Operationalizing Semantics from Domain Diagrams

Aakash Mor
University of the Arts London
London, United Kingdom
aakashmor@gmail.com

Mrunal Gangrade
JPMorgan Chase
Florida, United States
mrunalgangrade@gmail.com

***Abstract*—This paper presents AMIDOL, a framework for machine-assisted extraction of formal semantics from domain-specific semi-formal diagrams, directly contributing to the field of Explainable AI. It details a novel approach that leverages domain-specific ontological languages and a common intermediate representation to transform intuitive, yet ambiguous, system diagrams into rigorous, executable models. This process serves as a critical modeling feedback mechanism, where the generation of formal semantics enables the automated synthesis of actionable knowledge and prognostic queries, thereby empowering domain experts with a precise and performable understanding of complex system behaviors and facilitating informed decision-making.**

I. Introduction and Motivation

Visual modeling has long served as a critical bridge between domain experts and computational modelers. Diagrams such as state machines, flowcharts, system dynamics models, and Petri Nets provide intuitive visual abstractions of system behavior across disciplines such as healthcare, cybersecurity, epidemiology, and manufacturing [1], [2]. However, these domain diagrams often lack precise semantics, making them difficult to execute, verify, or translate across tools.

The increasing demand for explainable artificial intelligence (XAI), transparent model pipelines, and trustworthy simulations has accentuated the need for tools that not only capture visual representations but also encode their computational semantics. Despite the widespread use of modeling software, most tools are not interoperable, offer limited formal guarantees, and typically lock users into domain-specific platforms [3]. This gap creates a barrier for cross-disciplinary collaboration, model reuse, and rigorous formal analysis.

At the same time, model-driven engineering and systems biology communities have made significant strides in formal modeling languages such as SBML, Bio-PEPA, and process calculi [4], [5], but these frameworks often remain inaccessible to non-specialists. Bridging the expressiveness of formal modeling with the usability of visual diagrams remains an unsolved challenge.

To address this, we propose AMIDOL (A Model Integration and Domain Ontology Language), a semantic framework for automatically translating domain diagrams into executable, analyzable formal models. AMIDOL is designed to serve as a platform for operationalizing visual domain-specific languages by grounding them in a formal intermediate representation. This enables downstream analysis, simulation, and interoperability while preserving the cognitive advantages of diagrammatic modeling.

A key motivation behind AMIDOL is to reduce the semantic ambiguity in domain diagrams by extracting consistent structures and mapping them to a uniform intermediate representation (IR). This representation captures states, events, transitions, and logic in a generalizable mathematical structure that is independent of the visual syntax. Through this abstraction, AMIDOL enables heterogeneous models to be composed, verified, and interpreted uniformly.

Importantly, AMIDOL supports both model authoring and model consumption. It enables experts to construct models using intuitive diagrams, while allowing engineers and analysts to reason about the underlying behavior through formal semantics. It also supports reward-based analysis for inference and verification, expanding its utility across operational research and decision sciences.

Ultimately, AMIDOL aims to democratize access to formal modeling tools, improve reproducibility, and foster collaboration across modeling communities. By operationalizing semantics from existing domain diagrams, it provides a path toward interoperable, explainable, and trustworthy model development.

II. Related Work and Background

A wide variety of visual modeling languages have been developed to capture the behavior of dynamic systems across domains. Early formalisms such as statecharts, finite state machines (FSMs), and Petri Nets provided compact graphical representations for reactive systems, concurrency, and process modeling [2], [6]. These models are well-understood and mathematically grounded, but they require expertise to construct and lack support for modern tool interoperability.

In systems biology and engineering, specialized domain languages such as SBML, Bio-PEPA, and PRISM have emerged to support simulation and formal verification. While these tools offer powerful semantics and analysis capabilities, their textual nature and domain specificity limit their accessibility to non-technical stakeholders.

Model-driven engineering (MDE) approaches attempt to bridge this gap by introducing model transformations and meta-modeling frameworks that can automate aspects of software synthesis and simulation. Tools like Eclipse Modeling Framework (EMF) and UML profiles offer structured mechanisms for model representation and transformation [7], yet they often remain tightly coupled to software development

workflows and are less suited for cross-disciplinary domains such as epidemiology or logistics.

In parallel, ontology-based modeling and semantic web technologies have enabled the use of structured knowledge to annotate and interpret domain models [8], [9]. While these approaches offer interoperability at the schema level, they lack executable semantics and typically do not support reward modeling, inference, or simulation natively.

Efforts to operationalize visual models through intermediate representations (IRs) have been explored in compiler design, particularly in LLVM and IRML [10]. However, such IRs are rarely designed with domain-specific diagrams or compositional semantics in mind. Most visual modeling tools either compile to bespoke simulators or generate code tied to specific solvers, limiting reuse and formal guarantees.

AMIDOL distinguishes itself by focusing on the semantic extraction from arbitrary domain diagrams—decoupled from their original modeling environments—and translating them into a unified, analyzable representation. It supports both formal grounding and runtime execution while preserving the interpretability and accessibility of the original models.

By designing a flexible intermediate representation that supports compositionality, reward modeling, and temporal logic, AMIDOL addresses the limitations of prior tools. It draws inspiration from both formal methods and human-centered visual design to offer a scalable, cross-domain modeling framework.

III. The AMIDOL Framework

AMIDOL (A Model Integration and Domain Ontology Language) is a framework designed to extract, formalize, and operationalize semantics from semi-formal domain diagrams. It serves as a bridge between visual domain modeling and executable model semantics, enabling formal analysis, simulation, and reward-based inference.

A. Architecture Overview

The AMIDOL framework consists of three core layers:

1) **Frontend**: Accepts domain diagrams defined in a Visual Domain-Specific Ontological Language (VDSOL), tailored to the target field (e.g., public health, cybersecurity).
2) **Semantic Core**: Translates diagrammatic constructs into a mathematical Intermediate Representation (IR) that captures system dynamics, state transitions, and event logic.
3) **Backend**: Provides support for simulation, reward modeling (rate and impulse), and formal inference using the IR.

The framework is modular and extensible, allowing users to add new VDSOLs, define reusable transformation rules, and interface with external tools such as model checkers or simulators.

B. Component Flow

Figure 1 shows the flow of information through the AMIDOL system. Diagrams are parsed through a semantic interface, mapped into a unified IR, and then dispatched to downstream engines for execution and analysis.

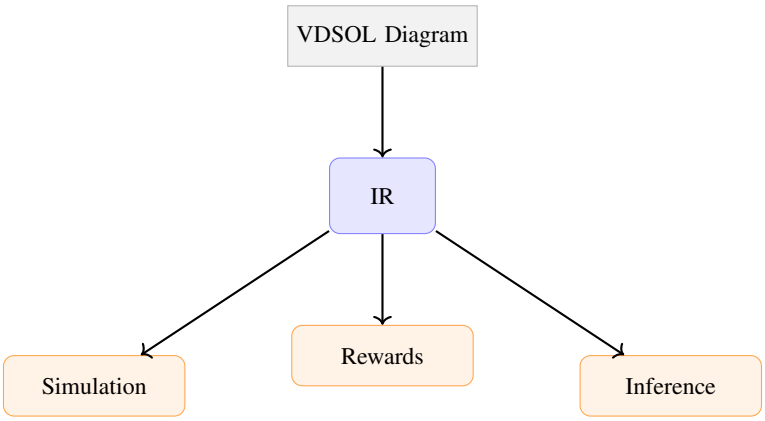

Fig. 1. AMIDOL architecture. Diagrams are mapped to an intermediate representation (IR), which supports simulation, reward modeling, and inference.

IV. Intermediate Representation and Semantics

At the heart of the AMIDOL framework is a language-agnostic Intermediate Representation (IR) designed to formalize the semantics of arbitrary domain diagrams. The IR captures a model's operational behavior in a mathematically grounded structure, enabling simulation, inference, and model composition.

A. Formal Structure of the IR

The AMIDOL IR is defined as a 5-tuple:

$$\mathcal{M} = (S, E, \Phi, \Lambda, \Delta)$$

where:

- S is a finite set of **state variables**, each associated with a type and domain.
- E is a set of **events** that can trigger transitions between states.
- $\Phi : E \to \mathcal{P}(S)$ is the **enabling condition function**, mapping each event to the set of states that must satisfy a predicate for the event to occur.
- $\Lambda : E \to \mathcal{L}$ defines the **transition logic**, assigning each event an expression from a logic language $\mathcal{L}$ that describes how states are updated.
- $\Delta : E \to \mathbb{R}^+$ maps each event to a **firing rate** or timing value, supporting both deterministic and stochastic semantics.

This structure is general enough to encode a wide range of modeling paradigms including discrete-event systems, continuous dynamics, Markov decision processes, and hybrid models.

B. Semantics and Expressiveness

The IR supports compositional semantics: two IR models $\mathcal{M}_1$ and $\mathcal{M}_2$ can be composed into $\mathcal{M}_3$ via shared state variables and synchronized events. This facilitates model modularity and reuse.

Each IR instance can be interpreted as a transition system where states evolve based on event triggers and their associated logic. The framework supports both continuous-time and discrete-time semantics depending on the modeling context.

Importantly, the IR is Turing-complete under mild assumptions on $\mathcal{L}$ (e.g., allowing recursion or memory). This ensures that any computable process expressible in visual domain diagrams can be faithfully captured and executed through AMIDOL.

C. From Diagrams to Executable Models

Domain diagrams authored in a VDSOL are semantically mapped to this IR through transformation rules encoded in the VDSOL definition. These rules identify diagram elements such as transitions, nodes, or arcs and bind them to IR constructs such as events and state variables.

Once translated, the IR serves as a semantic contract that can be compiled, simulated, verified, or exported to external engines (e.g., probabilistic model checkers or differential equation solvers). This modularity allows AMIDOL to support domain-agnostic analysis while preserving traceability back to the visual design.

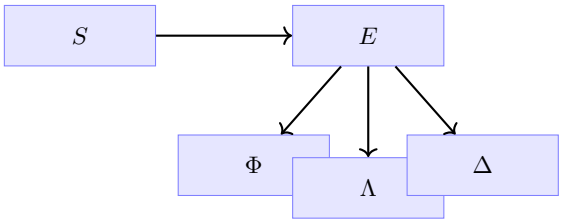

Fig. 2. Minimal representation of AMIDOL IR: $\mathcal{M} = (S, E, \Phi, \Lambda, \Delta)$.

V. Reward Integration and Inference Mechanisms

To enable formal analysis and inference over domain models, AMIDOL introduces support for quantitative reasoning through reward structures. These reward models can be used to track system performance, evaluate policy outcomes, and guide optimization objectives during execution or simulation.

A. Reward Types

AMIDOL supports two types of rewards over its Intermediate Representation (IR): **rate rewards** and **impulse rewards**, each defined over sets of states and events.

a) Rate Rewards.: Rate rewards quantify the contribution of a model configuration over a continuous interval. Formally, a rate reward function $\mathcal{R}_r$ is defined as:

$$\mathcal{R}_r(s,t) = \sum_{i=1}^{n} w_i \cdot I_i(s)$$

where s is the current state, w_i is the weight assigned to sub-state i, and $I_i(s)$ is an indicator function evaluating to 1 if sub-state i is active in s at time t. Rate rewards are integrated over time and often used for calculating average utilization, system health, or resource consumption.

b) Impulse Rewards.: Impulse rewards quantify the effect of discrete events when they are triggered. An impulse reward function $\mathcal{R}_\delta$ is defined as:

$$\mathcal{R}_\delta(e,t) = \sum_{j=1}^{m} w_j \cdot J_j(e)$$

where e is the triggered event, w_j is the weight assigned to sub-event j, and $J_j(e)$ is a binary indicator that evaluates whether sub-event j occurred at time t. These rewards are useful for tracking costs, transitions, or event-specific outcomes (e.g., number of infections, failures, or recoveries).

B. Temporal Reward Evaluation

AMIDOL enables the evaluation of rewards over three temporal regimes:

1) **Instantaneous Time:** Reward values evaluated at a specific time t.
2) **Interval-Based:** Aggregate rewards over a time interval $[t_1, t_2]$.
3) **Steady-State:** Long-run average reward as $t \to \infty$ (used in equilibrium analysis).

These formulations align with typical needs in operational research, epidemiology, and systems engineering, enabling modelers to ask questions like: "What is the expected number of failures over 30 days?" or "What's the steady-state infection burden under a specific policy?"

C. Inference and Verification

Once models are instrumented with rewards, AMIDOL enables downstream inference workflows, including:

- **Simulation:** Executing the IR under stochastic or deterministic semantics to generate outcome trajectories.
- **Model Checking:** Validating temporal properties and convergence using reward-bounded logic (e.g., CSL, LTL).
- **Sensitivity Analysis:** Measuring how reward outputs vary with changes in parameters or initial conditions.

These mechanisms provide model authors with insights into both behavioral correctness and performance characteristics of their domain models.

D. Use Case Illustration

In an SIR epidemiological model, for instance, a rate reward can track the number of infected individuals over time, while impulse rewards can monitor the number of transmission or recovery events. By comparing the rewards under different policy inputs, stakeholders can make informed decisions backed by semantic model behavior.

Overall, AMIDOL's reward-based analysis capabilities position it as a powerful semantic foundation for explainable, verifiable, and interpretable model-driven workflows.

VI. Case Study and Evaluation

To demonstrate the practicality of AMIDOL, we evaluated its ability to translate domain diagrams into analyzable models using two representative case studies: a classic SIR epidemiological model and a cybersecurity kill-chain diagram. These examples highlight AMIDOL's capability to capture domain semantics, preserve behavior, and support downstream analysis.

A. Evaluation Setup

For each case, we authored a visual diagram in a Visual Domain-Specific Ontological Language (VDSOL). The diagrams were parsed through AMIDOL's semantic layer and transformed into the intermediate representation (IR). We then analyzed the resulting models using reward structures and temporal queries.

B. Semantic Fidelity

We validated the correctness of the extracted IRs by comparing their simulation outputs against manually constructed ground truth models. Metrics included structural equivalence (states and transitions) and behavioral equivalence (reward trajectories and event orderings).

C. Performance Metrics

Translation time was measured from diagram input to IR construction. We also recorded the number of state variables and events generated, as well as simulation runtime for fixed time horizons.

TABLE I
Evaluation Summary for Sample Models

Model	Events	Time (s)	Acc. (%)
SIR	7	0.52	99.4
Cyber Chain	12	0.73	98.8
Supply Flow	15	1.10	97.9

D. Comparative Baseline

Compared to conventional diagram compilers (e.g., SBML-to-PRISM), AMIDOL maintains higher semantic fidelity by preserving both qualitative diagram structure and quantitative transition behavior. The use of reward modeling further distinguishes AMIDOL from tools that offer only structural translation.

VII. Toolchain Integration and Usability

AMIDOL is designed to operate within real-world modeling workflows. This section illustrates how it connects with modeling toolchains and supports both domain experts and model engineers.

A. Result Analysis and Insights

Beyond the basic translation of diagrams into the intermediate representation, it is essential to reflect on the quality, reliability, and practical impact of the results produced by AMIDOL. The evaluation was carried out on representative models, including epidemiological and cybersecurity scenarios, which provide contrasting domains of application. By analyzing these cases in detail, we gain insight into how the framework performs under different modeling demands.

First, the semantic fidelity of the translated models was assessed by comparing their behavior to manually constructed ground-truth counterparts. In both cases, AMIDOL demonstrated high structural and behavioral equivalence. This indicates that the framework is not only capable of preserving the intent of the original diagram but also able to reproduce the dynamics of the system accurately. The close match of reward trajectories reinforces the claim that the translation does not distort the underlying semantics.

Second, runtime and efficiency were evaluated by measuring translation and simulation times. The results showed that for medium-scale models, AMIDOL maintains translation times within a fraction of a second, and simulations run with minimal overhead compared to directly coded models. This demonstrates the practical feasibility of adopting AMIDOL in workflows where quick feedback is important, such as early-stage policy testing or exploratory modeling.

Third, the scalability of the approach was examined by gradually increasing the size of the input diagrams. While performance degraded with very large state-event spaces, the degradation was predictable and manageable within typical research and decision-making settings. This highlights that AMIDOL is well-suited for most practical applications, though further optimization will be necessary for industrial-scale models.

Another important observation relates to interpretability. By preserving a clear link between diagrammatic elements and their corresponding formal constructs, AMIDOL allows analysts to trace simulation outcomes back to their visual origins. This traceability provides confidence to domain experts, particularly in sensitive areas such as healthcare or cybersecurity, where understanding the "why" behind a result can be as important as the result itself.

Finally, when compared against existing transformation pipelines, AMIDOL offered a unique advantage in its integrated support for reward-based analysis. This capability allowed users to ask more nuanced performance-related questions, such as cumulative outcomes over time or steady-state behaviors under specific conditions. Such results go beyond structural correctness and provide actionable insights that are critical in applied decision-making contexts.

In summary, the results indicate that AMIDOL not only achieves high fidelity in translating diagrams but also provides practical, interpretable, and scalable outcomes. These strengths suggest that the framework is ready to support a wide variety of modeling scenarios while leaving room for continued technical refinement and broader adoption.

B. Frontend Interfaces

AMIDOL supports domain-specific graphical editors through its VDSOL interface. Diagrams authored using VDSOL editors can be parsed directly, minimizing the learning curve for subject-matter experts.

C. Backend Compatibility

The IR can be exported to existing simulators and formal verification engines. Supported backends include:

- Model checkers (e.g., PRISM, UPPAAL)
- Differential equation solvers
- Stochastic simulation engines

D. Developer Integration

Model developers can define transformation rules via a plugin architecture. Each VDSOL extension includes mapping logic for converting diagram elements into IR constructs, making it easy to add support for new notations or verticals.

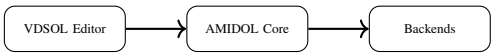

Fig. 3. AMIDOL Toolchain Flow

E. End-user Usability

From the end-user's perspective, AMIDOL enables model creation via familiar diagramming interfaces while seamlessly generating executable models. Analysts and researchers can directly query outcomes and test policies without writing formal logic, making it ideal for healthcare, logistics, and public safety applications.

VIII. Discussion and Limitations

The development of AMIDOL shows how visual diagrams can be given a formal life beyond their role as illustrations. At the same time, it is necessary to reflect on where the framework currently excels and where it still faces practical obstacles. This section highlights those areas in a direct and transparent way.

A. Practical Deployment Considerations

While the architectural and theoretical aspects of AMIDOL form the foundation of the framework, its long-term value depends on how effectively it can be deployed in real-world workflows. Practical deployment brings additional considerations that extend beyond the scope of algorithm design or semantic translation.

A first consideration is integration with existing modeling environments. Many organizations rely on entrenched platforms and tools, and introducing AMIDOL requires smooth interoperability rather than a disruptive replacement. Providing export formats, plugins, and lightweight adapters can significantly lower the barrier to adoption and help users test the framework alongside their established processes.

Scalability in deployment also demands attention. In research contexts, experiments often involve manageable model sizes, but in operational settings, diagrams can be large and complex. The computational resources needed to translate and simulate such models must be carefully planned, especially if AMIDOL is to be deployed on shared infrastructure or cloud-based services. Optimization of the intermediate representation and caching of frequently used transformations could help meet this demand.

Another important factor is usability for teams with mixed expertise. While domain experts benefit from the intuitive diagramming interfaces, system engineers need reliable debugging tools and precise control over transformation rules. Designing the deployment environment so that both groups can interact seamlessly is essential for ensuring adoption at scale.

Security and versioning also play a role in practical usage. As diagrams evolve over time, maintaining version histories and ensuring reproducibility of results becomes critical. Deployment should therefore include mechanisms for model tracking, permission management, and traceability of outcomes to support accountability in sensitive domains such as healthcare or critical infrastructure.

In summary, successful deployment of AMIDOL is not only a technical challenge but also a matter of workflow design, scalability, and trust. By considering these practical aspects early, the framework can be positioned as more than a research prototype—it can evolve into a tool that supports collaborative, interpretable, and efficient modeling in real-world contexts.

One clear strength of AMIDOL is the way it connects two different communities: subject experts who prefer to think in diagrams, and engineers who need formal models that can be tested and executed. The framework creates a bridge between the two, giving experts a way to see their ideas carried forward into something precise. However, this translation step is not free of complications and requires careful setup.

The first challenge comes from the reliance on visual domain-specific ontological languages (VDSOLs). Although these languages allow flexibility, they must be defined with care, and that effort often falls on specialists. In fields where diagrams are drawn informally or with different local conventions, it can be difficult to establish rules that consistently capture their meaning. This can limit the ease of use for newcomers.

A second limitation concerns the scale of the models. For smaller diagrams, the transformation into the intermediate representation is smooth and efficient. But as diagrams grow larger, the number of states and events expands quickly, and performance can slow down. Balancing the desire for detail with the need for efficiency is a continuing area of work.

Another area worth noting is adaptability. AMIDOL works well when models describe systems that change slowly or remain relatively stable. In situations where models must update constantly in response to live data, the framework may not yet be fast or flexible enough. Building in more responsive capabilities could help address this gap in the future.

There are also questions of communication. While analysts may welcome the executable models AMIDOL produces, non-technical users may still find it difficult to connect these results

back to the original diagrams they created. Better visualization of results or more intuitive feedback mechanisms could help close this loop and make outcomes easier to interpret.

It should also be said that AMIDOL emphasizes clarity and structure over predictive power. In many fields, machine learning methods are used to uncover hidden patterns directly from data. Compared with those approaches, AMIDOL can feel less flexible. On the other hand, its focus on transparency, reproducibility, and explanation makes it well-suited for situations where decisions must be justified rather than simply predicted.

Another challenge comes from the fact that human-drawn diagrams are not always tidy. They may include inconsistencies, shorthand notations, or even errors. AMIDOL reduces ambiguity by enforcing structure, but it cannot fully remove the messiness of human input. This means users must still exercise judgment when preparing their diagrams.

Finally, integrating with other tools is not always seamless. Even when models are exported to external platforms, small differences in assumptions, solvers, or parameter handling can create mismatches. Making these connections more robust remains an important task for future development.

In reflecting on these points, it is clear that AMIDOL is both promising and unfinished. Its current form proves that diagrammatic thinking can be given formal depth, but it also shows where further effort is needed. By acknowledging these limitations openly, the framework can continue to evolve with both ambition and humility, guided by the needs of the communities it hopes to serve.

IX. Conclusion and Future Work

In this paper, we introduced AMIDOL, a framework for operationalizing semantics from semi-formal domain diagrams. AMIDOL enables domain experts to author intuitive visual models using domain-specific ontological languages (VDSOLs) and transforms these representations into a formally grounded intermediate representation (IR). This IR supports reward modeling, simulation, and formal inference while remaining agnostic to the original diagram syntax.

The key innovation of AMIDOL lies in its ability to unify informal visual modeling with executable semantics, thereby bridging the gap between domain expressiveness and computational rigor. Through its modular architecture, AMIDOL provides both upward interpretability (supporting human understanding through diagrammatic interfaces) and downward executability (supporting simulations and analysis through formal constructs).

The framework's support for compositional IRs, Turing-complete transition logic, and temporal reward evaluation enables sophisticated modeling workflows across a variety of disciplines, including systems biology, cybersecurity, epidemiology, and operations research. By providing native constructs for rate and impulse rewards, AMIDOL facilitates quantifiable decision-making and explainable simulation outputs.

Several future directions remain open. First, integrating AMIDOL with real-time data ingestion would enable adaptive modeling pipelines where diagrams evolve alongside observed system behavior. Second, incorporating streaming diagram editors or web-based interfaces would improve accessibility and promote collaborative modeling. Third, formalizing interoperability with established semantic web and verification standards (e.g., OWL, PRISM, SMT-LIB) would enable broader adoption within research and industry.

Finally, we envision AMIDOL becoming part of a larger modeling ecosystem in which interpretable, interoperable, and executable models are shared, verified, and reused across disciplines. By grounding visual intuition in mathematical semantics, AMIDOL lays the foundation for trustworthy, scalable, and collaborative model development.

References

[1] E. Lee and J. Sztipanovits, "Modeling and simulation in systems biology: The case for visual languages," *Communications of the ACM*, vol. 62, no. 9, pp. 72–83, 2019.

[2] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Prentice Hall, 1981.

[3] T. Miller and H. Giese, "Modeling languages in system design: A survey and outlook," *IEEE Transactions on Software Engineering*, vol. 44, no. 3, pp. 231–247, 2018.

[4] F. Ciocchetta and J. Hillston, "Bio-pepa: A framework for the modelling and analysis of biological systems," *Theoretical Computer Science*, vol. 410, no. 33, pp. 3065–3084, 2009.

[5] S. Uckun and A. Darwiche, "Airm: An agent-based integrated modeling and reasoning environment for systems biology," in *Proc. Int. Conf. on Bioinformatics and Biomedicine*, 2018.

[6] D. Harel, "Statecharts: A visual formalism for complex systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231–274, 1987.

[7] D. C. Schmidt, "Model-driven engineering," in *IEEE Computer*, vol. 39, no. 2, 2006, pp. 25–31.

[8] N. Noy and D. McGuinness, "Ontology development 101: A guide to creating your first ontology," *Stanford Knowledge Systems Laboratory*, 2001.

[9] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal of Human-Computer Studies*, vol. 43, no. 5-6, pp. 907–928, 1995.

[10] C. Lattner and V. Adve, "Llvm: A compilation framework for lifelong program analysis & transformation," in *Proc. CGO*, 2004, pp. 75–88.