Digital Engineering and Digital Twin

Tech Science Press

https://doi.org/10.32604/dedt.2026.082492

ARTICLE

Experimental Frame–System Under Test (EFSUT): A Principled Foundation for Model Choice and Lifecycle Management in Digital Twins

Bernard P. Zeigler*

RTSync Corp., 6909 W. Ray Road, Chandler, AZ, USA

*Corresponding Author: Bernard P. Zeigler. Email: zeigler@rtsync.com

Received: 17 March 2026; Accepted: 30 April 2026; Published: 02 June 2026

ABSTRACT: As Digital Twin (DT) applications expand into complex, dynamic environments, a formal methodology is lacking to ensure that the embedded digital models remain adequate for specific stakeholder goals over time. This article introduces the Experimental Frame–System Under Test (EFSUT) methodology, providing a principled foundation for linking high-level stakeholder questions to the specific models capable of answering them. EFSUT organizes the digital engineering process around three core constructs: stakeholder questions, experimental frames that formalize observational requirements, and models related through morphisms. This structure allows developers to reason about model choice, reduction, and adequacy with technical rigor and transparency. To demonstrate the methodology's impact on lifecycle management, we present a DT architecture for an example ecological system. Because the system is subject to seasonal variability and structural drift, the architecture embeds the DT in a continuous lifecycle of sensing, prediction, drift detection, and recalibration. This practical application illustrates how EFSUT strengthens digital engineering practice—from initial design to long-term model assurance. We conclude by discussing future research in automated frame derivation, Model-based System Engineering toolchain integration, and scalable decision pipelines. Research is also needed for formal verification of frame–model applicability and scalable simulation-based decision pipelines.

KEYWORDS: EFSUT methodology (experimental frame–system under test); modeling and simulation (M&S) formalism; discrete event system specification (DEVS); model morphisms; experimental frames; digital twin engineering; digital twin lifecycle management; systems theory & engineering; model-based systems engineering (MBSE); model reduction

1 Introduction

Digital engineering and Digital Twin (DT) technologies increasingly demand a lifecycle-consistent structure that links model development, system verification, and operational decision-making. The shift from isolated "silos" to an integrated "digital thread" is a major focus as engineering organizations look to modeling and simulation to maintain lifecycle consistency. As such organizations expand their reliance on model-centric processes, the absence of a unifying methodological backbone often leads to fragmentation: models are built in isolation, testing workflows are improvised for each project, and decision-support analyses are constructed *ad hoc* [1]. These disconnected practices undermine traceability, reduce confidence in simulation-based evidence, and limit the reusability of digital-twin components across the system lifecycle [2–6].

1.1 The EFSUT Perspective as a Unifying Methodology

The *Experimental Frame–System Under Test* (EFSUT) perspective offers a principled alternative. Rooted in the Theory of Modeling and Simulation [7], EFSUT provides a coherent scaffold that ties stakeholder questions to observational regimes, model abstraction levels, test conditions, and decision-oriented outputs. Rather than treating modeling, testing, and decision support as separate activities, EFSUT organizes them within a single conceptual structure, ensuring that every simulation or analysis is grounded in explicit assumptions and traceable to stakeholder intent.

Experimental Frames (EFs) were introduced as formal specifications of the observational and contextual conditions under which models operate. The basic concepts underpinning the EFSUT methodology were developed with the theoretical framework of the organization of partial models—families of models each valid under specific frames [8]. This theory established EFs as the structuring principle for determining applicability of frames to models in relation to model simplification processes such as order reduction and abstraction. A subsequent ecosystem study demonstrated how frames form a partial order under a defined derivability relation, aligning model abstraction levels with the simulation experimental demands of frames [9]. Later work by Traoré and Muzy [10] formalized EFs as manipulable, hierarchical entities with behavioral semantics, enabling automated reasoning and compatibility checks [11]. Recent advances [12] positioned EFs as semantic anchors for model discovery, reuse, and curation creating an architecture where EFs bridge stakeholder objectives and model capabilities, supporting automated configuration of simulation workflows.

1.2 Distinct Uses of EFSUT Methodology

In the following sections we clarify three distinct uses of the EFSUT methodology that together support lifecycle-consistent digital engineering and digital-twin design:

- **Model Development:** using EFSUT to structure multiresolution model families aligned with observational commitments.
- **System Testing:** deriving test harnesses, admissible inputs, and validation scenarios directly from experimental frames.
- **Decision Support:** integrating EFSUT with simulation to enable branching simulations, uncertainty quantification, and evidence-based reasoning.

Together, these three types of uses position EFSUT as a unifying methodological scaffold for digital engineering and DT development, spanning model creation, system verification, and decision-based deployment. This article presents concepts, practical workflows, and examples to show how EFSUT strengthens digital-engineering practice throughout the product lifecycle.

1.3 Overview of the Digital Twin Illustration

To illustrate the methodology in digital engineering practice, the article develops a Digital Twin (DT) architecture for an orchard ecological system characterized by strong seasonal variability and ecological drift. Because the real orchard's dynamics change from season to season, a DT must continuously maintain correspondence with the real system to justify its predictions and support agricultural decisions such as predator introduction timing, barrier placement, or resource management. The architecture presented here embeds the twin in a cyclic workflow of sensing, prediction, drift detection, recalibration, and model assurance.

Finally, we conclude by discussing how research along these lines is a promising direction. Future work can explore automated experimental frame derivation and integration with MBSE toolchains. Research is also needed for formal verification of frame–model applicability and scalable simulation-based decision pipelines The result is a unified framework in which Experimental Frames guide the interpretation of data, morphisms justify model transformations, and applicability ensures that predictions remain grounded in the real.

2 Overview of the EFSUT Framework

This section describes the methodological approach used to formalize the three complementary uses of the EFSUT framework. The method integrates theoretical analysis, structural modeling, and cross-domain exemplification.

Before outlining the EFSUT framework, we review key background concepts.

An *experimental frame* is a specification of the conditions under which the system is observed or experimented with. As such, an experimental frame is the operational formulation of the objectives that motivate a modeling and simulation project. For example, out of the multitude of variables that relate to a forest, the set {lightning, rain, wind, smoke} represents one particular choice. Such an experimental frame is motivated by the interest in modeling the way lightning ignites a forest fire. A more refined experimental frame would add the moisture content of the vegetation and the amount of unburned material as variables. Thus, many experimental frames can be formulated for the same system (both source system and model) and the same experimental frame may apply to many systems. Why would we want to define many frames for the same system? Or apply the same frame to many systems? For the same reason, we might have different objectives in modeling the same system or have the same objective in modeling different systems.

Derivability: The degree to which one experimental frame is more restrictive in its conditions than another is formulated in the derivability relation. A more restrictive frame leaves less room for experimentation or observation than one from which it is derivable.

Let E and E′ be Experimental Frames. We say that E is *derivable from* E′ and write $E \leq E'$ if E can be obtained from E′ by a finite composition of:

- selection of compare variables,
- restriction of admissible input/output behaviors,
- coarsening of observable quantities.

Applicability: An Experimental Frame EF is *applicable* to a Model M if the EF's admissible input segments, output map, and validity conditions can be meaningfully interpreted on M. This requires compatible input/output ports and interpretable EF semantics.

Model Morphism: Let M and M′ be DEVS models. A *morphism* $M \to M'$ is a structure-preserving mapping that ensures the behavior of M′ simulates or approximates the behavior of M at a specified level of abstraction.

Fig. 1 shows how stakeholder questions flow into *experimental frames* (EF), which then branch into three distinct uses: (1) model development via EF-driven model hierarchies, (2) system testing via EF-derived test harnesses, and (3) decision support via simulation-based scenario exploration. All three uses remain anchored in the same EF definitions, ensuring lifecycle-consistent digital-engineering practice.

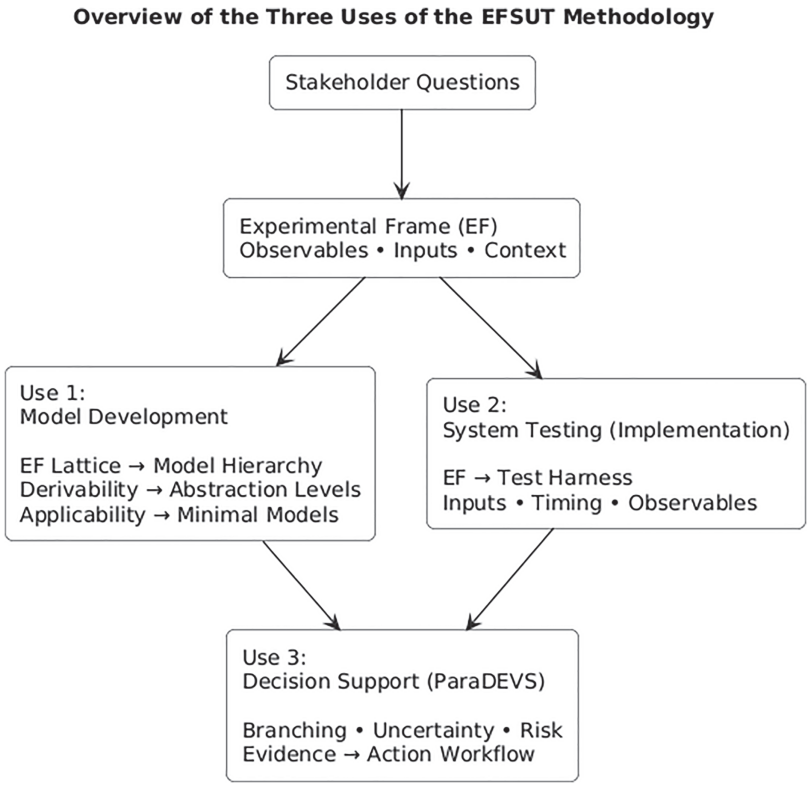

Figure 1: Overview of uses of EFSUT methodology: Three distinct uses of the EFSUT methodology together support lifecycle consistent digital engineering and digital twin practice: Model development, system testing, and decision support.

2.1 Conceptual Decomposition of EFSUT

We begin by decomposing the EFSUT methodology into its core components:

1. **Experimental Frames (EFs):** formal specifications of observational regimes, admissible inputs, and measurable outputs.
2. **System Under Test (SUT):** the model, implementation, or digital-twin component evaluated under an EF.
3. **Decision Metrics:** the mapping from raw outputs to decision-oriented observables.

This decomposition provides the basis for distinguishing the three uses of EFSUT.

Next, we briefly overview the EFSUT development process. Fig. 2 displays the EFSUT development pipeline, showing how each phase feeds the next. The EFSUT methodology is a structured approach for building, selecting, and deploying models in support of defensible decision-making. In the following we provide an informal overview of the methodology while the full definition will be available in (Zeigler in preparation). Each stage of the development corresponds to a distinct phase in the transformation from stakeholder questions to simulation-backed recommendations, with traceability and minimality enforced throughout. Referring to Fig. 2, we briefly describe the phases:

1. **Questions: Requirements Elicitation**

The process begins not with models, but with questions. Stakeholders articulate what they need to know—what must be observed, what decisions must be supported. These questions define the observational commitments, not the modeling assumptions.

2. **Experimental Frame (EF) Partial Order**

Each question induces an Experimental Frame (EF)—a formal specification of inputs, outputs, and validity conditions. These frames are organized into a partial order, expressing the derivability relationship where informally, one frame is derivable from another if the behavioral data associated with the first can be derived from the second. Roughly, this means that the first frame is less demanding in its data demands than the second. of refinement, aggregation, and selection. The partial order structure allows the system to determine which models are needed and how frames relate to one another.

3. **Model Family**

For each EF, the methodology selects (or constructs if not available) a minimally sufficient model capable of satisfying the frame's requirements. These models are organized into a family of models, structured by abstraction level, scope, and compositionality.

4. **EF → Model Applicability Mapping**

This is the heart of EFSUT: each EF is related to the models that can answer its motivated questions. This mapping ensures that every model is traceable to its observational justification and the subset of models provides a basis for choosing a model to simulate based on the accuracy/computational complexity tradeoff.

5. **Mission-Thread Construction**

A mission thread is a sequence of EF–Model pairs, each step:

1. Selects the model via applicability mapping
2. Executes the model
3. Evaluates the output
4. Informs the next EF
5. Continue until decision criteria are satisfied

This produces a transparent decision chain, where each inference is grounded in simulation and justified by minimal modeling. This structured approach to decision support is built from EF-driven model execution.

6. **Decision Support Output**

The output is a defensible recommendation supported by EF definitions, model applicability, simulation results, and mission-thread traceability. This process guarantees that decisions are model-justified, frame-aware, and always traceable to their data and modeling sources.

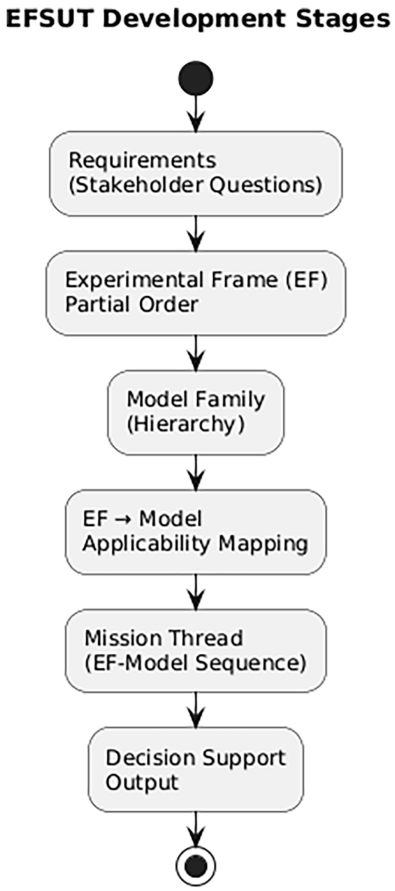

Figure 2: EFSUT development stages: Requirements → EF partial order → model hierarchy → applicability → mission thread → decision. Each arrow represents a justified transition, and each box enforces minimality, traceability, and compositional clarity.

2.2 Construction of Relations on EFs and Models

2.2.1 Construction of EF Partial Orders

We construct EF partial orders using the *derivability* relation mentioned above, defined by selection (variable or spatial subset). The partial order structure is used to identify model abstraction levels, test coverage requirements, and decision-relevant observables.

2.2.2 Mapping EF Partial Orders to Model Families

For each EF, we identify the minimally sufficient model capable of answering its questions. This process results in a model family that is aligned with the EF partial order. The family may contain models at various levels of resolution and computational complexity. The applicability relation determines which models are appropriate for each frame.

2.2.3 Deriving Test Harnesses from EFs

To formalize EFSUT's role in system testing, we derive a set of admissible input sequences, establish specific timing constraints, identify expected observables, and define validation criteria directly from the EF

definitions. This comprehensive approach ensures that testing remains traceable to stakeholder intent and is consistent across all implementations.

2.2.4 Smulation-Based Decision Support

For decision-oriented applications, we integrate the EF partial order with simulation to support branching scenario exploration, uncertainty quantification and sensitivity analyses. This method ensures that decision metrics are grounded in explicit experimental frames and model outputs.

3 Illustrative Application Domains

To highlight the versatility of the EFSUT methodology, we briefly overview representative applications from three distinct domains. Each example illustrates how experimental frames can be systematically applied to support model construction, validation, and integration, showcasing the method's adaptability and effectiveness across various fields of engineering and science.

3.1 Ecological Multiresolution Modeling

The EFSUT approach is employed to coordinate models at multiple spatial and temporal resolutions, such as linking detailed individual-based models of species behavior with aggregate population dynamics models. Experimental frames are used to formalize observation scales, define relevant environmental scenarios, and ensure consistency between fine-grained and coarse-grained representations. This creates a family of models that can answer questions at different abstraction levels while maintaining traceability and coherence.

3.2 Naval Swarm Engagement Modeling

In this context, the methodology guides the development of simulation models for coordinated operations among fleets of autonomous vehicles or vessels. Experimental frames specify engagement scenarios, operational constraints, and metrics of interest (such as survivability or mission effectiveness). Model hierarchies constructed under these frames allow for rapid reconfiguration and validation of tactics, supporting both high-fidelity simulations and abstracted decision-support tools within the same workflow.

3.3 Financial Hybrid Risk Modeling

The EFSUT process is applied to hybrid discrete event/continuous models that capture the stochastic evolution of financial assets, for example using the Ornstein-Uhlenbeck (OU) process to represent mean-reverting factors. Experimental frames define market conditions, regulatory policies, and performance criteria, enabling systematic evaluation of trading strategies or risk management solutions. The approach supports composability and reuse, facilitating integration with larger financial analysis toolchains.

3.4 Transition to Detailed Ecological Example

Collectively, these case studies underscore the generality and broad applicability of the EFSUT methodology. By explicitly linking stakeholder objectives, experimental frames, model hierarchies, and validation mechanisms, the approach provides a unifying framework for simulation-based engineering across ecological, defense, and financial domains. In the sequel we focus on the ecological application to illustrate the concepts in more detail and to demonstrate application to DT design for dynamic environments.

4 Use Case 1: Multiresolution Model Family Development

The ecological system example presented here illustrates the first and foundational use of the EFSUT methodology: constructing a family of models from the bottom up by progressively introducing Experimental Frames (EFs) and selecting the minimal models capable of satisfying them.

In the EFSUT framework, model construction does not begin with a fully articulated system model. Instead, it begins with a question, which determines the minimal observational commitments required to answer it. These commitments are formalized as Experimental Frames, and each EF induces the construction (or selection) of the simplest simulation model that can satisfy its observational and behavioral requirements. As additional questions arise, new EFs are introduced, placed within the EF partial order, and paired with correspondingly richer models. In this way, the model family grows incrementally, justifiably, and traceably, with each step grounded in the mission-driven logic of the EFSUT methodology.

4.1 Ecological System Description

Reference [9] demonstrated the application of EF theory in multilevel, multiformalism modeling, using predator–prey mite experiments as a concrete case. *This chapter has been made easily available for convenient perusal for the interested reader*. Briefly described, the experimental universe consisted of small number of oranges embedded in a larger array of rubber balls, where the oranges constitute cells of food source and balls constitute empty cells that must be traversed in search of food. Populated by mite species, some of which prey on others, this universe becomes a microscopic representation of a complex ecological system. There are multiple scientific questions for such a universe that no single model can answer. Instead, a structured family of models—each aligned with a specific experimental frame—is required. The frames organize questions that motivate model development, how frames are organized by demands on data acquisition, and how models relate to frames. The entire modeling enterprise becomes coherent when these relationships are made explicit. Three diagrams (Frame Partial Order, Frame–Model Partial order, Best-Model Table) help visualize this structure.

4.2 Experimental Frames for the Ecosystem

In Fig. 3, each experimental frame, EF specifies the conditions under which a system is observed and the variables that can be measured. Frames encode the scientific questions being asked.

In the ecosystem example, frames vary along two axes:

- Spatial scope: global (all cells) vs. local (subset of cells)
- Level of abstraction (resolution): continuous counts vs. aggregated totals vs. state occupancy counts

For example, the most detailed frame, E_global, observes food, prey, and predator counts in every cell. More abstract frames observe only totals, or even just discrete occupancy categories. Each frame defines a set of possible data sets and determines which questions can be meaningfully posed. The Frame Hierarchy Diagram shows how these frames are organized into planes of abstraction.

Fig. 3 shows how to integrate multiple models into a coherent architecture by aligning them with the frames they serve. Instead of treating models as competing alternatives, we organize them into a structured, interoperable family, each justified by the observational constraints of its frame. This approach exploits multiresolution modeling and compositional simulation to support model-based systems engineering.

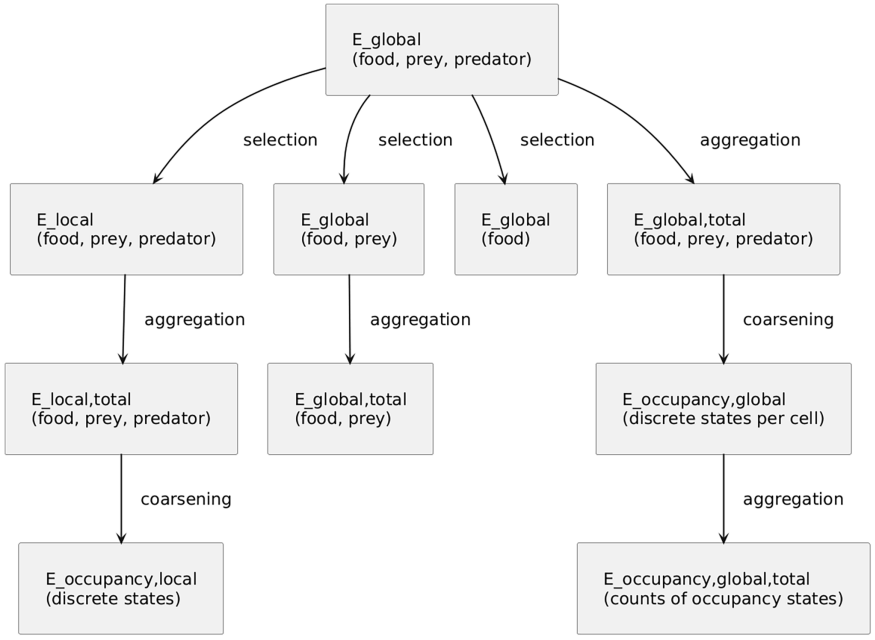

Figure 3: Experimental frame partial order for ecological system example.

4.3 Derivability Relation and Frame Hierarchy

As indicated above, frames are partially ordered by the *derivability* relation (≤). The ecology system application more particularly illustrates the constructive definition, EF1 is derivable from EF2 if the data observable in EF1 can be obtained from the data in EF2 by applying one or more of the following operations:

- **Selection**—This operation involves identifying and extracting a specific subset of variables or cells from the broader data set, allowing development to focus on particular aspects of the system under study.
- **Aggregation**—This process consists of combining or summing values of variables across multiple cells, resulting in a consolidated total that represents the overall state or quantity for the selected region.
- **Coarsening**—This method entails transforming continuous numerical counts into discrete, categorical groupings, simplifying the data by assigning it to specific categories rather than maintaining fine-grained measurements.

These operations always reduce information, so derivability flows from more detailed (highly resolved) to more abstract frames. This creates a partial order: global continuous frames at the top, occupancy-total frames at the bottom. Fig. 3 captures this structure and shows how resolution decreases, or abstraction increases, as you move downward.

4.4 Applicability Relation and Model Hierarchy

Each model is constructed at a specific level of resolution and expressed in a particular formalism (differential equations, discrete event, or hybrid) (Zeigler, Kim and Praehofer 2000). A model *accommodates* a frame if it can answer all questions posed within that frame without requiring information the frame does not provide. We often use the converse relation; a frame *is applicable to* a model if the model accommodates the frame.

This yields a second partial order:

- The **Base Model**, formulated using differential equations and the discrete event formalism, is designed to answer questions that require the highest level of spatial detail, and therefore accommodates only the most highly resolved frames such as E_global and E_local.
- The **Lumped Model**, which employs discrete event methods, is specifically constructed to address frames that aggregate totals across the entire system or local regions, such as E_global,total, providing a summarized representation rather than detailed spatial information.
- The **Occupancy Model**, also based on discrete event formalism, is tailored to frames that capture the occupancy status of individual cells throughout the system, such as E_occupancy,global, where each cell's state is represented as a discrete variable.
- The **Random Phase–Space (RPS) Model**, constructed with differential equations, applies to the most abstract frames, specifically occupancy-total frames like E_occupancy,global,total, where only the aggregated counts of occupancy states are represented, omitting any detail about individual cells.

In Fig. 4, the Frame-Model Partial order, frames are located on the left and models on the right. On the left, frames are ordered from most highly resolved (top) to most abstract (bottom):

The arrangement of frames and models follows a logical progression from the most detailed representations to the most abstract. At the top of the hierarchy, the frame labeled E_global provides full spatial detail, capturing the entire system at its most granular level. Just beneath this, E_local focuses on a subset of cells, narrowing the spatial resolution to specific regions within the system. Moving further down, E_global,total aggregates totals across the entire system, offering a simplified, summarized view. The E_occupancy,global frame introduces discrete states per cell, representing the occupancy of each cell individually. Finally, at the most abstract level, E_occupancy,global,total records only the counts of occupancy states across the system, summarizing the discrete states without reference to individual cells.

Correspondingly, models are organized from the most detailed to the most abstract. The Base Model, utilizing differential equations and the discrete event formalism, is capable of accommodating the highest resolution frames, such as E_global and E_local, answering questions that require full spatial detail. The Lumped Model, based on discrete event approaches, is suited for frames related to aggregated totals, like E_global,total. The Occupancy Model, which also employs discrete event formalism, addresses frames with discrete occupancy states, such as E_occupancy,global. The Random Phase–Space (RPS) Model, constructed with differential equations, is matched to the most abstract frame, E_occupancy,global,total, where counts of occupancy states are sufficient and no further detail is needed.

The applicability arrows indicate the specific models capable of addressing each frame's questions.

- The Base Model is specifically designed to handle only the frames with the highest level of spatial detail, ensuring that questions requiring full system granularity can be answered accurately.
- The Lumped Model is suited for frames that aggregate data across the whole system or specific local regions, providing summarized results for totals at the global or local scale.
- The Occupancy Model is tailored to address frames that represent the status of individual cells throughout the system, focusing on the occupancy state of each cell as a discrete variable.
- The Random Phase–Space (RPS) Model is applicable exclusively to the most abstract occupancy-total frame, where only aggregate counts of occupancy states are considered and individual cell details are omitted.

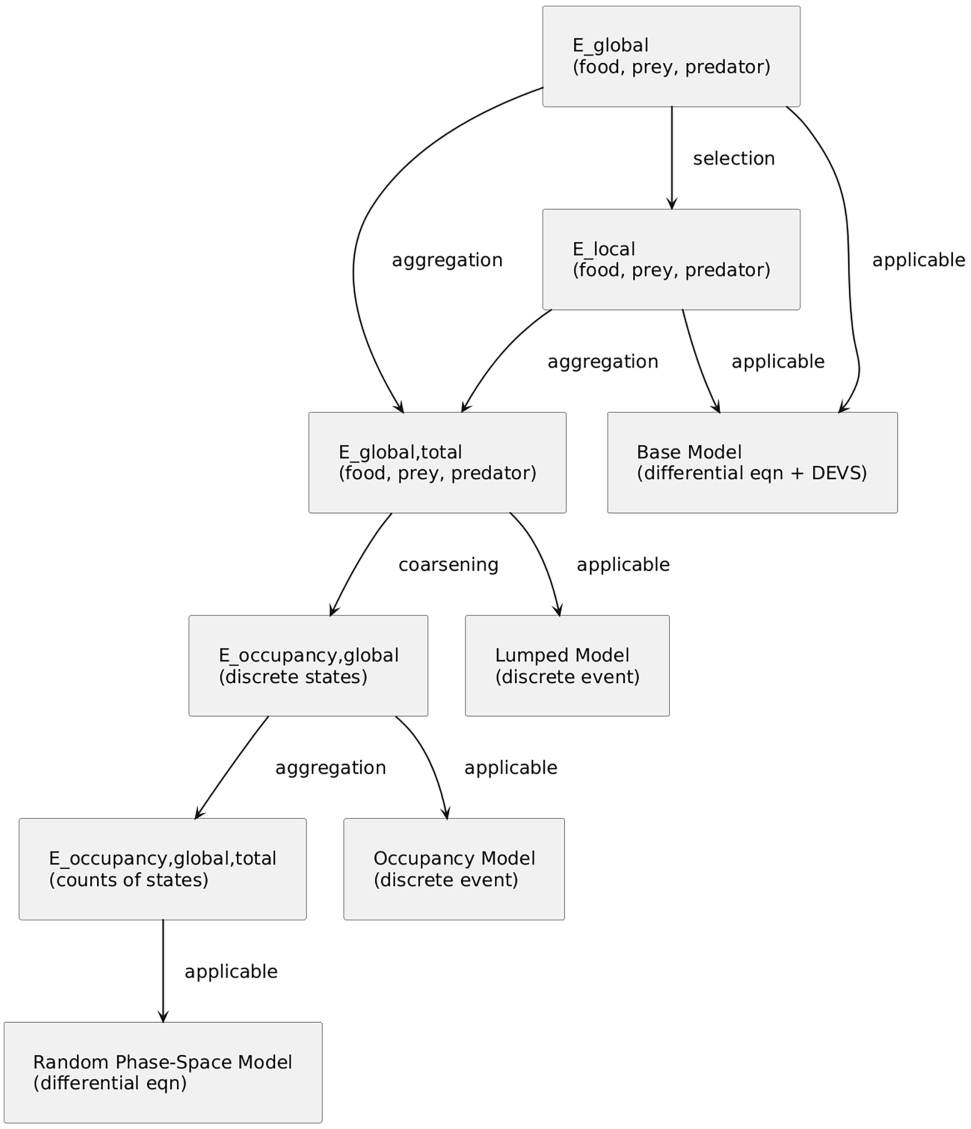

Figure 4: Derivability and applicability relations juxtaposed—derivability and applicability relations are labelled with "derivable" and "applicable" resp. Also shown are model morphic relations.

4.5 Best-Model Table and Minimal Model Selection

The applicability relation mirrors the derivability structure with Table 1 identifying an accommodating model for each frame.

Table 1 illustrates the applicability relation as employed to exemplify a representative applicable model for each frame. More than one model can be applicable, with the selection based on other desired criteria in an application instance such as computational complexity and accuracy.

Table 1: Applicability of experimental frames to models.

Frame	Description	Accommodating Model	Why This Model Accommodates the Frame
E_global	Full spatial detail: food, prey, predator per cell	**Base Model**	Preserves continuous local dynamics and cell-level migration detail required by the frame.
E_local	Detailed variables for a subset of cells	**Base Model**	Still requires spatially resolved continuous dynamics; lumped abstractions lose needed detail.
E_global,total	Totals of food, prey, predator across all cells	**Lumped Model**	Aggregation removes per-cell requirements; discrete-event lumped dynamics match total behavior.
E_occupancy, global	Discrete occupancy states per cell	**Occupancy Model**	Discrete-state representation aligns with coarsened per-cell occupancy categories.
E_occupancy, global,total	Totals of occupancy states	**Random Phase–Space (RPS) Model**	Aggregated occupancy counts correspond to deterministic differential equations over occupancy.

5 Use Case 2: Test Development

EFS and Models organized by derivability, applicability, and morphic relations enable the development of test harnesses. In the following we illustrate the development of one set for verification (is the model implemented right) and one set for validation (is the model valid with respect to the real system).

5.1 Verification Test Harness

A verification test harness for the ecosystem models is designed to determine whether the Lumped Model correctly implements the behavior derived from the Base Model when interpreted under the frame (E_global,total), which aggregates food, prey, and predator counts across all patches. In this setting, the frame–model pair consists of the aggregated observational frame and the Lumped Model, a discrete-event abstraction of the Base Model. The harness includes a generator that emits initial conditions matching Huffaker's aggregated universe setups, an acceptor that records total prey and predator counts at each event time, and a transducer that compares the resulting trajectories with precomputed summaries from the Base Model, such as expected oscillation patterns or extinction thresholds. Verification proceeds by treating the Base Model's Lotka–Volterra trajectories as ground truth, running the Lumped Model under identical initial conditions, and checking whether its event-driven updates reproduce the expected aggregate dynamics within acceptable tolerance. Any deviations that exceed these bounds are flagged as implementation errors, indicating that the abstraction has not been correctly realized.

5.2 Validation Test Harness

A validation test harness addresses a different question: whether the Occupancy Model is valid with respect to the real ecological system when interpreted under the frame (E_occupancy, global, total.), which counts cells in discrete states such as empty, prey-occupied, or predator-occupied. The frame–model pair

here consists of the occupancy-based observational frame and the Occupancy Model, a discrete-event representation with simplified state transitions. The harness uses a generator that initializes patch states according to Huffaker's experimental layouts, including the number of oranges and the placement of dispersal barriers. An acceptor records occupancy transitions such as prey colonization or predator arrival, while a transducer computes persistence metrics—time to extinction, proportion of occupied patches, and related indicators—and compares them to empirical data. Validation uses historical occupancy records, as reference data. The Occupancy Model is run under matching conditions, and its behavior is evaluated for its ability to reproduce key ecological outcomes such as predator–prey coexistence or oscillatory dynamics. The model is accepted as valid when its behavior falls within observed ranges and preserves the qualitative patterns documented in the real system.

5.3 Role of EFSUT in System Testing

These two workflows serve complementary purposes. Verification ensures that abstraction steps, such as the transition from the Base Model to the Lumped Model, preserve the intended dynamics and are implemented correctly. Validation ensures that simplified models, such as the Occupancy Model, remain faithful to real-world ecological phenomena when interpreted under appropriate frames. Both rely on the structural guarantees provided by the EFSUT methodology: applicability ensures that each frame–model pair is well-formed, and morphisms justify the abstraction relations that underlie verification.

6 Use Case 3: Mission Thread (Decision Structure)

Farmers need a concise but comprehensive understanding of an orchard's mite conditions, beginning with an early and accurate assessment of infestation severity across the grove so they can detect emerging outbreaks and track how pest populations evolve over time. They also require clear identification of the specific trees or regions most at risk, enabling targeted and efficient intervention. Because weather strongly shapes mite dynamics, farmers depend on insight into how temperature, humidity, and other environmental factors may accelerate or suppress outbreaks, allowing them to make informed, climate-aware management decisions. In addition, they need practical guidance on selecting and timing interventions—whether chemical treatments or biological controls—to minimize both damage and resource use. Finally, they must understand how these management choices affect yield and cost so they can balance economic sustainability with effective, long-term pest control.

Fig. 5 presents a realistic decision process (mission thread) that addresses these questions and is developed based on ecological modeling and simulation of the form illustrated above. The Experimental Frames are outlined as follows:

Experimental Frames Referring to Fig. 5:

- **Global frame:** Focuses on capturing trends across the entire grove. This frame addresses broad, overall questions such as understanding whether there is a significant outbreak affecting the whole system or detecting large-scale changes in mite or predator populations.
- **Local frame:** Concentrates on identifying hotspots at the level of individual trees. This frame is used when farmers need to pinpoint which specific trees or regions within the orchard are most affected, supporting targeted interventions.
- **Intervention frame:** Designed for comparing different management strategies. This frame enables evaluation of various treatments or control methods, helping determine which intervention is most effective or efficient for controlling mite populations.

- **Environmental frame:** Examines how weather conditions, such as temperature and humidity, affect mite and predator dynamics. It allows farmers to assess the influence of environmental variations on outbreak development or suppression.
- **Dispersal frame:** Addresses the dynamics of how mites spread from tree to tree. This frame helps answer questions about the speed and pattern of pest dispersal across the orchard, supporting decisions on containment measures.
- **Predator introduction frame:** Focuses on the success of biological control efforts. It evaluates whether introducing natural predators is effective in reducing mite populations and helps predict the outcomes of such biocontrol strategies.
- **Decision metrics frame:** Integrates information on yield, costs, and sustainability. This frame supports high-level decision making by analyzing the overall impact of pest management choices on crop productivity and economic outcomes.

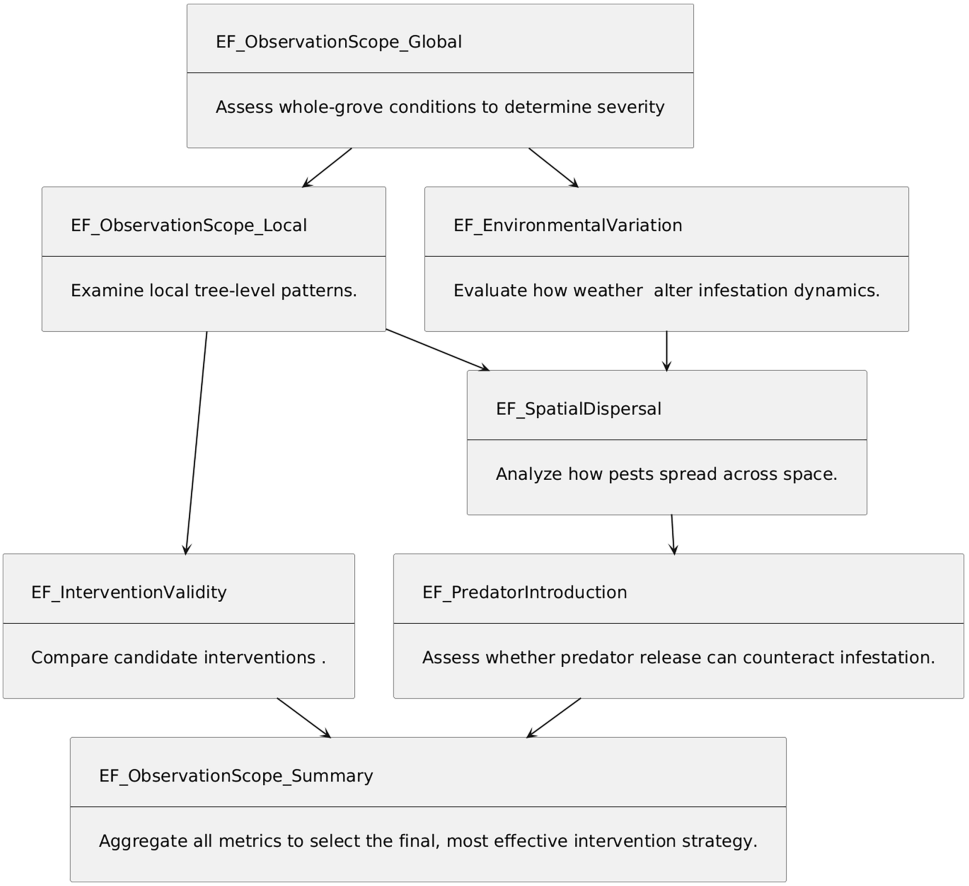

Figure 5: Experimental frame-based mission thread for ecology-aware agriculture.

The Models for that can accommodate these Experimental Frames are:

- **Mite and predator dynamics:** model the population dynamics of both pests and their natural enemies, capturing how their numbers change over time under different scenarios.
- **Weather effects:** incorporate the impact of environmental factors, such as temperature and humidity, on the development, survival, and spread of mites and predators.

- **Tree-to-tree dispersal:** track how mites move between individual trees, enabling detailed analysis of spatial spread and the identification of potential outbreak centers.
- **Intervention application:** allow simulation of various management actions, including chemical treatments or biological controls, to assess their effectiveness and optimize application timing.
- **Yield outcomes:** provide feedback on crop yield under different pest management strategies, helping farmers evaluate the economic and practical impacts of their decisions.

The Mission Thread (Workflow) that describes how a farmer uses the Experimental Frames is:

- **Global frame:** Helps the farmer assess the overall severity of the outbreak. By examining the situation at the orchard-wide level, the farmer can determine if the issue is minor or requires urgent, broad-scale action—essentially answering the question, "Is this outbreak serious?"
- **Local frame:** Focuses on identifying specific areas within the orchard where mite populations are concentrated. This frame enables the farmer to pinpoint hotspots and prioritize monitoring or treatments where they are most needed—answering, "Where are the hotspots?"
- **Intervention frame:** Allows the farmer to compare different management strategies, such as chemical sprays or biological control, to determine which is most effective for the current outbreak. This frame addresses the question, "Which strategy works best?"
- **Environmental frame:** Considers how changing environmental factors, such as an impending heatwave, might alter the dynamics of the outbreak or the effectiveness of interventions. The farmer uses this frame to anticipate and adapt to environmental changes—exploring, "How will the heatwave change things?"
- **Dispersal frame:** Examines how quickly and in what pattern mites are likely to move from tree to tree. This helps the farmer predict the potential spread of the outbreak and informs decisions regarding containment measures—answering, "How fast will it spread?"
- **Predator introduction frame:** Evaluates the likely success of releasing natural enemies (predators) to control the mite population. This frame helps the farmer anticipate whether biocontrol efforts will be effective—answering, "Will biocontrol succeed?"
- **Summary frame:** Synthesizes insights from all the frames to help the farmer develop the most effective overall management plan. By considering all factors, the farmer can decide on the best course of action—answering, "What's the best overall plan?"

Decision Making (Outcome)

- The stakeholder (farmer) can employ the decision structure (mission thread) in Fig. 6 to make decisions. As an example, a farmer might decide to release predator mites before an anticipated heatwave, using evidence from model predictions to stabilize the grove and prevent further spread of the outbreak. This proactive action is supported by transparent reasoning and measurable outcomes. The benefits of the decision process are summarized as:
- **Transparent, traceable steps from farmer's needs to model evidence:** The decision structure outlined in Fig. 6 enables the farmer to follow a clear, logical workflow. Each step in the process—from identifying the farmer's specific needs and questions to gathering and interpreting model-based evidence—is documented and easy to follow. This transparency ensures that every decision made can be traced back to its origin, providing accountability and clarity throughout the management process.
- **Clear metrics: severity, spread, intervention effectiveness, predator success, yield preserved, cost–benefit:** The decision structure emphasizes the use of well-defined metrics to assess all aspects of the mite outbreak. These metrics include the overall severity of the situation, how quickly and widely the mites are spreading, the effectiveness of various intervention strategies, the success rate of introduced

predators, the amount of crop yield preserved, and a thorough analysis of the costs vs. the benefits of each action. By systematically evaluating these criteria, the farmer gains a comprehensive understanding of the problem and potential solutions.

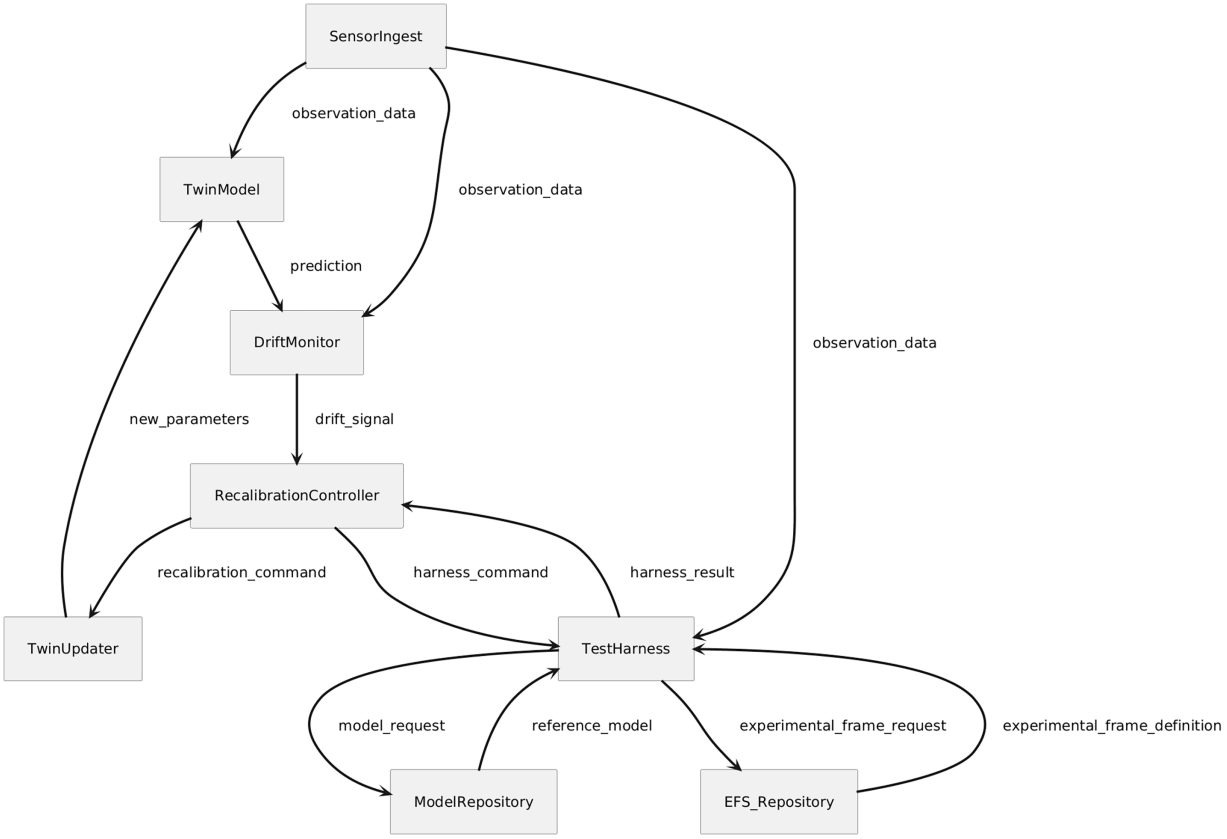

Figure 6: Digital Twin architecture expressed as DEVS coupled model the top-level Digital Twin with internal couplings implement the cyclic workflow of sensing, prediction, drift detection, recalibration, and model assurance.

In summary, by following this structured process the farmer can make decisions with confidence. Each choice is grounded in solid evidence and thorough analysis, minimizing uncertainty and maximizing the likelihood of successful outcomes.

7 Digital Twin Architecture for Seasonal Orchard Dynamics

7.1 DEVS-Structured Digital Twin Architecture

We illustrate implementation of test harnesses in the context of a DT architecture in Fig. 6 using the Discrete Event System Specification (DEVS) formalism [7]. Seasonal ecological systems rarely behave in a stationary or predictable manner. In the classic predator–prey orchard experiments [7] each season introduced new dispersal patterns, altered patch occupancies, and shifts in prey–predator persistence. These changes represented ecological drift: the system's structure and behavior evolved as environmental conditions, resource distributions, and species interactions changed. A DT intended to support agricultural decision-making must therefore remain aligned with this drifting reality. A static model, calibrated once and left unchanged, will inevitably diverge from the real orchard's dynamics, undermining its predictive value and eroding trust in its recommendations.

Agricultural decisions—such as when to introduce predators, how to manage dispersal barriers, or how to adjust resource distributions—depend on accurate forecasts of population persistence, outbreak risk, and spatial spread. These decisions are time-sensitive and season-specific. A DT that does not track seasonal variability cannot justify its predictions, because the ecological processes it encodes no longer correspond to the processes unfolding in the orchard. Maintaining correspondence between the twin and the real system is therefore not optional; it is a prerequisite for using the twin as a decision-support tool.

The architecture introduced here is designed to meet this requirement by embedding the DT within a cyclic workflow of sensing, prediction, drift detection, recalibration, and model assurance. Each season's observations enter the system through a dedicated ingestion component, allowing the twin to update its internal state and generate predictions for comparison. A drift monitor evaluates whether the twin's predictions remain within acceptable tolerance of observed behavior. When deviations exceed this threshold, the system initiates a recalibration cycle that is not automatic but supervised: a controller triggers a test harness that performs both verification and validation before any parameter updates are applied.

Verification ensures that the recalibrated twin remains consistent with the modeling assumptions and abstraction relations encoded in the model hierarchy. Validation ensures that the recalibrated twin remains faithful to the real orchard's ecological behavior. Only when both checks succeed does the system authorize the update of the twin's parameters. This closed-loop design ensures that the DT remains both internally correct and externally aligned with the real system, preserving its ability to support agricultural decisions grounded in current ecological conditions.

Fig. 6 illustrates cyclic DT workflow that emerges naturally from the Frame/Model structure: each component exchanges messages that advance the system through sensing, prediction, drift detection, recalibration, and model assurance. The architecture functions as a closed-loop supervisory system in which the twin is continuously aligned with the real orchard dynamics while maintaining internal correctness through verification and external fidelity through validation.

The cyclic Digital Twin workflow in Fig. 6 can be understood as a concise sequence of supervisory steps that repeat each season. The following high-level view is intended to help readers grasp the main idea without following every detailed interaction among components.

1. **Ingest Seasonal Observations** *SensorIngest* collects the latest orchard data—patch states, prey/predator counts, dispersal outcomes—and forwards these observations both to the *TwinModel* and the *DriftMonitor*.
2. **Predict Current-Season Dynamics** The *TwinModel* updates its internal state using the new observations and generates predicted ecological trajectories for the current season.
3. **Compare Predictions to Reality** The *DriftMonitor* receives both observed and predicted data and computes their deviation.
    - If deviation is within tolerance, the twin is considered well-aligned and the cycle continues normally.
    - If deviation exceeds tolerance, recalibration is required.
4. **Trigger Recalibration When Needed** When drift is detected, the *RecalibrationController* initiates a supervised recalibration cycle. It activates the *TestHarness* to evaluate candidate parameter updates and signals the *TwinUpdater* to prepare for possible model revision.
5. **Run Verification and Validation Tests** The *TestHarness* performs two complementary checks:
    - Verification using reference models from the *ModelRepository* to ensure internal correctness.
    - Validation using real observations (via *SensorIngest*) to ensure external ecological fidelity. EF definitions from the *EFS_Repository* ensure that all tests match the correct observational commitments.

6. **Approve or Reject Recalibration** The *TestHarness* returns a single *harness_result* to the *RecalibrationController*.
    - If the recalibrated parameters pass both verification and validation, the controller authorizes the *TwinUpdater* to apply them.
    - If not, recalibration is rejected and the system continues with existing parameters.
7. **Return to Normal Operation** Once updated, the *TwinModel* resumes its role in the sensing–prediction–monitoring loop, now recalibrated for the next season's dynamics.

This streamlined cycle ensures that Digital Twin remains continuously aligned with the real orchard system while maintaining internal correctness and external ecological validity.

Appendix A contains a sketch of a DEVS formalism description of the architecture in Fig. 6. The top-level Digital Twin is defined as a DEVS coupled model whose internal couplings implement the cyclic workflow of sensing, prediction, drift detection, recalibration, and model assurance.

7.2 Verification and Validation within the DT Lifecycle

Verification and validation play distinct but complementary roles in the recalibration cycle.

Verification ensures that the recalibrated twin remains consistent with the modeling assumptions and abstraction relations encoded in the model hierarchy. The TestHarness accomplishes this by comparing the recalibrated twin's behavior to reference models retrieved from the ModelRepository. These reference models may include the Base Model, Lumped Model, or other canonical forms that serve as correctness anchors. The harness checks whether the recalibrated twin preserves expected invariants, event sequences, or aggregated dynamics. A failure here indicates an implementation or parameterization error, not a mismatch with the real system.

Validation, by contrast, ensures that the recalibrated twin remains faithful to the real orchard dynamics. The TestHarness uses observational data from SensorIngest to evaluate whether the twin reproduces key ecological patterns—such as predator–prey oscillations, dispersal success rates, or patch occupancy distributions. A failure here indicates that the model, even if internally correct, no longer captures the real system's behavior and may require structural revision rather than simple recalibration.

Together, verification and validation ensure that recalibration does not degrade the twin's correctness or ecological fidelity. Only when both succeed does the RecalibrationController authorize the TwinUpdater to apply new parameters.

7.3 DT Architecture Summary

The DEVS coupled model forms a coherent DT architecture in which each component contributes to a continuous cycle of sensing, prediction, comparison, recalibration, and assurance. SensorIngest provides real-world data; TwinModel predicts ecological dynamics; DriftMonitor detects deviations; RecalibrationController supervises corrective action; TestHarness ensures correctness and fidelity; EFS_Repository and ModelRepository provide the formal and structural context for testing; and TwinUpdater applies approved parameter updates. The result is a robust, cyclic workflow that maintains the twin's alignment with a drifting ecological system while preserving the integrity of its internal modeling assumptions.

The resulting architecture is not merely a mechanism for recalibration; it is a principled framework for maintaining scientific and operational integrity in the face of seasonal variability. By integrating continuous sensing, formal model relations, and rigorous testing, the DT becomes a trustworthy partner in agricultural management—capable of adapting to ecological drift while providing reliable predictions for real-world decisions.

8 Related Work

Research on Experimental Frames (EFs) and their role in modeling and simulation has a long lineage, originating in Zeigler's foundational work on the Theory of Modeling and Simulation (TMS) [6,7] The EF concept matured significantly in the context of multilevel and multiformalism modeling. Zeigler's ecosystem case study [8] showed that frames can be organized to guide transitions from detailed models to simpler abstractions, establishing the practice of matching model abstraction to observational requirements in EFs. Later studies extended EF formalism for model reuse and automated reasoning. Traoré and Muzy [9,10] established EFs as primary specifications, highlighting their symbolic manipulation, verification, and compatibility assessment. They also defined the critical distinction between EF generation and acceptance modes, important for digital-twin experiments. Parallel work on model reuse and curation highlighted the importance of EF-driven approaches for discovering and composing models. Zeigler's methodology leveraged EFs to retrieve, validate, and integrate legacy models, promoting EF-based curation through model annotation of intended use, assumptions, and constraints to enable automated selection and assembly.

Recent studies [12,13] introduced a formal framework using I/O automata for EF-driven validation, allowing model checking to verify if models meet frame constraints. This method uses quantitative and qualitative metrics to assess simulation validity, showing EFs can be both experimental and formal validation tools.

In the domain of software and cyber-physical systems, EFs have been used to structure quality evaluation and virtual testing. Cao et al. [13] extended DEVSML to support EF specification for software reliability and performance evaluation, showing how EF-driven generators, acceptors, and Metrics Computers can automate early-phase quality assessment. Their work illustrates the growing relevance of EFs in digital engineering workflows where simulation-based evaluation must be repeatable, traceable, and aligned with stakeholder concerns. The EFSUT methodology under development [14] brings the EF concept into modern digital-engineering settings, making sure each model run is linked to stakeholders' goals and that the level of model detail matches the observational requirements defined by the frame.

Together, these contributions establish a functional foundation for the present work. They demonstrate that EFs are not merely simulation artifacts but central organizing structures for model development, reuse, validation, and decision support. The EFSUT methodology extends this tradition by unifying these roles into a single lifecycle-spanning framework suitable for digital engineering and digital-twin ecosystems.

9 Contributions

Table 2 compares EFSUT with several frameworks and methodologies commonly used for model verification and testing. Each column summarizes how the respective methodology addresses core aspects of simulation and model evaluation, including internal correctness, stakeholder relevance, model selection, and test organization.

In brief, this comparison clarifies the unique contributions of EFSUT, particularly its emphasis on aligning verification and testing processes with stakeholder questions and intent.

Table 2: Comparison of EFSUT with several common frameworks and methodologies.

Framework	Traditional Approach	EFSUT Distinction
DEVS-Based Verification [15]	Ensures internal model correctness (closure, determinism, conformance). Focuses on structural validity, but does not address suitability for stakeholder questions or intent.	Starts with stakeholder questions, refines them into executable Experimental Frames (Derivability), checks when frames apply to models (Applicability), and relates models across abstraction levels (Morphism), connecting correctness to test relevance.
SES-Driven Model Families [16]	Organizes model families via decomposition, specialization, and configuration. Generates variants but lacks guidance on evaluation or configuration relevance for stakeholder concerns.	Uses Derivability to create Frames for specific concerns, Applicability to match SES-generated models to relevant Frames, and Morphism to justify substitution. Structures both models and tests for coherent selection and evaluation.
Model-Based Testing (MBT) [17]	Generates systematic test cases from a behavioral model, assuming fixed test specs, a single reference model, and direct mapping. Does not derive tests from stakeholder concerns or relate multiple models.	Defines test intent through Questions, derives test specs (Frames), checks test applicability, and relates models (Morphism). Acts as a meta-framework, organizing and justifying MBT in a broader context.
Formal Verification & Model Checking [18]	Provides exhaustive guarantees via temporal logic, requiring precise specs and finite models. Limited to systems suitable for symbolic exploration.	Derives testable Frames from stakeholder Questions without formal logic. Supports simulation-based evaluation for systems with continuous, stochastic, or unbounded dynamics, complementing formal methods with principled testing beyond model checking.

10 Conclusions and Further Development

The EFSUT methodology offers a unifying structure for digital engineering and DT development by explicitly linking stakeholder questions, model abstraction levels, system implementations, and decision-oriented analytics. Across the digital-engineering lifecycle—design, development, verification, operation, and sustainment—EFSUT provides a disciplined way to specify observational regimes, derive appropriate models, and ensure that every simulation or test is grounded in clearly articulated experimental conditions. This explicitness is essential for building trustworthy digital twins whose behavior can be traced back to well-defined assumptions and validated against consistent criteria.

As a model-development discipline, EFSUT enables the construction of coherent multiresolution model families aligned with a derivability-ordered partial order of experimental frames. This ensures that digital-twin components are not isolated artifacts but members of a structured, interoperable architecture. Such

model families support multilevel fidelity, multiformalism integration, and lifecycle reuse—capabilities increasingly required in aerospace, defense, manufacturing, and other engineering domains.

As a test-engineering framework, EFSUT provides a principled basis for generating test harnesses, admissible input sequences, and validation scenarios directly from experimental frames. This strengthens digital-thread verification by ensuring that system implementations are evaluated under the same observational regimes that motivated their design. The result is improved traceability, reproducibility, and confidence in both virtual testing and physical system verification.

As a decision-support methodology, EFSUT integrates naturally with simulation to support branching simulations, uncertainty quantification, and evidence-based reasoning. By grounding decision metrics in explicit experimental frames, the methodology ensures that digital-twin outputs remain interpretable, auditable, and aligned with stakeholder needs. This is particularly important as digital twins increasingly serve operational roles in monitoring, prediction, and real-time decision-making.

Taken together, these three uses demonstrate that EFSUT is not merely a modeling construct but a lifecycle-spanning methodological scaffold for digital engineering. It provides the conceptual clarity and structural discipline needed to integrate multiresolution models, virtual testing workflows, and decision-oriented analytics into a coherent digital-twin ecosystem.

Continued research is essential to advance this foundation. Promising directions include the automated derivation of Experimental Frames from requirements artifacts, integration with MBSE toolchains, formal verification of frame–model applicability, and scalable simulation-based decision pipelines [19–21].

As systems increase in complexity and digital engineering becomes more central to lifecycle management [22–26], the demand for principled, traceable, and minimal modeling methodologies will grow. EFSUT provides a rigorous foundation to meet this need.

For example, ongoing work seeks to extend the X language with first-class constructs for observation commitments, experimental frames, and applicability relations, embedding them directly into the modeling syntax [27–29]. These enhancements aim to unify requirements, architecture, and DEVS-based simulation semantics, enabling automated traceability from stakeholder intent to executable models and ensuring minimal sufficient model selection during verification.

Advancing these capabilities will strengthen the foundations of digital-twin practice and help realize the vision of transparent, trustworthy, and lifecycle-consistent digital engineering.

Acknowledgement: Not applicable.

Funding Statement: The author received no specific funding for this study.

Availability of Data and Materials: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Ethics Approval: Not applicable.

Conflicts of Interest: Bernard P. Zeigler is an employee at a company that develops software mentioned in the paper.

Appendix A Sketch of DEVS Specification of the DT Architecture

As indicated above, a DEVS-formalism sketch of the Digital Twin is defined as a DEVS coupled model whose internal couplings implement the cyclic workflow of sensing, prediction, drift detection, recalibration, and model assurance.

Each subsystem in the Digital Twin architecture is represented as a component DEVS model.

Appendix A.1 Coupled Model Specification

The Digital Twin is defined as the coupled DEVS model:

Components:

1. **SensorIngest**
2. **TwinModel**
3. **DriftMonitor**
4. **RecalibrationController**
5. **TestHarness**
6. **ModelRepository**
7. **EFS_Repository**
8. **TwinUpdater**

The message types correspond to the labels in the coupling diagram: data, pred, drift, harness_cmd, recalib_cmd, ef_req, ef_def, model_req, ref_model, obs_data, harness_result, new_params.

1. **SensorIngest**

Represents seasonal data ingestion from the real orchard.

- **Inputs:** none (environmentally driven).
- **Outputs:** data (observed patch states).
- **State:** most recent observation record.
- **External transition:** triggered by arrival of new seasonal data.
- **Output:** emits data to TwinModel, DriftMonitor, and TestHarness.
- **Time advance:** zero when new data arrives; ∞ otherwise.

2. **TwinModel**

Represents the current Digital Twin ecological model (Lumped, Occupancy, or Base).

- **Inputs:** data (observations), new_params (updated parameters).
- **Outputs:** pred (predicted ecological state).
- **State:** model parameters, internal ecological state.
- **External transition:**
  - On data: update internal state.
  - On new_params: replace parameter set.

Internal transition: advance simulation state.
Output: emit pred at scheduled events.
Time advance: model-dependent.

3. **DriftMonitor**

Computes deviation between observed and predicted behavior.

- **Inputs:** data, pred.
- **Outputs:** drift.
- **State:** last observed state, last predicted state, deviation metric.
- **External transition:** update stored values and recompute deviation.
- **Output:** emit drift when deviation changes.
- **Time advance:** zero when both inputs available; ∞ otherwise.

4. **RecalibrationController**

Supervises recalibration and test harness activation.

- **Inputs:** drift, harness_result.
- **Outputs:** harness_cmd, recalib_cmd.
- **State:** thresholds, pending recalibration flag, test status.
- **External transition:**
  - On drift: if deviation > tolerance, issue recalibration commands.
  - On harness_result: accept or reject recalibration.
- **Output:**
  - harness_cmd to TestHarness.
  - recalib_cmd to TwinUpdater.
- **Time advance:** zero when decisions required; ∞ otherwise.

5. **TestHarness**

Executes verification and validation tests.

- **Inputs:** harness_cmd, ef_def, ref_model, obs_data.
- **Outputs:** ef_req, model_req, harness_result.
- **State:** test configuration, EF definition, reference model, test outcomes.
- **External transition:**
  - On harness_cmd: begin test cycle.
  - On ef_def and ref_model: configure tests.
  - On obs_data: perform validation.
- **Output:** harness_result summarizing pass/fail.
- **Time advance:** zero during test execution; ∞ otherwise.

6. **ModelRepository**

Provides reference models for verification.

- **Inputs:** model_req.
- **Outputs:** ref_model.
- **State:** stored model library.
- **External transition:** respond to requests.
- **Output:** emit requested reference model.
- **Time advance:** zero.

7. **EFS_Repository**

Provides EF definitions for test harness configuration.

- **Inputs:** ef_req.
- **Outputs:** ef_def.
- **State:** stored EF definitions.
- **External transition:** respond to EF requests.
- **Output:** emit EF definition.
- **Time advance:** zero.

8. **TwinUpdater**

Applies recalibrated parameters to the TwinModel.

- **Inputs:** recalib_cmd.
- **Outputs:** new_params.
- **State:** parameter estimation logic.
- **External transition:** compute new parameters when commanded.
- **Output:** emit new_params.
- **Time advance:** zero when recalibration triggered; ∞ otherwise.

External Input Couplings (EIC)

None (environmental data enters SensorIngest directly).

External Output Couplings (EOC)

None (outputs are internal to the DT lifecycle).

Internal Couplings (IC)

Using the simplified notation "A –msg→ B" to denote a coupling carrying message *msg*:

- SensorIngest – obs_data→ TwinModel
- SensorIngest – obs_data→ DriftMonitor
- SensorIngest –obs_data→ TestHarness
- TwinModel –pred→ DriftMonitor
- DriftMonitor –drift→ RecalibrationController
- RecalibrationController –harness_cmd→ TestHarness
- RecalibrationController –recalib_cmd→ TwinUpdater
- TestHarness –ef_req→ EFS_Repository
- EFS_Repository –ef_def→ TestHarness
- TestHarness –model_req→ ModelRepository
- ModelRepository –ref_model→ TestHarness
- TestHarness –harness_result→ RecalibrationController
- TwinUpdater –new_params→ TwinModel

These couplings implement the full cyclic workflow: sensing → prediction → drift detection → recalibration → verification/validation → parameter update → sensing.

Appendix A.2 Interpretation

This DEVS specification formalizes the Digital Twin as a supervisory control loop in which:

- **SensorIngest** anchors the twin to real seasonal data.
- **TwinModel** produces predictions under current parameters.
- **DriftMonitor** detects divergence.
- **RecalibrationController** governs when recalibration is allowed.
- **TestHarness** ensures correctness (verification) and ecological fidelity (validation).
- **TwinUpdater** applies new parameters only when justified.
- **ModelRepository** and **EFS_Repository** provide the formal context for testing.

The resulting coupled model is a fully DEVS-compliant Digital Twin architecture suitable for simulation execution and lifecycle-consistent digital engineering.

References

1. Blackburn MR, West TD. Fundamentals of digital engineering. Hoboken, NJ, USA: John Wiley & Sons, Inc.; 2023.
2. U.S. Department of Defense. Digital engineering strategy. Washington, DC, USA: DoD; 2018.
3. Bersbach D, West TD, Blackburn MR et al. Digital twins need digital threads for success. Huntsville, AL, USA: Digital Engineering Working Group; 2022.
4. Saporiti N, Cannas VG, Pozzi R, Rossi T. Challenges and countermeasures for digital twin implementation in manufacturing plants: a Delphi study. Int J Prod Econ. 2023;261(2):108888. doi:10.1016/j.ijpe.2023.108888.
5. Samak TV, Samak CV, Brault J, Harber C, McCane K, Smereka J, et al. A systematic digital engineering approach to verification and validation of autonomous ground vehicles in off-road environments. In: Proceedings of IEEE SysCon 2020; 2020 Aug 24–Sep 20; Montreal, QC, Canada. Piscataway, NJ, USA: IEEE; 2020. p. 1–8.
6. Zhang L, Zhao C, editors. Modeling and simulation based systems engineering. Singapore: World Scientific; 2023.
7. Zeigler BP, Muzy A, Kofman E. Theory of modeling and simulation. 3rd ed. London, UK: Academic Press; 2018.
8. Wach P, Salado A. The need for semantic extension of SysML to model the problem space. In: Recent trends and advances in model based systems engineering. Cham, Switzerland: Springer International Publishing; 2022. p. 279–89.
9. Zeigler BP. Multilevel multiformalism modeling: an ecosystem example. In: Theoretical systems ecology. New York, NY, USA: Academic Press; 1979. p. 17–54.
10. Traoré MK, Muzy A. Capturing the dual relationship between simulation models and their context. Simul Model Pract Theory. 2006;14(2):126–42. doi:10.1016/j.simpat.2005.03.002.
11. Wach P, Beling P, Salado A. Formalizing the representativeness of verification models using morphisms. Insight. 2023;26(1):27–32. doi:10.1002/inst.12426.
12. Zeigler BP. A methodology to characterise simulation models for discovery and composition: a system theory-based approach to model curation for integration and reuse. Int J Simul Process Model. 2022;19(1–2):3–13. doi:10.1504/ijspm.2022.130277.
13. Cao B, Huang L, Hu J. Experimental frame design using E-DEVSML for software quality evaluation. In: Proceedings of SEKE 2015; 2015 Jul 6–8; Pittsburgh, PA, USA.
14. Risco-Martín JL, Mittal S, Henares K, Cardenas R, Arroba P. xDEVS: a toolkit for interoperable modeling and simulation of formal discrete event systems. Softw Pract Exp. 2023;53(3):748–89. doi:10.1002/spe.3168.
15. Samuel KG, Bouare NM, Maïga O, Traoré MK. A DEVS-based pivotal modeling formalism and its verification and validation framework. Simulation. 2020;96(12):969–92. doi:10.1177/0037549720958056.
16. Folkerts H. An architecture for model behavior generation for multiple simulators [dissertation]. Wismar, Germany: University of Applied Sciences Wismar; 2024.
17. Traore MK, Gorecki S, Ducq Y. A formal framework for digital twin modeling, verification, and validation. In: Digital twins, simulation, and the metaverse. Cham, Switzerland: Springer Nature Switzerland; 2024. p. 119–43.
18. Clarke EM Jr, Grumberg O, Kroening D, Peled D, Veith H. Model checking. Cambridge, MA, USA: MIT Press; 2018.
19. Ansari AM, Nazir M, Mustafa K. Ontology-based classification and detection of the smart home automation rules conflicts. IEEE Access. 2024;12(4):85072–88. doi:10.1109/ACCESS.2024.3415632.
20. Capocchi L, Santucci JF, Tigli JY, Gomnin T, Lavirotte S, Rocher G. Actuation conflict management in Internet of Things systems DevOps: a discrete event modeling and simulation approach. In: Proceedings of the 7th IFIP WG 5.5 International Cross-Domain Conference on Internet of Things, IFIPIoT 2024; 2024 Nov 6–8. Nice, France. Cham, Switzerland: Springer Nature Switzerland; 2024. p. 189–206.
21. Van Tendeloo Y, Vangheluwe H. An evaluation of DEVS simulation tools. Simulation. 2017;93(2):103–21. doi:10.1177/0037549716678330.
22. Risco-Martin JL, Mittal S, Fabero JC, Malagon P, Ayala JL. Real-time hardware/software co-design using DEVS-based transparent M&S framework. In: Proceedings of the Summer Computer Simulation Conference (SCSC '16); 2016 Jul 24–27; Montreal, QC, Canada.
23. Wymore AW. A mathematical theory of systems engineering: the elements. Huntington, NY, USA: Krieger; 1967.

24. Reggio G. A UML-based proposal for IoT system requirements specification. In: Proceedings of the 10th International Workshop on Modelling in Software Engineering; 2018 May 27–28; Gothenburg, Sweden. p. 9–16.
25. Xiao R, Wu Z, Wang D. A finite-state-machine model driven service composition architecture for Internet of Things rapid prototyping. Future Gener Comput Syst. 2019;99(3):473–88. doi:10.1016/j.future.2019.04.050.
26. da Silva Fonseca JP, de Sousa AR, de Souza Tavares JJZ. Modeling and controlling IoT-based devices' behavior with high-level Petri nets. Procedia Comput Sci. 2023;217(1):1462–9. doi:10.1016/j.procs.2022.12.345.
27. Escamilla-Ambrosio PJ, Robles-Ramírez DA, Tryfonas T, Rodríguez-Mota A, Gallegos-García G, Salinas-Rosales M. IoTsecM: a UML/SysML extension for Internet of Things security modeling. IEEE Access. 2021;9:154112–35. doi:10.1109/ACCESS.2021.3125979.
28. Chreyh R, Wainer GA. CD++ repository: an Internet-based searchable database of DEVS models and their experimental frames. In: Proceedings of the 2009 Spring Simulation Multiconference; 2009 Mar 22–27; San Diego, CA, USA.
29. Xie K, Zhang L, Wang X, Wang K, Li Y. Integrated modelling and simulation method of hybrid systems based on X language. IET Collab Intell Manuf. 2024;6(4):e70006. doi:10.1049/cim2.70006.