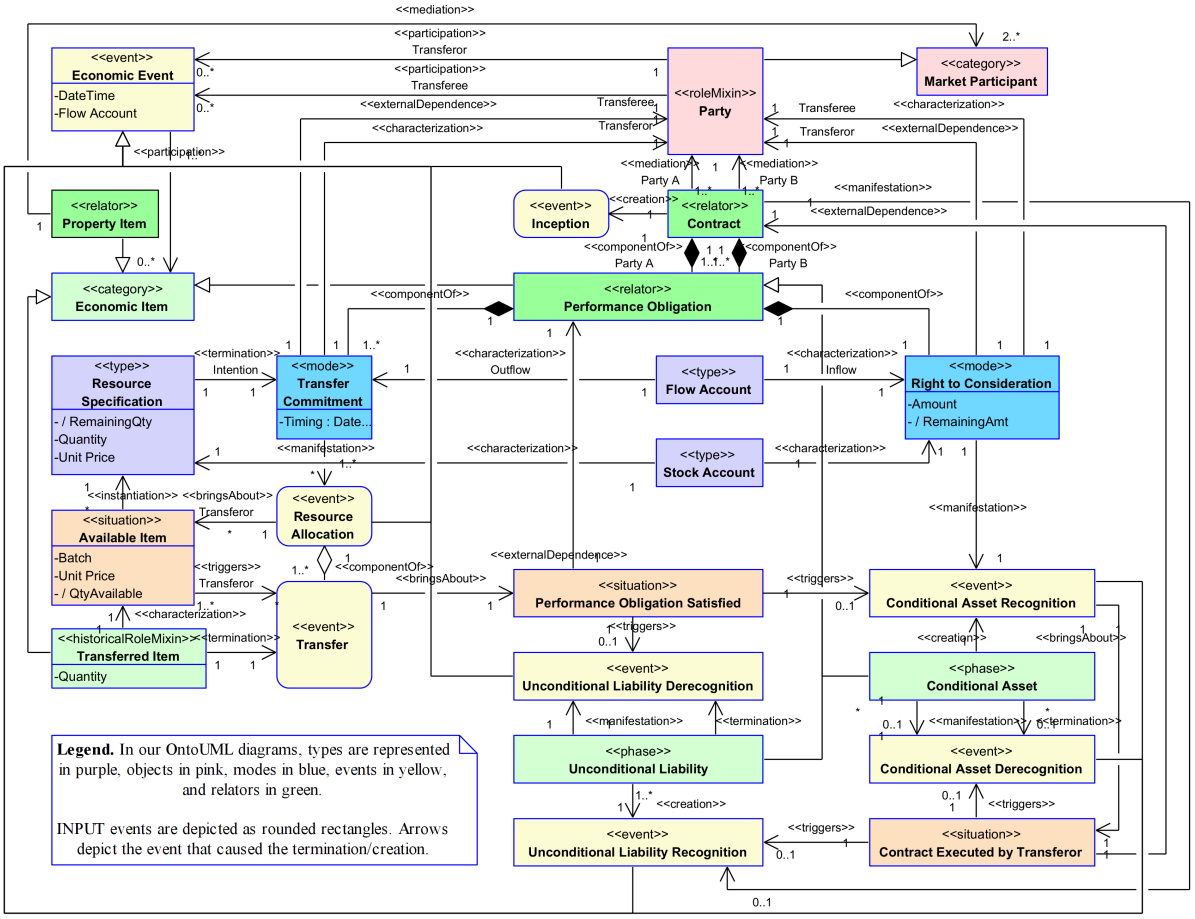

Figure 2: Relevant OntoUML-based fragment of interactive COFRIS Ontology. Independent view.

Axiom 2: Bundle Atomicity Constraint ensures the exchange is indivisible by stating that for any non-empty, proper subsets of each party's obligations, the value exchange is unbalanced.

$$\forall (A' \subset O_A, B' \subset O_B), A' \neq \emptyset, B' \neq \emptyset :$$
$$\sum_{i \in A'} \text{Amount}(\text{Right to Consideration}_i) \neq \sum_{j \in B'} \text{Amount}(\text{Obligation to Consideration}_j) \quad (2)$$

To govern the manifestation of dispositions, our framework employs a three-tiered model of qualities, acting as a validation funnel from the contractual to the actual, led by axioms:

Remaining Quality (The Contractual State): Represents the outstanding portion of a Disposition. It is governed by the Non-Overfulfillment Axiom (**Axiom 3**), ensuring the system never fulfills more than what was initially promised.

Available Quality (The Situational Constraint): Represents the portion of the Remaining Quality that is materially possible to fulfill, constrained by real-world situations (e.g., inventory) as governed by the Situational Constraint Axiom (**Axiom 4**).

Manifested Quality (Agent's Manifestation): Represents the quantity actualized in an INPUT event, governed by the Agency Axiom (**Axiom 5**), which ensures a user can only manifest what is available.

The contract lifecycle is driven by *Transfer* events manifesting Transfer Commitments, terminating resources of the Transferor, and creating resources of the Transferee. Fulfilling all commitments of a Performance Obligation brings about an *Obligation Satisfied* situation. This triggers a *Conditional Asset Recognition* event, transitioning the Performance Obligation relator into the *Conditional Asset* phase—an earned but not yet unconditional claim. Per IFRS 15, this is when a Contract Asset and Revenue are recognized.

Subsequently, if all of a Transferor's obligations are satisfied, a *Contract Executed by Transferor* situation occurs. This triggers events that reclassify the contract's balances, terminating both parties'

Contract Assets and transforming the Transferee's obligations into *Unconditional Liabilities*. Per IFRS 15, this is the crucial step where Receivables are recognized, as the right to payment is now only conditional on the passage of time. The contract is fully executed once all obligations of both parties are satisfied, with the final state resolved through *Unconditional Liability Derecognition* events.

4. Transformation Pipeline

Our pipeline for transforming a static conceptual model into an interactive, executable testbed is depicted in Figure 3. The framework is organized into three distinct phases: Modeling, Transformation/Generation, and Runtime.

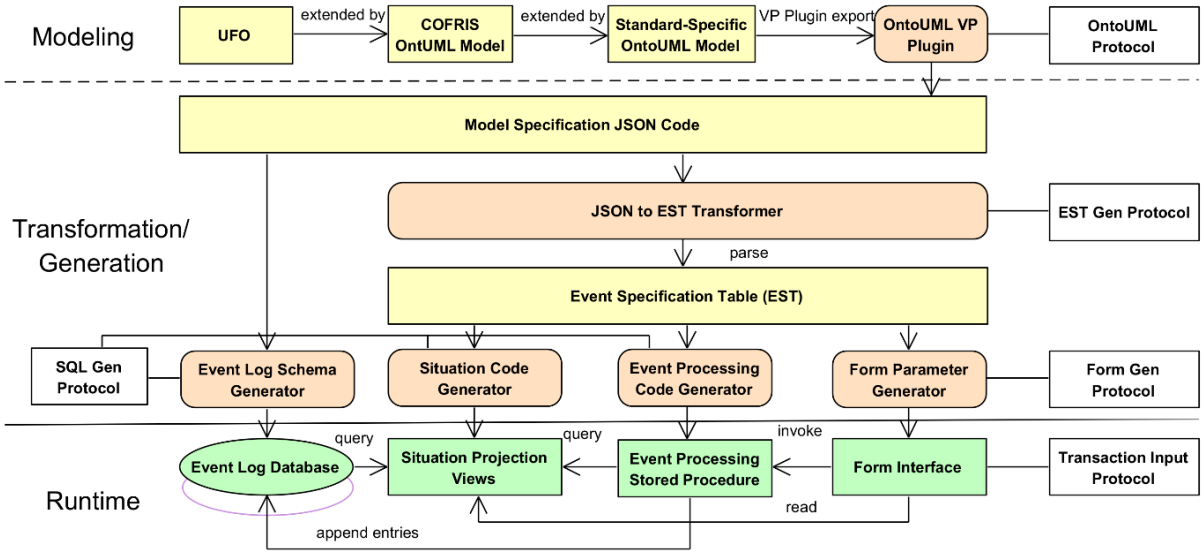

Figure 3: Model-to-Code Development Pipeline.

Modeling Phase: The process begins with a foundational ontology (UFO), which is extended and refined into a core economic exchange ontology (COFRIS), and then further refined to capture the specific rules of a normative standard (e.g., IFRS 15) in an OntoUML model. After syntactical checking, this model is exported into machine-readable Model Specification JSON Code.

Transformation/Generation Phase: The Model Specification JSON Code serves as the single source of truth for a set of generators. Our approach introduces a 'one immutable table per event log' transformation process for inheritance hierarchies, complementing traditional flattening and lifting strategies [14] for event-driven models. We also employ an 'attribute manifestation' technique, in which the attributes of dispositions are inherited or overridden by the events that manifest them. This approach also handles 'Dynamic Classification'—prevalent in OntoUML roles and phases—by leveraging the immutability of our database to query for the 'latest' phase or role of a kind.

Concurrently, a JSON to EST Transformer parses the JSON to produce a technology-neutral Event Specification Table (EST). This EST serves as a logical blueprint and can be interpreted as an FSM model of the application.

The EST then drives three parallel code generators: the Situation Code Generator (for Situation Projection Views), the Event Processing Code Generator (for the main Stored Procedure), and the Form Parameter Generator (for the Form Interface).

Runtime Phase: The generated artifacts form an interactive system. A user interacts with the Form Interface, which reads the current state from Situation Projection Views. Submitting an action invokes the Event Processing Stored Procedure, which appends entries to the Event Log Database. This update is immediately reflected in the views, closing the interactive loop.

Transformation Algorithms. Our framework transforms the OntoUML model into executable artifacts through a two-stage process detailed in Algorithm 1 and Algorithm 2, which operate on a

universal event structure. The core attributes are listed in Table 1.

Table 1
Core attributes used by the EST and EventLog.

Classifier (Key)	Description	Notes
EventSeq	A unique, monotonically increasing sequence	
EventType	The name of the event class E (e.g., "Transfer")	
DateTime	The timestamp of the event's occurrence	
Contract (CID)	The identifier of the Contract in which E occurs	
Transferor	The name of the Party taking obligation to perform or performing in this specific event	
Transferee	The name of the Party that is the beneficiary of the obligation or action in this specific event	
Performance Obligation (PID)	The identifier of the Performance Obligation E commits or manifests	Correlative: Performance Right
Transfer Commitment	The identifier of the Transfer Commitment E commits or manifests, opposite to the Right to Consideration	Correlative: Receipt Claim
Right to Consideration	The identifier of the Right to Consideration E sets or manifests, opposite to the Transfer Commitment	Correlative: Obligation to Consideration
Timing	The specific time-point or period relevant to the event occurrence	
Resource Specification	The name of the Resource (or Claim) Item being exchanged	Default: Money
Batch	The identifier of the resource receipt being consumed	
Quantity	The quantity of the resource involved	Default: 1
UnitPrice	The price, rate, or cost per unit of the resource	Default: 1
Amount	The total value of the event	Default: Qty * UnitPrice
StockAccount	The stock account name manifested or affected	Default: Inventory
FlowAccount	The flow account name manifested or affected	
RemainingQty	of a Transfer Commitment of the Resource Type	Derived
AvailableQty	of a Resource for Transfer in a Receipt	Derived
LatestPhase	of an Exchange Obligation	Derived
RemainingAmt	of a Right to Consideration	Derived
RemainingAmt	of a Contract for Transferor	Derived
DebitCredit (DC)	of a StockAccount	Derived

For each event, the EST specifies its canonical name, related dispositions, qualities, enabling situations, input attributes, effects on other objects (creation/termination), and triggered subsequent events.

Algorithm 1 in Figure 4 translates the relational semantics of the OntoUML model 'M' into the EST, systematically processing each event class to map its payload attributes, validation rules, and causal triggers. Algorithm 2 in Figure 5 acts as a code generator, translating the declarative rules in the EST into modular executable SQL artifacts, namely Projection Views and Event Handler Procedures. The procedures implement a Data Sourcing Waterfall to gather attributes and validate actions, and their reaction logic is generated from the EST's trigger specifications.

Journal Entry Screen Formation. The entry form, generated from the EST, is the primary interface for capturing, validating, and processing input. For events manifesting dispositions, the user selects the relevant disposition, determines available resources, chooses what to manifest, and initiates processing. System feedback, including journal lines and summary data, is then displayed. The form is composed of several dynamically populated segments derived from the model, such as the Disposition Hierarchy (Levels 0-3), lines for creating entries, and grids for displaying system-generated outputs. The form is generated from a view separate for each input event of the underlying model, in which model classes are mapped to form segments. See an example of three levels combined in one grid in Figure 6.

```
1.  Let DirectionMap = ConstructDirectionalGraph(M.diagrams).
2.  Let EST be an empty table with columns [Event, IsInput, SourceType, SourcePath,
    TargetField, ValidationRule, Triggers].
3.  Let Events = Find all classes in M where stereotype ="event".
4.  FOR EACH event E in Events DO
5.      // For each event, specify the source for every field in the EventLog payload.
6.      // Rule 1: Specify sources from the Disposition MAP.
7.      Let D = the <<mode>> or <<relator>> manifested by E.
8.      IF D exists THEN
9.          FOR EACH property P in the transitive closure of D's hierarchy
            (e.g., D -> Obligation -> Contract) DO
10.             Create a row in EST: Event=E.name, SourceType="Disposition",
                SourcePath=P, TargetField=MapToEventLog(P).
11.         END FOR
12.         Create a row in EST: Event=E.name, ValidationRule="AvailableQty <=
            D.RemainingQty".
13.     END IF
14.     // Rule 2: Specify sources from the enabling Situation.
15.     Let S = the <<situation>> that enables E.
16.     IF S exists THEN
17.         FOR EACH property P of S (e.g., UnitCost, AvailableQty) DO
18.             Create a row in EST: Event=E.name, SourceType="Situation", SourcePath=P,
                TargetField=MapToEventLog(P).
19.         END FOR
20.     END IF
21.     // Rule 3: Specify sources from the User Input.
22.     IF E has propertyAssignment "Category: INPUT" THEN
23.         FOR EACH field F provided by the user (e.g., ManifestedQty) DO
24.             Create a row in EST: Event=E.name, IsInput=true, SourceType="INPUT",
                SourcePath=F, TargetField=MapToEventLog(F).
25.         END FOR
26.         Create a row in EST: Event=E.name, ValidationRule="ManifestedQty <=
            AvailableQty".
27.     END IF
28.     // Rule 4: Specify sources from target Objects, Phases, and Qualities.
29.     LET O = the <<historicalRole>> or <<phase>> created or terminated by E,
30.         IF O exists THEN
31.            FOR EACH property of O DO
32.                Create a row in EST: Event=E.name, SourceType="Object",
33.                SourcePath=P, TargetField=MapToEventLog(P).
34.         END FOR
35.     // Rule 5: Specify Causal Reactions (Triggers).
36.     Let TriggerList = FindCausalChains(E, M, DirectionMap).
37.     Add a central row to EST: Event=E.name, Triggers=TriggerList.
38. END FOR
39. RETURN EST
```

Figure 4: Algorithm 1: Transformation to EST

5. The Framework in Action: Validation, Communication, and Implementation

Validation and Communication: Clarifying Ambiguities through Interaction. The primary role of our interactive artifact is to serve as a testbed for standard-setters, preparers, and auditors to resolve the persistent 'application matters' identified in the IFRS 15 PIRs. Instead of relying on textual interpretation, users can test their understanding against an executable implementation of the standard's logic.

The standard setting is a long and complicated process involving many people. We should also regard policy-setters for an enterprise as playing a similar role. The IASB for IFRS 15 received more than 1500 comment letters when developing the standard [2]. That process could involve more people and comments if the testbed for such a standard were available as a web application, whereby users would test their transactions and proposed results, instead of relying on texts only.

```
1.  Let V = GenerateProjectionViews(EST).
2.  Let H be an empty set of SQL Procedure definitions.
3.  FOR EACH unique EventType ET in EST DO
4.      Let E_Proc = Create a new procedure named "Process_" + ET.
5.      E_Proc accepts parameters relevant to its action.
6.      // Generate the Data Sourcing Waterfall by interpreting the EST.
7.      Let SourceRows = Find all rows in EST where Event = ET.
8.      Let JoinBlock = Generate SQL JOINs based on SourceClasses in SourceRows.
9.      Let SelectList = Generate SQL SELECT list based on SourcePaths in SourceRows.
10.     Let WhereClause = Generate SQL WHERE clause based on ValidationRules in
        SourceRows.
11.     // Generate the Primary Action (INSERT)
12.     Append "INSERT INTO EventLog (...) SELECT " + SelectList + " FROM " + JoinBlock
        + " WHERE " + WhereClause + " to E_Proc.
13.     // Generate the Reaction Logic (Direct Procedure Calls)
14.     Let Triggers = GetTriggersForRow(EST, ET).
15.     FOR EACH trigger T in Triggers DO
16.         // Generate the idempotent check based on T.condition_type
17.         Append idempotent "IF (Condition_Before = false AND Condition_After = true)
            THEN BEGIN"
18.             Append "EXEC Process_" + T.event + " @parameters...;"
19.         Append "END;"
20.     END FOR
21.     Add E_Proc to the set H.
22. END FOR
23. RETURN H (along with a main dispatcher procedure)
```

Figure 5: Algorithm 2: Transformation of EST to a Modular SQL Process Manager

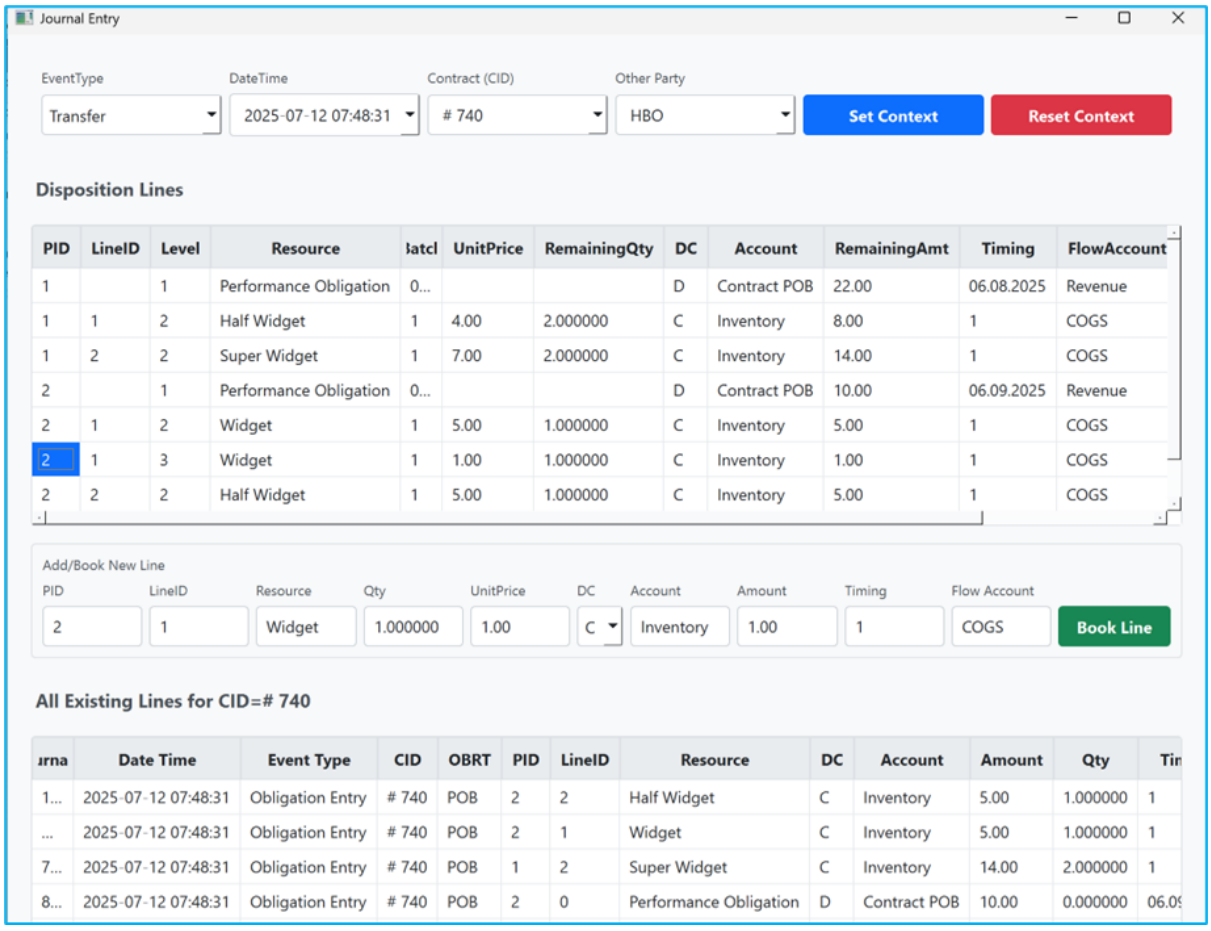

Figure 6: Example Transferor's Entry Form of Input events: The interface, generated from the EST.

A straightforward comparison between the OntoUML diagram, the entry form, and the resulting ledger provides a more immediate comprehension of the accounting model. Our current implementation focuses on prohibiting entries that violate the model, though it could be extended to analyze syntactically valid entries, for instance, via Process Mining. We demonstrate this with several scenarios from the PIRs.

1. Clarifying Scope and Constraints through Ontological Typing: Preparers often struggle with the scope of IFRS 15, as the term "ordinary activities" is not formally defined. Our solution enforces scope ontologically. For example, a user attempting to model the sale of old factory equipment (an IAS 16 transaction) in our tool would be presented with a dropdown list of valid resources for an IFRS 15 contract, such as Goods (Inventory) or Services. The resource Property, Plant, and Equipment would not be on the list, and the system's very structure—its types and constraints derived from the ontology—prevents the misapplication of the standard. Similarly, interactivity can clarify complex principal-versus-agent scenarios by formally modeling which party controls the underlying resource.

2. Clarifying "Distinct" Obligations by Visualizing Their Economic Effect: The concept of a "distinct" Performance Obligation (POB) is a cornerstone of IFRS 15, but is poorly understood. Determining if goods/services are "separately identifiable" is a major challenge, especially in software and SaaS contracts [2]. In particular, the standard is initially silent on the accounting effect of satisfying a non-distinct part of a distinct bundle. Other GAAPs and user practices introduce some intermediate accounts in this situation. A user models a SaaS contract as a distinct POB with two promises: (1) a software license and (2) critical, inseparable updates. The user wants to test what happens if they transfer the license on Day 1. The user selects the license for manifestation in the tool and executes a Transfer event for it. Because the ontology "knows" the updates are inseparable, it does not trigger Revenue. Instead, it only generates a journal entry for Expenses that can be the only result, as we can conclude from further reading the IFRS 15.98–103: "If the contract is terminated, ..., all costs associated with the partially satisfied POB are recognized as a loss." The user sees the economic substance. Transferring a non-distinct part of a bundle is not revenue; The interactive simulation makes it clear that only the satisfaction of the entire bundled POB can result in Revenue.

3. Clarifying the Lifecycle of Contract Balances: The distinction between a Contract Asset (an entity's right to consideration that is conditional on something other than the passage of time) and a Receivable (an unconditional right) is subtle and a frequent point of confusion for preparers. Our ontology models these as distinct «phase»s State Accounts of the contract's claim side, with clear state transitions triggered by events. A Receivable phase is typically entered when ALL POBs are satisfied – the Contract is executed by the Transferor. A user simulates a construction contract. Scenario 1 (Mid-project): The user executes a Transfer event for 50% of the project. Resulting Journal: DR Contract Asset, CR Revenue. The user sees a Contract Asset on the Ledger. Scenario 2 (Project Completion): The user executes the final Transfer. The ontology's rules know that at this point, the right to payment is now unconditional (dependent only on the passage of time as per the payment terms). The system automatically triggers a reclassification event. DR Accounts Receivable, CR Contract Asset. The user sees the dynamic lifecycle. By interacting with different scenarios (e.g., invoicing before performance), the user sees the dynamic lifecycle and builds an intuitive understanding that reading the standard alone cannot provide.

4. Improving on the Standard through Symmetric Modeling: IFRS 15 is written almost exclusively from the seller's perspective. This creates conceptual gaps, as noted by IFRIC 22 [4] regarding advance consideration, where the obligations of the payer are not fully explored. This one-sided view makes it difficult to model reciprocal exchanges like barter contracts coherently. Because our model is based on a universal Economic Exchange ontology, it is inherently symmetrical. A contract involves two parties, each with reciprocal Performance Obligations and Consideration Rights. The IFRS 15 Contract is simply a specialization where one party's POB is to deliver goods/services and the other's is (typically) to deliver cash. For a barter transaction whereby a company provides consulting services in exchange for advertising services, the user creates a single IFRS 15 Contract where: Party A has a POB to deliver Consulting Services. Party B has a POB to deliver Advertising Services. The framework handles this seamlessly. When Party A completes its consulting, it recognizes revenue. When Party B does, it recognizes revenue. The system correctly models the dual-sided nature of the exchange within a single,

coherent contract object.

Implementation: Executable Specification for Implementers. The framework's final function is to provide an unambiguous, executable specification for software developers, drastically reducing implementation risk. Instead of interpreting pages of ambiguous text, a developer is given two artifacts generated directly from the ontology: the EST and a reference T-SQL implementation (see Figure 7).

```
IF (SELECT RemainingAmt FROM dbo.LineID_Balance
    WHERE CID = @CID AND Transferor=@Transferor AND PID = @PID) = 0
BEGIN
    INSERT INTO dbo.journal(DateTime, EventType, CID, Transferor, Transferee, PID, Amount, DC,
    StockAccount, FlowAccount)
    SELECT @DateTime, 'Conditional Asset Recognition', CID, Transferor, Transferee, PID, Amount,
    DC,'Conditional Asset', FlowAccount
    FROM dbo.PID_Balance
    WHERE CID=@CID AND PID=@PID AND Transferor=@Transferor AND StockAccount='Performance Obligation';
    IF NOT EXISTS
    (SELECT 1 FROM dbo.PID_Balance WHERE CID=@CID AND Transferor=@Transferor AND RemainingAmt <> 0)
    BEGIN … Logic for Contract Fulfilled by Transferor would be triggered here
```

Figure 7: A fragment of generated (but reformatted) SQL code of a Conditional Asset Recognition.

6. Discussion - From Types to Instances, From Static to Executable

In COFRIS, the contract is modeled as an MAP relator of mutual performances, containing nested MAP relators that link each performance obligation to its consideration value. This structure clarifies reciprocity both at the whole contract and at the level of each individual obligation. A central difficulty we faced was that institutional events like *inception* or *transfer* cannot be treated as additional mere classes in a diagram. Their semantics arise only in execution: who initiates them, with what multiplicities, what scope they affect, and how their effects propagate. A single transfer entry, for example, updates the stock, the commitment, then the enclosing performance obligation, and eventually the contract as a whole. To capture this cascade, the modeler must mentally 'execute' the ontology, much like running a program, before validating it by actual instantiation.

This executional perspective clarifies why existing notations, such as BPMN, are insufficient. They specify control flow, but do not reify the institutional commitments and reciprocal structures that define contracts, focusing on task order rather than the evolution of obligations and rights. Our approach treats events as manifestations of dispositions within relators, ensuring that execution semantics remain consistent with institutional meaning.

The executable artifact generated from the ontology thus has a dual role. It can be used as a validation instrument, where ontologists, accountants, and standard-setters instantiate events to see how obligations and rights evolve over time. At the same time, the same transformation produces a generated application: a working database and event interface. Importantly, this is not an ERP system and is not designed as a generic low-code environment. Its primary aim is validation and communication, making the ontology's consequences visible, testable, and discussable. We argue that executable instantiation is essential for ontology validation. Static diagrams alone cannot reveal the dynamic implications of commitments and rights; only through execution can the semantics of standards like IFRS 15 be made unambiguous and communicable.

7. Related Work

Our work is situated within Ontology-Driven Development but diverges from several related streams. Frameworks such as Symboleo [15], while also grounded in UFO and UFO-L, generate executables from their own DSL rather than from OntoUML diagrams. This introduces different meta-properties and omits many UFO/UFO-B concepts central to our approach, including dispositions, MAPs, and relators.

Moreover, Symboleo emphasizes normative and legal contracts, whereas we target the economic and accounting dualities that underpin standards like IFRS 15. Our approach also differs from the general-purpose code generation from OntoUML, such as efforts to automatically derive database schemas [16], which concentrate on producing static application structures. By contrast, our goal is not to deliver an end-user system, but to create a run-time artifact for validation and communication of the ontology itself.

A recent project by Grievink et al. [17] also explored the automated transformation from OntoUML to Java class generation and UFO-A stereotypes, ignoring behavioral aspects, persistence strategies, and events. By contrast, our work is the first to address OntoUML execution from the UFO-B perspective, focusing on dispositions, events, and their institutional semantics.

8. Conclusion and Future Work

This research has shown that moving from a static core ontology to an interactive artifact is not a linear path, but rather a virtuous cycle. Making an ontology executable acts as a catalyst for refinement, compelling resolution of ambiguities that would otherwise remain hidden in static diagrams.

In response to our research question, we demonstrated how a core ontology of economic exchange, grounded in UFO, can be specialized to formally represent the IFRS 15 revenue standard and how this model can be transformed into an interactive artifact that clarifies ambiguities for diverse stakeholders. The framework not only validates, communicates, and specifies business rules, but also strengthens the ontology itself. By closing the loop between theory and execution, our approach supports the co-development of rigorous conceptual models and reliable systems.

This work also opens up several important questions for future research. We have shown that significant portions of a complex domain model can be made executable directly from its OntoUML specification. However, this raises the question of whether OntoUML is sufficient for generating a complete interactive artifact and what might be missing. For example, our current approach relies on the embedding of situational logic in names and descriptions and requires a custom generator to interpret them. A more robust solution might involve a dedicated constraint language complementary to the ontological model. This leads to a further research question: should such constraints be formally added to the OntoUML standard itself, or should they exist as a separate but interoperable layer?

Nonetheless, we have demonstrated that substantial generation is feasible. Defining the precise boundaries of this approach—and assessing whether it is sufficient from a full software requirements perspective—remains a promising area for investigation. Future work will extend standards coverage, enrich the executable solution with quantitative constraints, and integrate the framework with the IFRS Taxonomy (IFRSAT) to enable a fully machine-readable representation of accounting standards, including presentation and disclosure.

Declaration on Generative AI

During the preparation of this work, the authors used GPT-5, Overleaf, and Grammarly in order to: Grammar and spelling check. After using these tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] IFRS 15–Revenue from Contracts with Customers. IFRS Foundation (2014), London.

[2] International Accounting Standards Board (2024) Post-Implementation Review of IFRS 15 Revenues from Contracts with Customers. Project Summary and Feedback Statement, 30 September 2024. IFRS Foundation.

[3] Financial Accounting Standards Board (2024) Post-Implementation Review Report: Topic 606–Revenue from Contracts with Customers.

[4] IFRIC (2025) IFRIC 22–Foreign Currency Transactions and Advance Consideration. Accessed June 2025. https://www.ifrs.org/issued-standards/list-of-standards/ifric-22-foreign-currency-transactions-and-advance-consideration/
[5] Guizzardi, G. (2005) Ontological foundations for structural conceptual models. Ph.D. thesis, CTIT, Centre for Telematics and Information Technology, Enschede.
[6] Guizzardi, G., Almeida, J.P.A., Santos, V.A., Guizzardi, R.S.S. (2018) Endurant types in ontology-driven conceptual modeling: Towards OntoUML 2.0. In: Proceedings of the 37th International Conference on Conceptual Modeling (ER 2018), pp. 136–150. Springer, Cham.
[7] Almeida, J.P.A., Falbo, R.A., Guizzardi, G. (2019) Events as entities in ontology-driven conceptual modeling. In: Proceedings of the 38th International Conference on Conceptual Modeling (ER 2019), pp. 469–483. Springer, Cham.
[8] Baratella, R., Fumagalli, M., Oliveira, Í., Guizzardi, G. (2022) Understanding and Modeling Prevention. In: Guizzardi, R., Ralyté, J., Franch, X. (eds) Research Challenges in Information Science. RCIS 2022. LNCS, vol 446. Springer, Cham.
[9] Blums, I., Weigand, H. (2023) Consolidating economic exchange ontologies for financial reporting standard setting. Data & Knowledge Engineering 145, 102148.
[10] Nardi, J.C., Pires, L.F., van Sinderen, M., Almeida, J.P.A., Guizzardi, G., et al. (2015) A commitment-based reference ontology for services. Information Systems 54, 263–288.
[11] Griffo, C., Almeida, J.P.A., Guizzardi, G. (2016) A Pattern for the Representation of Legal Relations in a Legal Core Ontology. In: Proceedings of the 29th International Conference on Legal Knowledge and Information Systems (JURIX 2016), LNCS, pp. 191–194. Springer, Cham.
[12] Amaral, G.C.M. (2025) An Ontology Network in Finance and Economics: Money, Trust, Value, Risk and Economic Exchanges. LNBIP 532, Springer, Cham.
[13] ISO/IEC (2015) FDIS 15944-4: 2015. Information technology – business operational view – part 4: business transactions scenarios – accounting and economic ontology. ISO.
[14] Guidoni, G.L. (2023) Transforming Ontology-Based Conceptual Models into Relational Schemas. PhD Thesis, Federal University of Espírito Santo, Vitória, ES.
[15] Sharifi, S., Parvizimosaed, A., Amyot, D., Logrippo, L., Mylopoulos, J. (2020) Symboleo: Towards a specification language for legal contracts. In: 2020 IEEE 28th International Requirements Engineering Conference (RE), pp. 364–369. IEEE.
[16] Pergl, R., Xu, L., Almeida, J.P.A., Guizzardi, G. (2020) Automated Generation of Relational Database Scripts from OntoUML Conceptual Models. In: Proceedings of the 11th International Conference on Formal Ontology in Information Systems (FOIS 2020), pp. 241–255. IOS Press.
[17] Grievink, J. (2025) Ontology-Driven Software Development: Generating Java Code from OntoUML models. In: Proceedings of the 15th International Conference on Formal Ontology in Information Systems (FOIS 2025), pp. 166–180. IOS Press.