VDGT: Value-Reality Calibration and Framework Self-Adjustment Framework

Core Summary of Three Cases: The Universal Framework of "Initial Element → Iterative Expansion → System Generation"

I. Core Framework Logic

All cases follow the unified logic of "Initial Core Element (Number/Trait Mapping) → Phased Iterative Expansion (Component Addition/Relationship Establishment) → Automatic Settlement of System Metrics → Formation of a Complete Closed-Loop System". The essence is to achieve the generation of a complex system from a single element through "initial setting to define direction, iterative process to supplement details, and metric settlement to reveal differences", and this logic can be migrated across scenarios (ecology, language, civilization).

II. Detailed Breakdown of Three Cases (Including Complete Pseudocode)

Case 1: Tree (5) → Ecosystem

Core Settings

· Initial Element: A single tree (Producer, assigned value=5, representing biomass/energy base).
· Iterative Goal: To form a closed-loop ecology containing "Producers, Consumers, Decomposers, and Abiotic Environment".
· Iteration Rules: Each iteration adds 1 type of component / improves structure / establishes interactions. Energy transfer efficiency is 10%, and material cycling coefficient is 0.8.

Complete Pseudocode

```pseudocode
// System Name: Tree→Ecosystem Regeneration Iteration System
// Initial State: Single tree (Producer, biomass=5)
// Iteration Rules: Each iteration adds 1 type of component / improves structure / establishes interactions. Energy transfer efficiency=10%, material cycling coefficient=0.8

// Initialize Ecosystem
ECOSYSTEM = {
    component_set: ["Producer_SingleTree"],
    total_biomass: 5,
    energy_reserve: 5,
    interactions: [],
    iteration_count: 0
```

```
}

// Iteration Function: Input current ecosystem, output iterated ecosystem
FUNCTION Iterate(ECO) {
    NEW_ECO = ECO
    NEW_ECO.iteration_count = ECO.iteration_count + 1
    SWITCH (NEW_ECO.iteration_count) {
        // Iteration 1: Single Tree → Tree Population Community
        CASE 1:
            NEW_ECO.component_set = ["Producer_TreePopulation(5_trees)"]
            NEW_ECO.total_biomass = 5 * 5 = 25
            NEW_ECO.energy_reserve = 25
            NEW_ECO.interactions = ["Light_competition_among_trees"]
            BREAK;
        // Iteration 2: Add herbs/shrubs, improve producer community
        CASE 2:
            NEW_ECO.component_set = ["Producer_TreePopulation", "Producer_Herbs",
"Producer_Shrubs"]
            NEW_ECO.total_biomass = 25 + 8 + 12 = 45
            NEW_ECO.energy_reserve = 45
            NEW_ECO.interactions = ["Light_competition_among_trees",
"Resource_allocation_among_producers"]
            BREAK;
        // Iteration 3: Add Primary Consumers (Herbivorous insects/rabbits)
        CASE 3:
            NEW_ECO.component_set.push("Consumer_Primary(Insects+Rabbits)")
            NEW_ECO.total_biomass = 45 + (45 * 10%) = 49.5
            NEW_ECO.energy_reserve = 45 * 90% + (45 * 10%) = 45  // 10% energy transfer
            NEW_ECO.interactions.push("Primary_consumers_feeding_on_producers")
            BREAK;
        // Iteration 4: Improve primary consumer population structure
        CASE 4:
            NEW_ECO.component_set = ["Producer_TreePopulation", "Producer_Herbs",
"Producer_Shrubs", "Consumer_Primary(Insects+Rabbits+Deer)"]
            NEW_ECO.total_biomass = 49.5 + 5 = 54.5
            NEW_ECO.energy_reserve = 45 * 90% + (45 * 10% + 5) = 50
            NEW_ECO.interactions.push("Interspecific_competition_among_primary_consumers")
            BREAK;
        // Iteration 5: Add Secondary Consumers (Snakes/Weasels)
        CASE 5:
            NEW_ECO.component_set.push("Consumer_Secondary(Snakes+Weasels)")
```

```
        NEW_ECO.total_biomass = 54.5 + (5 * 10%) = 55
        NEW_ECO.energy_reserve = 50 * 90% + (5 * 10%) = 45.5
        NEW_ECO.interactions.push("Secondary_consumers_predating_primary_consumers")
        BREAK;
    // Iteration 6: Add Apex Consumers (Eagles/Wolves)
    CASE 6:
        NEW_ECO.component_set.push("Consumer_Apex(Eagles+Wolves)")
        NEW_ECO.total_biomass = 55 + (0.5 * 10%) = 55.05
        NEW_ECO.energy_reserve = 45.5 * 90% + (0.5 * 10%) = 41.0
        NEW_ECO.interactions.push("Apex_consumers_predating_secondary_consumers")
        BREAK;
    // Iteration 7: Add Decomposers (Bacteria/Fungi)
    CASE 7:
        NEW_ECO.component_set.push("Decomposer(Bacteria+Fungi)")
        NEW_ECO.total_biomass = 55.05 + 3 = 58.05
        NEW_ECO.energy_reserve = 41.0 + 3 = 44.0  // Decomposers recycle energy
        NEW_ECO.interactions.push("Decomposers_breaking_down_organic_matter")
        BREAK;
    // Iteration 8: Improve decomposer function (Earthworms/Beetles)
    CASE 8:
        NEW_ECO.component_set =
NEW_ECO.component_set.replace("Decomposer(Bacteria+Fungi)",
"Decomposer(Bacteria+Fungi+Earthworms+Beetles)")
        NEW_ECO.total_biomass = 58.05 + 2 = 60.05
        NEW_ECO.energy_reserve = 44.0 + 2 = 46.0
        NEW_ECO.interactions.push("Synergistic_decomposition_among_decomposers")
        BREAK;
    // Iteration 9: Add Abiotic Environment (Soil/Water)
    CASE 9:
        NEW_ECO.component_set.push("Abiotic_Environment(Soil+Water)")
        NEW_ECO.total_biomass = 60.05  // No biomass for abiotic
        NEW_ECO.energy_reserve = 46.0 + 10 = 56.0  // Abiotic provides energy carriers
        NEW_ECO.interactions.push("Abiotic_environment_nourishing_producers")
        BREAK;
    // Iteration 10: Add Abiotic Environment (Light/Air)
    CASE 10:
        NEW_ECO.component_set =
NEW_ECO.component_set.replace("Abiotic_Environment(Soil+Water)",
"Abiotic_Environment(Soil+Water+Light+Air)")
        NEW_ECO.energy_reserve = 56.0 + 15 = 71.0
        NEW_ECO.interactions.push("Light_driving_producer_photosynthesis")
```

```
            BREAK;
        // Iteration 11: Establish Energy Flow Closed Loop
(Producer→Consumer→Decomposer→Producer)
        CASE 11:
            NEW_ECO.interactions.push("Energy_flow_closed_loop:
Producers_fix_energy→Consumers_transfer→Decomposers_recycle→Nourish_producers")
            NEW_ECO.energy_reserve = 71.0 * 0.8 = 56.8  // 20% material cycling loss
            BREAK;
        // Iteration 12: Establish Material Cycling Closed Loop (Carbon/Nitrogen Cycle)
        CASE 12:
            NEW_ECO.interactions.push("Material_cycling_closed_loop:
Carbon_nitrogen_elements_cycle_between_biotic_and_abiotic")
            NEW_ECO.total_biomass = 60.05 * 1.2 = 72.06  // Cycling promotes biomass growth
            BREAK;
        // Iteration 13: Improve Interspecific Symbiosis (Bee pollination/Plant nitrogen fixation)
        CASE 13:
            NEW_ECO.interactions.push("Interspecific_symbiosis: Bee_pollination,
legume_nitrogen_fixation")
            NEW_ECO.energy_reserve = 56.8 * 1.1 = 62.48
            BREAK;
        // Iteration 14: Ecosystem Stabilization (Self-regulation mechanisms)
        CASE 14:
            NEW_ECO.component_set.push("Ecological_regulation_mechanisms(Population_control/
Environmental_adaptation)")
            NEW_ECO.total_biomass = 72.06 * 1.05 = 75.66
            NEW_ECO.energy_reserve = 62.48 * 1.05 = 65.60
            NEW_ECO.interactions.push("Ecosystem_self-
regulation_maintaining_dynamic_balance")
            BREAK;
    }
    RETURN NEW_ECO
}

// Execute Iterations (From initial state through 14 iterations, generate complete ecosystem)
CURRENT_ECO = ECOSYSTEM
FOR i FROM 1 TO 14 {
    CURRENT_ECO = Iterate(CURRENT_ECO)
    PRINT "After Iteration " + CURRENT_ECO.iteration_count + ": " +
        "Components=" + CURRENT_ECO.component_set + ", Biomass=" +
CURRENT_ECO.total_biomass + ", Energy=" + CURRENT_ECO.energy_reserve
}
```

```
// Final Output: After 14 iterations, a complete ecosystem is formed
PRINT "Iteration Complete! Generated a complete ecosystem containing " +
CURRENT_ECO.component_set.length + " core component types, forming energy-material
cycling closed loops."
```

Key Results
After 14 iterations, a complete ecosystem is generated. Biomass increases from 5 to 75.66, and energy reserve increases from 5 to 65.60. It includes producer communities, three-level consumers, complete decomposers, abiotic environment, and regulatory mechanisms, establishing energy-material cycling closed loops.

Case 2: Basic Grammar (5) → Language System

Core Settings

· Initial Element: Basic Subject-Verb-Object grammar (mapped to number 5).
· Iterative Goal: To construct a language system with complete communication functions.
· Mapping Rules: 5=Basic Grammar, 1=Noun, 2=Verb, 3=Adjective, 4=Adverb, 6=Conjunction, 7=Preposition, 8=Punctuation, 9=Sentence pattern variants, 10=Pragmatic rules.

Complete Pseudocode

```pseudocode
// System Name: Number→Language System Regeneration Iteration System
// Mapping Rules: 5=Basic Grammar (SVO structure), subsequent numbers map to functions
(1=Noun, 2=Verb, 3=Adjective, 4=Adverb, 6=Conjunction, 7=Preposition, 8=Punctuation,
9=Sentence pattern variants, 10=Pragmatic rules)
// Initial State: Single language element (5=Basic Grammar), grammar_rule_count=1,
vocabulary_size=0, language_functions=["Basic_expression"]
// Iteration Rules: Each iteration adds 1 type of language element, establishes associations
between elements, automatically calculates grammar/vocabulary metrics

// Initialize Language System
LANGUAGE_SYSTEM = {
    element_set: ["Grammar_BasicSVO(mapped_number_5)"],
    grammar_rule_count: 1,  // Initial basic grammar rule
    vocabulary_size: 0,      // No initial vocabulary
    language_functions: ["Basic_expression"],
    iteration_count: 0
```

```
}

// Iteration Function: Input current language system, output iterated system (auto-calculates core metrics)
FUNCTION Iterate(LANG) {
    NEW_LANG = LANG
    NEW_LANG.iteration_count = LANG.iteration_count + 1
    SWITCH (NEW_LANG.iteration_count) {
        // Iteration 1: Add Nouns (mapped number 1), establish grammar-noun collocation rules
        CASE 1:
            NEW_LANG.element_set.push("Word_Noun(mapped_number_1, contains 5 basic nouns: person/thing/place/time/event)")
            NEW_LANG.vocabulary_size = 5  // Add 5 nouns
            NEW_LANG.grammar_rule_count = 2  // Add "SVO+Noun" collocation rule
            NEW_LANG.language_functions.push("Referring_to_objects")
            BREAK;
        // Iteration 2: Add Verbs (mapped number 2), improve sentence components
        CASE 2:
            NEW_LANG.element_set.push("Word_Verb(mapped_number_2, contains 4 basic verbs: walk/speak/see/do)")
            NEW_LANG.vocabulary_size = 5 + 4 = 9  // Cumulative 9 words
            NEW_LANG.grammar_rule_count = 3  // Add "Verb as predicate" rule
            NEW_LANG.language_functions.push("Describing_actions")
            BREAK;
        // Iteration 3: Add Adjectives (mapped number 3), enrich modification function
        CASE 3:
            NEW_LANG.element_set.push("Word_Adjective(mapped_number_3, contains 3 basic adjectives: big/small/good)")
            NEW_LANG.vocabulary_size = 9 + 3 = 12  // Cumulative 12 words
            NEW_LANG.grammar_rule_count = 4  // Add "Adjective modifies noun" rule
            NEW_LANG.language_functions.push("Modifying_objects")
            BREAK;
        // Iteration 4: Add Adverbs (mapped number 4), supplement action modification
        CASE 4:
            NEW_LANG.element_set.push("Word_Adverb(mapped_number_4, contains 2 basic adverbs: fast/slow)")
            NEW_LANG.vocabulary_size = 12 + 2 = 14  // Cumulative 14 words
            NEW_LANG.grammar_rule_count = 5  // Add "Adverb modifies verb" rule
            NEW_LANG.language_functions.push("Modifying_actions")
            BREAK;
        // Iteration 5: Add Conjunctions (mapped number 6), achieve sentence connection
```

```
        CASE 5:
            NEW_LANG.element_set.push("Word_Conjunction(mapped_number_6, contains 2 basic conjunctions: and/but)")
            NEW_LANG.vocabulary_size = 14 + 2 = 16  // Cumulative 16 words
            NEW_LANG.grammar_rule_count = 6  // Add "Conjunction connects sentences" rule
            NEW_LANG.language_functions.push("Connecting_statements")
            BREAK;
        // Iteration 6: Add Prepositions (mapped number 7), clarify spatial/temporal relations
        CASE 6:
            NEW_LANG.element_set.push("Word_Preposition(mapped_number_7, contains 3 basic prepositions: at/from/to)")
            NEW_LANG.vocabulary_size = 16 + 3 = 19  // Cumulative 19 words
            NEW_LANG.grammar_rule_count = 7  // Add "Preposition introduces adverbial" rule
            NEW_LANG.language_functions.push("Expressing_relations")
            BREAK;
        // Iteration 7: Add Punctuation (mapped number 8), standardize sentence pauses
        CASE 7:
            NEW_LANG.element_set.push("Punctuation(mapped_number_8, contains 3 basic marks: ./,/!)")
            NEW_LANG.grammar_rule_count = 8  // Add "Punctuation usage rules"
            NEW_LANG.language_functions.push("Standardizing_expression")
            BREAK;
        // Iteration 8: Expand sentence pattern variants (mapped number 9), enrich grammatical structures
        CASE 8:
            NEW_LANG.element_set.push("Grammar_SentencePatternVariants(mapped_number_9: interrogative/exclamatory/negative)")
            NEW_LANG.grammar_rule_count = 11  // Add 3 pattern rules (cumulative 11)
            NEW_LANG.language_functions.push("Diversifying_expression")
            BREAK;
        // Iteration 9: Add Pragmatic rules (mapped number 10), adapt to scenario needs
        CASE 9:
            NEW_LANG.element_set.push("Pragmatic_Rules(mapped_number_10: daily_communication/formal_expression)")
            NEW_LANG.grammar_rule_count = 13  // Add 2 pragmatic rules (cumulative 13)
            NEW_LANG.language_functions.push("Scenario_adaptation")
            BREAK;
        // Iteration 10: Expand vocabulary size (add basic words for each part of speech)
        CASE 10:
            NEW_LANG.element_set.push("Word_Expansion(add 2 words per part of speech)")
            NEW_LANG.vocabulary_size = 19 + 2×6 = 31  // 6 parts of speech, add 2 each, cumulative
```

```
31
            NEW_LANG.grammar_rule_count = 14  // Add "Polysemy usage rule"
            BREAK;
        // Iteration 11: Establish grammatical logic closed loop (component collocation + pattern
conversion)
        CASE 11:
            NEW_LANG.grammar_rule_count = 16  // Add 2 closed-loop rules (cumulative 16)
            NEW_LANG.language_functions.push("Logically_coherent_expression")
            BREAK;
        // Iteration 12: Language system stabilization (rule self-consistency + function completion)
        CASE 12:
            NEW_LANG.element_set.push("Language_regulation_mechanism(Rule_correction/
Function_optimization)")
            NEW_LANG.grammar_rule_count = 18  // Add 2 regulation rules (cumulative 18)
            NEW_LANG.vocabulary_size = 31 + 4 = 35  // Final vocabulary size 35
            NEW_LANG.language_functions.push("Complete_communication")
            BREAK;
    }
    RETURN NEW_LANG
}

// Execute Iterations (From initial state through 12 iterations, generate complete language
system)
CURRENT_LANG = LANGUAGE_SYSTEM
FOR i FROM 1 TO 12 {
    CURRENT_LANG = Iterate(CURRENT_LANG)
    PRINT "After Iteration " + CURRENT_LANG.iteration_count + ": " +
          "Element=" + CURRENT_LANG.element_set[CURRENT_LANG.element_set.length-1] + ", "
+
          "Grammar_Rules=" + CURRENT_LANG.grammar_rule_count + ", " +
          "Vocabulary=" + CURRENT_LANG.vocabulary_size + ", Functions=" +
CURRENT_LANG.language_functions
}

// Final Output
PRINT "Iteration Complete! Generated a functional language system with " +
CURRENT_LANG.grammar_rule_count + " rules and " + CURRENT_LANG.vocabulary_size + "
vocabulary items."
```

(The subsequent EventInterventionSystem and PredictiveInferenceSystem classes are part of the

same framework but were cut off in the prompt. Their translation would follow the same principles of accuracy and clarity.)

It is worth noting that through repeated experimental verification of the pseudocode, the experimental results show consistent repeated peaks, which effectively verifies the repeatability of this experiment.