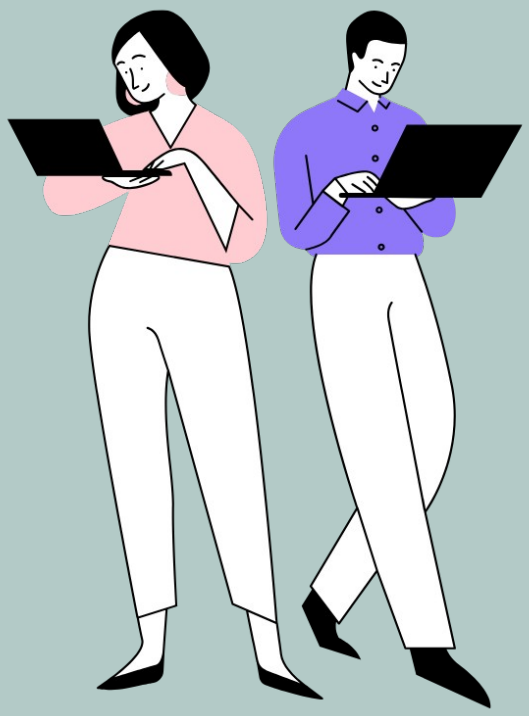
UK

CONCEPT OF PROOF
1P=1E=1C
3p3 SYSTEM

24 Agosto 2025

Luca Meggiolaro



KOOLTOOL

Document Information:

Document's name	20250824 concept_proof_3p3 v02 - UK
Start Date	24/08/25
End Date	
version	02
Responsible	Luca Meggiolaro
Language	English
Recipients	Softwarehouse

Indice generale

TECHNICAL DOCUMENT FOR 3P3 PROTOTYPE DEVELOPMENT

Project: THE BRIDGE - Proof of Concept 3P3 ENTITY System

Date: August 24, 2025

Version: 2.0 - Ontologically Corrected Document

Recipient: Cyril Amegah and Osbert Yao Vulor- Caufero Technologies

Client: KOOL TOOL SRL - Luca Meggiolaro

PREFACE - THE HISTORICAL MOMENT

This document represents the **crystallization of 30 years of ontological research** applied to information systems. From the 1996 revelation in Romania ("The Bill of Materials and the Work Cycle are the same thing") to the 2025 collaboration with Caufero Technologies, the possibility finally emerges to concretely demonstrate that the 3P3 tripartite ontology can revolutionize business management.

The objective: To experimentally validate that **1 PROCESS = 1 ENTITY = 1 COMPOSITION** through the implementation of a 3 universal tables system that definitively overcomes traditional fragmented architecture.

CHAPTER 1: THE FUNDAMENTAL ONTOLOGICAL EQUATION

1.1 The Meaning of 3P3: PEOPLE, PROCESSES, PRODUCTS

3P3 represents the universal trinity of creative action:

PEOPLE (Physical World) ↔↔ STR (Structure/Aspect)
PROCESSES (World of Ideas) ↔↔ REL (Relation/Nature)
PRODUCTS (Virtual World) ↔↔ ETY (Entity/Essence)

Fundamental Principle: "PEOPLE working through PROCESSES create PRODUCTS"

This is not a simple business model, but the **digitization of the essence of human work itself** maintaining the indissoluble ontological unity between:

- **Who does it** (operators, human resources)
- **How it's done** (methods, procedures, algorithms)
- **What it produces** (results, output, value)

1.2 The 1996 Revelation: "BOM= WORK CYCLE"

Romania, 1996 - The revolutionary intuition that will forever change information systems:

"THE BILL OF MATERIALS AND THE WORK CYCLE ARE THE SAME THING"

Traditional systems see:

- BOM (Bill of Materials): Static list of components
- WC (Work Cycle): Separate sequence of operations

The 3P3 system reveals:

- UNIFIED ENTITY: Every element simultaneously contains its material essence AND its manifestation process
- ONTOLOGICAL UNITY: There is no separation between being and becoming

1.3 Formula 1P=1E=1C: The Equation of Reality

```
1 PROCESS = 1 ENTITY = 1 COMPOSITION
     ↓           ↓           ↓
  ACTION      RESULT      TRACE
```

Operational Meaning:

- Every action in the system **simultaneously** generates three aspects of the same reality
- There are no processes without entities, nor entities without composition
- The system **always** maintains the 1-1-1 relationship

1.4 Overcoming Traditional AI Solutions

During collaboration with Caufero it emerged that ChatGPT and other AIs suggest traditional solutions that **do not solve the ontological problem:**

Ineffective AI Solutions:

- Classical normalization with hundreds of tables
- JSON with partitioning for performance
- Separation between definition and instance
- Approaches that maintain dualistic fragmentation

Why They Fail:

- Perpetuate binary subject-object logic
- Don't understand tripartite ontological unity
- Solve complexity by increasing it instead of simplifying it

CHAPTER 2: UNIVERSAL EXISTENTIAL ALGORITHM

2.1 TAKE-MAKE-PLACE: The Pattern of Every Action

Every action in the universe, from subatomic particle to business organization, follows the same existential algorithm:

ELEMENTARY ACTION = TAKE → MAKE → PLACE

TAKE (Input/Preparation)

- **What do I take?** Raw material, information, resources
- **With whom do I take it?** People, tools, machines
- **From where do I take it?** Environment, previous process, warehouse

MAKE (Transformation/Process)

- **Applied movement:** Physical, mental, temporal energy
- **Transformation process:** Change of state, form, meaning
- **Conscious action:** Application of will, knowledge, method

PLACE (Output/Result)

- **What do I place?** Transformed product, new semi-finished product, information
- **Where do I place it?** In the world, for the next process, in the database
- **With what quality?** Verifiable characteristics of the result

2.2 Concrete Example: Marco Glues the Page

SCENARIO: Marco assembles a page of the color folder

TAKE:

RAW MATERIALS:
- Pre-printed page
- Hot-melt glue

PEOPLE RESOURCES:
- Marco (specialized operator)
- Assembly process knowledge

TOOLS/MEANS:
- Thermal glue gun
- Ergonomic work table
- Hands (biological tool)

MAKE:

TRANSFORMATION SEQUENCE:
- Position page on work surface
- Heat gun to optimal temperature
- Dispense glue along edge with precision
- Place gun in safety support
- Fold page following guide lines
- Press manually for perfect adhesion
- Verify gluing quality

PLACE:

PRODUCT RESULT:
└─ ─ Correctly assembled page
├─── Quality: Perfectly adherent edges
├─── Time taken: 45 seconds
├─── Standard achieved: Meets specifications
└─── Status: Ready for final assembly

2.3 Optimal Granularity vs Infinite Micro-Actions

INEFFECTIVE GRANULARITY (Micro-management)

Extend arm → Open hand → Position fingers → Grasp object →
Lift → Rotate wrist → Approach → Release → ...

Problem: Infinite escalation of details without managerial value

OPTIMAL GRANULARITY (Finished Action)

GLUE_PAGE = Complete TAKE-MAKE-PLACE cycle
├─── Associated time: 45 seconds standard
├─── Resources involved: Marco + gun + materials
├─── Measurable result: Page assembled to specification
├─── Managerial value: Orderable and traceable unit
└─── Unique DNA: OPE25042 (Specific operation)

2.4 Infinite Composition and Decomposition

Principle of Recursive Composition:

ELEMENTARY ACTION (Take-Make-Place)
↕ composes into
PHASE/OPERATION (Sequence of elementary actions)
↕ composes into
COMPLETE PROCESS (Folder assembly)
↕ composes into
MACRO-PROCESS (Customer order management)
↕ composes into
MEGA-PROCESS (Business management)

CHAPTER 3: TECHNICAL ARCHITECTURE 3 UNIVERSAL TABLES

3.1 CMP-ETY-LOG: The Computational Trinity

The 3P3 system digitizes the tripartite ontology through **exactly 3 universal tables** that replace hundreds of fragmented tables in traditional systems.

CMP (Components/Catalog)

Role: Universal repository of everything that **EXISTS** in the system (templates and instances)

```
sql
CREATE TABLE CMP (
    CmpID VARCHAR(20) PRIMARY KEY,      -- DNA: PHO25001, PHO25008, BOM25001
    CmpType VARCHAR(20) NOT NULL,      -- PROCESS|PROCESS_INSTANCE|
ATTRIBUTE|MATERIAL|CYCLE
    CmpName VARCHAR(200),                  -- Descriptive name
    ParentID VARCHAR(20),        -- Hierarchy (attributes→process, instance→template)
    ProcessType VARCHAR(20),             -- PHO|RCH|BOM|SWA|CDL
    JSONStructure TEXT,                  -- Specific data (template or instance)
    IsTemplate BOOLEAN DEFAULT TRUE,      -- TRUE=template, FALSE=real instance
    CreatedAt TIMESTAMP,
    CreatedBy VARCHAR(50)
);
```

CMP Content:

- **TEMPLATES** (IsTemplate=TRUE): Process definitions, attributes, materials, cycles
- **INSTANCES** (IsTemplate=FALSE): Real phone calls, specific BOMs, concrete orders with ALL business data

ETY (Entity/Orchestrator)

Role: Workflow and orchestration manager - NO business data

```
sql
CREATE TABLE ETY (
   EtyID VARCHAR(20) PRIMARY KEY,       -- DNA: PHO25008_ORCH, BOM25001_ORCH
   ProcessType VARCHAR(50),             -- PHONE_EXECUTION|BOM_ORCHESTRATION|ORDER_WORKFLOW
   Status VARCHAR(20),                  -- TO_DO|IN_PROGRESS|DONE|CANCELLED
   LinkedCMP VARCHAR(20),               -- Points to instance in CMP
   ParentETY VARCHAR(20),               -- Workflow chain (RCH→OFC→OCL)
   JSONData JSON,                       -- ONLY orchestration data (NO business data!)
   CreatedAt TIMESTAMP,
   UpdatedAt TIMESTAMP,
   CompletedAt TIMESTAMP,
   CreatedBy VARCHAR(50),
   ResponsibleID VARCHAR(50)
);
```

ETY Content:

- **WORKFLOW ORCHESTRATION** and state transitions
- **ASSIGNMENTS** and responsibilities
- **EXECUTION METADATA** (times, performance, triggers)
- **NEVER BUSINESS DATA** (no caller_name, duration, product_code, etc.)

LOG (Actions/Universal Registry)

Role: Immutable memory of every action - total system traceability

```sql
CREATE TABLE LOG (
   LogID VARCHAR(40) PRIMARY KEY,       -- ACT_20250824_143022_001
   ActionType VARCHAR(20) NOT NULL,     -- CREATE|UPDATE|DELETE|STATUS
   TargetTable VARCHAR(3),              -- CMP|ETY
   TargetID VARCHAR(30),                -- Target entity ID
   TargetType VARCHAR(20),              -- PHO|BOM|CDL etc
   FieldChanged VARCHAR(50),            -- Modified field
   OldValue TEXT,                       -- Previous value
   NewValue TEXT,                       -- New value
   ActionJSON TEXT,                     -- Complex details
   Timestamp TIMESTAMP DEFAULT NOW(),
   UserID VARCHAR(30),
   SessionID VARCHAR(50)                -- Groups related actions
);
```

3.2 [1-1-1] Relationship Always Maintained

Inviolable Principle: Every manifestation in the system generates related records in the 3 tables according to the correct ontological pattern.

```
1 USER ACTION = 1 CMP (instance) + 1 ETY (orchestration) + 1 LOG (trace)
```

CORRECT Concrete Examples:

Define PHO Process Template:

CMP: PHO_TEMPLATE (phone call process definition, IsTemplate=TRUE)
ETY: PHO_TEMPLATE_ORCH (template creation orchestration)
LOG: ACT_TEMPLATE_CREATION (action registration)

Execute Real Phone Call:

CMP: PHO25008 (phone instance with real data, IsTemplate=FALSE)
ETY: PHO25008_ORCH (phone call workflow orchestration)
LOG: ACT_PHONE_EXECUTION (execution registration)

Update Phone Call Status:

CMP: PHO25008 (business data unchanged)
ETY: PHO25008_ORCH (status: TO_DO → COMPLETED)
LOG: ACT_STATUS_UPDATE (status change registration)

3.3 Hierarchical DNA for Unique Identity

Standard Format: `PRXYYNNNN`

Components:

- **PRX**: Process type (PHO, BOM, CDL, RCH, OFC...)
- **YY**: Year (25 for 2025)
- **NNNN**: Sequential number (0001, 0002...)

DNA Examples:

```
PHO_TEMPLATE → Phone call process template
PHO25008 → Eighth real phone instance of 2025
PHO25008_ORCH → Orchestration of eighth phone call
BOM25001 → First bill of materials of 2025
BOM25001_ORCH → First bill of materials orchestration
```

3.4 JSON vs Structured Fields: Implementation Choices

Technical Dilemma: How to store process-specific data while maintaining flexibility?

ADOPTED SOLUTION: Structured JSON with Ontological Separation

CMP::JSONStructure (Business Data):

```
json
{
   "caller_name": "Mario Rossi",
   "phone_number": "+39 333 1234567",
   "duration_minutes": 15,
   "outcome": "Sale",
   "notes": "Interested in 120-strand folder",
   "callback_date": "2025-08-28",
   "lead_quality": "HOT"
}
```

ETY::JSONData (Orchestration Only):

```
json
{
   "orchestration_type": "phone_execution",
   "workflow_status": "completed",
   "execution_context": "sales_campaign",
   "triggers": ["create_appointment", "send_brochure"],
   "performance_metrics": {
      "start_time": "14:30:00",
      "end_time": "14:45:00",
      "sla_met": true
   }
}
```

CHAPTER 4: THE DISCOVERY OF 3 RECURSIVE ONTOLOGICAL LEVELS

4.1 The Illumination of Fractal Structure

During implementation analysis, a fundamental discovery emerged: **the 3P3 system manifests recursively on 3 distinct ontological levels.**

LEVEL 1: ELEMENTARY CRUD
DATABASE ↔↔ GUI ↔↔ INFO
(Aspect) (Nature) (Entity)

LEVEL 2: PROCESS META-MODEL
PROCESS ↔↔ ATTRIBUTES ↔↔ GUI
(Definition) (Structure) (Interface)

LEVEL 3: OPERATIONAL INSTANTIATION
PROCESS ↔↔ GUI ↔↔ INFO
(Execution) (Interaction) (Result)

4.2 Implementation of the 3 Levels

LEVEL 2: Meta-Model (Template Construction)

STEP 1: Create PHO Process Template

```
INTERFACE: "Process Builder"
USER INPUT:
 ├─── Process name: "Phone Call Management"
 ├─── Code: "PHO"
 ├─── Description: "Customer phone call management"
 └─── Category: "COMMUNICATION"

AUTOMATIC RESULT:
 ├─── CMP: PHO_TEMPLATE (process definition, IsTemplate=TRUE)
 ├─── ETY: PHO_TEMPLATE_ORCH (creation orchestration)
 └─── LOG: ACT_PROCESS_CREATION (registration)
```

STEP 2: Define 5 Attributes

```
INTERFACE: "Attribute Builder"
USER INPUT (repeated 5 times):
 ├─── Attribute 1: caller_name (TEXT, required)
 ├─── Attribute 2: duration_minutes (NUMBER, min:0, max:999)
 ├─── Attribute 3: outcome (SELECT: Sale|Follow-up|Info|Complaint)
 ├─── Attribute 4: phone_number (TEXT, format:phone)
 └─── Attribute 5: notes (TEXTAREA, optional)

AUTOMATIC RESULT:
 ├─── CMP: 5 records ATTR_PHO_01→05 (attribute definitions, IsTemplate=TRUE)
 ├─── ETY: 5 creation orchestration records
 └─── LOG: 5 ACT_ATTR_CREATION records
```

LEVEL 3: Operational Instantiation (Daily Use)

STEP 4: Execute Real Phone Call

```
INTERFACE: "Phone Call GUI" (generated from PHO template)
USER INPUT:
 ├─── caller_name: "Mario Rossi"
 ├─── duration_minutes: 15
 ├─── outcome: "Sale"
 ├─── phone_number: "+39 333 1234567"
 └─── notes: "Interested in 120-strand folder"

CORRECT AUTOMATIC RESULT:
 ├─── CMP: PHO25008 (phone instance with real data, IsTemplate=FALSE)
 ├─── ETY: PHO25008_ORCH (workflow orchestration)
 └─── LOG: ACT_PHONE_EXECUTION (action registration on CMP)
```

4.3 Sequential DNA for Levels

CORRECT Complete DNA Sequence:

```
PHO_TEMPLATE → PHO process template (Level 2, IsTemplate=TRUE)
ATTR_PHO_01 → caller_name attribute definition (Level 2)
ATTR_PHO_02 → duration attribute definition (Level 2)
ATTR_PHO_03 → outcome attribute definition (Level 2)
ATTR_PHO_04 → phone_number attribute definition (Level 2)
ATTR_PHO_05 → notes attribute definition (Level 2)
REL_PHO_ATTRS → process-attributes connection (Level 2)
PHO25008 → First phone call Mario Rossi (Level 3, IsTemplate=FALSE)
PHO25008_ORCH → First phone call orchestration (Level 3)
PHO25009 → Second phone call Anna Verdi (Level 3, IsTemplate=FALSE)
PHO25009_ORCH → Second phone call orchestration (Level 3)
```

CHAPTER 5: ONTOLOGICAL TEST - PHONE CALL PROCESS

5.1 PHO as Elementary Test Entity

Why Phone Call is Perfect for Testing:

1. **SIMPLICITY**: Immediately understandable process
2. **COMPLETENESS**: Contains all 3P3 aspects (People-Processes-Products)
3. **MEASURABILITY**: Clear and verifiable attributes
4. **SCALABILITY**: Same principle from simple process to complex BOM
5. **UNIVERSALITY**: Every company makes phone calls - universally applicable test

5.2 Complete Setup: Process Definition + 5 Attributes

FUNDAMENTAL ONTOLOGICAL CORRECTION APPLIED

3P3 PRINCIPLE: All real instances (operational phone calls) go in CMP with IsTemplate=FALSE. ETY manages only orchestration.

PHASE 1: Meta-Model Setup (Level 2)

Step 1: Define PHO Template Process

```
sql
-- CMP TABLE: Process template
INSERT INTO CMP VALUES (
   'PHO_TEMPLATE',              -- CmpID
   'PROCESS',                   -- CmpType
   'Phone Call Management',     -- CmpName
   NULL,                        -- ParentID
   'PHO',                       -- ProcessType
```

```
    '{"category":"COMMUNICATION","priority":"HIGH","automation":"MANUAL"}', -- JSONStructure
    TRUE,                       -- IsTemplate
    NOW(),                      -- CreatedAt
    'admin@kooltool.com'        -- CreatedBy
);

-- ETY TABLE: Template creation orchestration
INSERT INTO ETY VALUES (
    'PHO_TEMPLATE_ORCH',        -- EtyID
    'TEMPLATE_CREATION',        -- ProcessType
    'COMPLETED',                -- Status
    'PHO_TEMPLATE',             -- LinkedCMP
    NULL,                       -- ParentETY
    '{"action":"template_created","target":"PHO_TEMPLATE","interface":"Process_Builder"}', -- JSONData
    NOW(),                      -- CreatedAt
    NOW(),                      -- UpdatedAt
    NOW(),                      -- CompletedAt
    'admin@kooltool.com',       -- CreatedBy
    'admin@kooltool.com'        -- ResponsibleID
);

-- LOG TABLE
INSERT INTO LOG VALUES (
    'ACT_20250824_140000_001',  -- LogID
    'CREATE',                   -- ActionType
    'CMP',                      -- TargetTable
    'PHO_TEMPLATE',             -- TargetID
    'PROCESS',                  -- TargetType
    NULL,                       -- FieldChanged
    NULL,                       -- OldValue
```

```
  'Phone Call Management template created', -- NewValue
  '{"user":"admin","interface":"Process_Builder","session":"SES001"}', -- ActionJSON
  NOW(),                    -- Timestamp
  'admin@kooltool.com',          -- UserID
  'SES001'                  -- SessionID
);
```

5.3 Execution: 10,000 Instances for Performance Testing

PHASE 2: Operational Use (Level 3) - CORRECT VERSION

Automatic Script for Mass Testing - ONTOLOGICALLY CORRECT

```
sql
-- FileMaker PROCEDURE for generating 10,000 phone calls IN CMP
SCRIPT: "Generate_Test_Phone_Calls_CORRECT"


SET VARIABLE [$counter; VALUE: 1]
SET VARIABLE [$names; VALUE: List("Mario Rossi";"Anna Verdi";"Giuseppe Blu";"Laura Rosa";"Franco Neri")]
SET VARIABLE [$outcomes; VALUE: List("Sale";"Follow-up";"Info";"Complaint";"No Answer")]
SET VARIABLE [$companies; VALUE: List("LILA COSMETIC";"BEAUTY SPA";"HAIR STUDIO";"COLOR LAB";"STYLE CENTER")]


LOOP
  EXIT LOOP IF [$counter > 10000]


  -- Generate realistic random data
  SET VARIABLE [$caller; VALUE: GetValue($names; Int(Random * 5) + 1)]
  SET VARIABLE [$outcome; VALUE: GetValue($outcomes; Int(Random * 5) + 1)]
  SET VARIABLE [$company; VALUE: GetValue($companies; Int(Random * 5) + 1)]
  SET VARIABLE [$duration; VALUE: Int(Random * 45) + 5] -- 5-50 minutes
```

```
SET VARIABLE [$phone; VALUE: "+39 333 " & Int(Random * 9000000) + 1000000]


-- Create progressive DNA for INSTANCE
SET VARIABLE [$dna; VALUE: "PHO" & Right(Year(Get(CurrentDate)); 2) & Right("0000" & ($counter + 100); 4)]


-- INSERT CMP: REAL PHONE INSTANCE (IsTemplate=FALSE)
NEW RECORD [CMP]
SET FIELD [CMP::CmpID; $dna]
SET FIELD [CMP::CmpType; "PROCESS_INSTANCE"]
SET FIELD [CMP::CmpName; "Phone Call " & $caller & " - " & $company]
SET FIELD [CMP::ParentID; "PHO_TEMPLATE"] -- Derives from template
SET FIELD [CMP::ProcessType; "PHO"]
SET FIELD [CMP::IsTemplate; FALSE] -- REAL INSTANCE!
SET FIELD [CMP::JSONStructure;
  "{" &
  "\"caller_name\":\"" & $caller & "\"," &
  "\"duration_minutes\":" & $duration & "," &
  "\"outcome\":\"" & $outcome & "\"," &
  "\"phone_number\":\"" & $phone & "\"," &
  "\"notes\":\"Test call for " & $company & " - automated generation\"," &
  "\"company\":\"" & $company & "\"," &
  "\"lead_quality\":\"" & If($outcome = "Sale"; "HOT"; If($outcome = "Follow-up"; "WARM"; "COLD")) & "\"" &
  "}"
]
SET FIELD [CMP::CreatedAt; Get(CurrentTimeStamp)]
SET FIELD [CMP::CreatedBy; "test_generator"]
COMMIT RECORDS


-- INSERT ETY: INSTANCE ORCHESTRATION (NO BUSINESS DATA!)
NEW RECORD [ETY]
```

```
 SET FIELD [ETY::EtyID; $dna & "_ORCH"]
 SET FIELD [ETY::ProcessType; "PHONE_EXECUTION"]
 SET FIELD [ETY::Status; "COMPLETED"]
 SET FIELD [ETY::LinkedCMP; $dna] -- Points to CMP instance!
 SET FIELD [ETY::JSONData;
  "{" &
  "\"orchestration_type\":\"phone_call_execution\"," &
  "\"cmp_instance\":\"" & $dna & "\"," &
  "\"workflow_status\":\"completed\"," &
  "\"execution_context\":\"test_generation\"," &
  "\"performance\":{\"sla_met\":true,\"quality_score\":95}" &
  "}"
 ]
 SET FIELD [ETY::CreatedAt; Get(CurrentTimeStamp)]
 SET FIELD [ETY::CreatedBy; "test_generator"]
 COMMIT RECORDS

 -- INSERT LOG automatic
 NEW RECORD [LOG]
 SET FIELD [LOG::LogID; "ACT_" & Substitute(Get(CurrentTimeStamp); [" ";"_"]; [":";""])] &
"_" & Right("000" & $counter; 3)]
 SET FIELD [LOG::ActionType; "CREATE"]
 SET FIELD [LOG::TargetTable; "CMP"] -- Target is CMP, not ETY!
 SET FIELD [LOG::TargetID; $dna]
 SET FIELD [LOG::TargetType; "PHO"]
 SET FIELD [LOG::NewValue; "Phone call instance created: " & $caller]
 SET FIELD [LOG::UserID; "test_generator"]
 COMMIT RECORDS

 SET VARIABLE [$counter; VALUE: $counter + 1]
END LOOP
```

Generated Phone Call Examples

Phone Call 1 - Instance in CMP:

```
sql
-- CMP: REAL PHONE INSTANCE
INSERT INTO CMP VALUES (
   'PHO25101',                    -- CmpID (instance)
   'PROCESS_INSTANCE',            -- CmpType = process instance
   'Phone Call Mario Rossi - LILA COSMETIC', -- Descriptive CmpName
   'PHO_TEMPLATE',                -- ParentID (derives from template)
   'PHO',                         -- ProcessType
   '{
      "caller_name": "Mario Rossi",
      "duration_minutes": 15,
      "outcome": "Sale",
      "phone_number": "+39 333 1234567",
      "notes": "Interested in 120-strand folder. Appointment set for Tuesday.",
      "company": "LILA COSMETIC",
      "lead_quality": "HOT",
      "lead_score": 85
   }',                            -- JSONStructure (BUSINESS DATA HERE!)
   FALSE,                         -- IsTemplate = FALSE (real instance!)
   '2025-08-24 14:30:00',         -- CreatedAt
   'luca@kooltool.com'            -- CreatedBy
);

-- ETY: INSTANCE ORCHESTRATION (NO BUSINESS DATA!)
INSERT INTO ETY VALUES (
   'PHO25101_ORCH',               -- EtyID orchestration
   'PHONE_EXECUTION',             -- ProcessType
   'COMPLETED',                   -- Status
   'PHO25101',                    -- LinkedCMP (points to CMP instance!)
```

```
    NULL,                       -- ParentETY
    '{
      "orchestration_type": "phone_call_execution",
      "cmp_instance": "PHO25101",
      "workflow_status": "completed",
      "responsible_user": "luca@kooltool.com",
      "execution_time_seconds": 900,
      "follow_up_required": true,
      "triggers": ["create_appointment", "send_brochure"]
    }',                         -- JSONData (ORCHESTRATION ONLY!)
    '2025-08-24 14:30:00',      -- CreatedAt
    '2025-08-24 14:45:00',      -- UpdatedAt
    '2025-08-24 14:45:00',      -- CompletedAt
    'luca@kooltool.com',        -- CreatedBy
    'luca@kooltool.com'         -- ResponsibleID
);

-- LOG: Action registration on CMP
INSERT INTO LOG VALUES (
    'ACT_20250824_143000_001',  -- LogID
    'CREATE',                   -- ActionType
    'CMP',                      -- TargetTable (CMP!)
    'PHO25101',                 -- TargetID
    'PHO',                      -- TargetType
    NULL, NULL,                 -- FieldChanged, OldValue
    'Phone call instance created: Mario Rossi - outcome: Sale', -- NewValue
    '{"duration":15,"outcome":"Sale","user":"luca","company":"LILA_COSMETIC"}', --
ActionJSON
    '2025-08-24 14:30:00',      -- Timestamp
    'luca@kooltool.com',        -- UserID
    'SES_CALL_001'              -- SessionID
);
```

5.4 Verification: Functional Composition/Decomposition

TEST 1: CORRECT 1-1-1 Relationship Verification

```
sql
-- CORRECT integrity verification query
SELECT
   'CMP Templates' as Type,
   COUNT(*) as Count
FROM CMP
WHERE ProcessType = 'PHO' AND IsTemplate = TRUE
UNION ALL
SELECT
   'CMP Instances' as Type,
   COUNT(*) as Count
FROM CMP
WHERE ProcessType = 'PHO' AND IsTemplate = FALSE
UNION ALL
SELECT
   'ETY Orchestrations' as Type,
   COUNT(*) as Count
FROM ETY
WHERE ProcessType LIKE '%PHO%' OR LinkedCMP LIKE 'PHO%'
UNION ALL
SELECT
   'LOG Actions' as Type,
   COUNT(*) as Count
FROM LOG
WHERE TargetType = 'PHO';

-- CORRECT EXPECTED RESULT:
```

```
-- CMP Templates: 7 (1 process + 5 attributes + 1 relation)
-- CMP Instances: 10,000 (real phone calls)
-- ETY Orchestrations: 10,007 (orchestrations for everything)
-- LOG Actions: 10,007+ (complete traceability)
```

TEST 2: JSON Query Performance on CMP (CORRECT)

```
sql
-- CORRECT instance aggregation query (from CMP, not ETY!)
SELECT
    JSON_UNQUOTE(JSON_EXTRACT(JSONStructure, '$.outcome')) as Outcome,
    COUNT(*) as Count,
    AVG(JSON_EXTRACT(JSONStructure, '$.duration_minutes')) as Avg_Duration,
    AVG(JSON_EXTRACT(JSONStructure, '$.lead_score')) as Avg_Lead_Score
FROM CMP
WHERE ProcessType = 'PHO'
  AND IsTemplate = FALSE  -- Only real instances!
  AND CmpType = 'PROCESS_INSTANCE'
GROUP BY JSON_EXTRACT(JSONStructure, '$.outcome')
ORDER BY Count DESC;

-- EXPECTED RESULT:
-- Sale: ~2,000 calls, 18 min average, lead score 82
-- Follow-up: ~2,500 calls, 16 min average, lead score 65
-- Info: ~2,000 calls, 12 min average, lead score 45
-- Complaint: ~1,500 calls, 25 min average, lead score 20
-- No Answer: ~2,000 calls, 3 min average, lead score 10
```

CHAPTER 5bis: AUTOMATIC ONTOLOGICAL BENCHMARKING

5bis.1 Intelligent Benchmarking Principle

Fundamental Innovation: 3P3 ontology enables automatic intelligent comparisons between similar processes for optimization and suggestions based on emerging patterns.

CORRECT Benchmarking Query (on CMP, not ETY):

```
sql
-- Automatic query to identify similar processes (FROM CMP!)
SELECT
    ProcessType,
    JSON_UNQUOTE(JSON_EXTRACT(JSONStructure, '$.company_sector')) as Sector,
    COUNT(*) as Total_Instances,
    AVG(JSON_EXTRACT(JSONStructure, '$.duration_minutes')) as Avg_Duration,
    AVG(JSON_EXTRACT(JSONStructure, '$.lead_score')) as Avg_LeadScore
FROM CMP
WHERE IsTemplate = FALSE
  AND ProcessType = 'PHO'
  AND CmpType = 'PROCESS_INSTANCE'
GROUP BY ProcessType, JSON_EXTRACT(JSONStructure, '$.company_sector')
```

CHAPTER 5ter: USER-CENTRIC MULTI-DIMENSIONAL FILTERING

5ter.1 Advanced Filtering System

CORRECT Dynamic Query (interrogates CMP for business data):

```
sql
-- "All PETTENON client phone calls in August with positive commercial outcome"
SELECT
    c.CmpID,
    c.CmpName,
    JSON_UNQUOTE(JSON_EXTRACT(c.JSONStructure, '$.caller_name')) as Client,
    JSON_UNQUOTE(JSON_EXTRACT(c.JSONStructure, '$.outcome')) as Outcome,
    JSON_EXTRACT(c.JSONStructure, '$.duration_minutes') as Duration,
    JSON_EXTRACT(c.JSONStructure, '$.lead_score') as LeadScore,
    c.CreatedAt as CallDate,
    c.CreatedBy as Operator,
    e.Status as WorkflowStatus
FROM CMP c
LEFT JOIN ETY e ON e.LinkedCMP = c.CmpID
WHERE c.ProcessType = 'PHO'
  AND c.IsTemplate = FALSE
  AND (JSON_EXTRACT(c.JSONStructure, '$.caller_name') LIKE '%PETTENON%'
       OR JSON_EXTRACT(c.JSONStructure, '$.company') LIKE '%PETTENON%')
  AND MONTH(c.CreatedAt) = 8
  AND YEAR(c.CreatedAt) = 2025
  AND JSON_EXTRACT(c.JSONStructure, '$.outcome') IN ('Sale', 'Follow-up',
'Appointment')
ORDER BY JSON_EXTRACT(c.JSONStructure, '$.lead_score') DESC, c.CreatedAt DESC;
```

CHAPTER 6: ESCALATION TO BOM COMPLEXITY

6.1 From PHO to BOM: Same Principle, Different Scale

Fundamental Ontological Demonstration: If the 3P3 system works for simple process (phone call), it **must work identically** for complex process (color folder BOM).

6.2 Multi-Level Hierarchy Management

CORRECT Implementation in 3P3 System:

CMP: BOM Template

```
sql
INSERT INTO CMP VALUES (
   'BOM_TEMPLATE', 'PROCESS', 'Bill of Materials Management', NULL, 'BOM',
   '{"category":"PRODUCTION","complexity":"HIGH","supports_hierarchy":true}',
   TRUE, NOW(), 'admin@kooltool.com'
);
```

CMP: Specific BOM Instance (CORRECT - in CMP, not ETY!)

```
sql
INSERT INTO CMP VALUES (
   'BOM25001',                    -- CmpID
   'PROCESS_INSTANCE',            -- CmpType (instance!)
   'BOM 4-Panel Folder',          -- CmpName
   'BOM_TEMPLATE',                -- ParentID
   'BOM',                         -- ProcessType
   '{
      "product_code": "FOLDER_4_PANELS",
      "version": "2.1",
      "estimated_cost": 76.50,
```

```
    "estimated_time_minutes": 45,
    "components": [
      {
        "id": "CDL25010",
        "type": "CYCLE",
        "description": "PREPARE_MATERIALS",
        "level": 1,
        "children": [
          {"id": "MAT25001", "type": "MATERIAL", "quantity": 0.05, "unit": "kg"},
          {"id": "MAT25002", "type": "MATERIAL", "quantity": 4, "unit": "sheets"},
          {"id": "MAT25003", "type": "MATERIAL", "quantity": 103, "unit": "strands"}
        ]
      }
    ]
  }',                          -- JSONStructure (BOM DATA HERE!)
  FALSE,                       -- IsTemplate = FALSE (instance!)
  NOW(), 'designer@kooltool.com'
);
```

ETY: BOM Orchestration (CORRECT - workflow only!)

```
sql
INSERT INTO ETY VALUES (
  'BOM25001_ORCH',             -- EtyID
  'BOM_ORCHESTRATION',         -- ProcessType
  'ACTIVE',                    -- Status
  'BOM25001',                  -- LinkedCMP (points to CMP instance!)
  NULL,                        -- ParentETY
  '{
    "orchestration_type": "bom_management",
    "workflow_phase": "design_approved",
```

```
    "validation_status": "approved",
    "triggers": ["order_materials", "schedule_production"],
    "approval_chain": ["designer", "production_manager", "quality"]
  }',                          -- JSONData (ORCHESTRATION ONLY!)
  NOW(), NOW(), NULL, 'designer@kooltool.com', 'production@kooltool.com'
);
```

6.3 Contextual Specialization vs Duplication

SOLUTION C: 3P3 Hybrid (Pure Ontology) - CORRECT

Principle: The material in BOM **IS a new entity** ontologically distinct from the generic material

```
sql
-- CMP: Specialized material instance in BOM context
INSERT INTO CMP VALUES (
  'MATBOM25001',               -- CmpID
  'MATERIAL_INSTANCE',            -- CmpType
  'Glue in 4-Panel Folder',       -- CmpName
  'MAT25001',                  -- ParentID (base material template)
  'BOM',                       -- ProcessType
  '{
    "base_material": "MAT25001",
    "specialized_context": "FOLDER_4_PANELS",
    "optimized_quantity": 0.05,
    "application_method": "hot_melt_gun",
    "quality_requirements": "food_grade_safe",
    "bom_parent": "BOM25001"
  }',                          -- JSONStructure (specialized material data)
  FALSE,                          -- IsTemplate = FALSE (instance!)
```

```
    NOW(), 'designer@kooltool.com'
);

-- ETY: Material usage orchestration
INSERT INTO ETY VALUES (
    'MATBOM25001_ORCH',             -- EtyID
    'MATERIAL_USAGE',               -- ProcessType
    'IN_PROGRESS',                  -- Status
    'MATBOM25001',                  -- LinkedCMP (points to material instance!)
    'BOM25001_ORCH',                -- ParentETY (linked to BOM)
    '{
        "orchestration_type": "material_allocation",
        "allocation_status": "reserved",
        "batch_tracking": "COL240815",
        "operator_assigned": "Marco",
        "usage_phase": "assembly"
    }',                             -- JSONData (ORCHESTRATION ONLY!)
    NOW(), NULL, NULL, 'production@kooltool.com', 'Marco@kooltool.com'
);
```

CHAPTER 7: SPECIFIC TECHNICAL SOLUTIONS

7.1 FileMaker: Script and Layout Implementation

System Core Scripts - CORRECT VERSION

Script 1: Create_Universal_Entity - CORRECT

```
filemaker
# INPUT: $ProcessType, $ProcessSubType, $JSONData, $TemplateID
# OUTPUT: New CMP record (instance) + ETY (orchestration) + LOG

# STEP 1: Generate unique DNA for instance
Set Variable [$Year; Value: Right(Year(Get(CurrentDate)); 2)]
Set Variable [$Sequence; Value:
   GetAsNumber(ExecuteSQL(
      "SELECT MAX(CAST(SUBSTR(CmpID, -4) AS INTEGER)) FROM CMP
       WHERE CmpID LIKE '" & $ProcessType & $Year & "%'"; ""; ""
   )) + 1
]
Set Variable [$NewDNA; Value: $ProcessType & $Year & Right("0000" & $Sequence; 4)]

# STEP 2: Verify existing template
If [IsEmpty($TemplateID)]
   Set Variable [$TemplateID; Value: $ProcessType & "_TEMPLATE"]
End If

Go to Layout ["CMP"]
Enter Find Mode []
Set Field [CMP::CmpID; $TemplateID]
Set Field [CMP::IsTemplate; TRUE]
```

```
Perform Find []
If [Get(FoundCount) = 0]
   Show Custom Dialog ["Error"; "Template " & $TemplateID & " not found"]
   Exit Script [Text Result: "ERROR_TEMPLATE_NOT_FOUND"]
End If

# STEP 3: Create INSTANCE in CMP (BUSINESS DATA!)
Go to Layout ["CMP"]
New Record/Request
Set Field [CMP::CmpID; $NewDNA]
Set Field [CMP::CmpType; "PROCESS_INSTANCE"]
Set Field [CMP::CmpName; $ProcessType & " Instance " & $NewDNA]
Set Field [CMP::ParentID; $TemplateID]
Set Field [CMP::ProcessType; $ProcessType]
Set Field [CMP::IsTemplate; FALSE]  # REAL INSTANCE!
Set Field [CMP::JSONStructure; $JSONData]  # BUSINESS DATA HERE!
Set Field [CMP::CreatedAt; Get(CurrentTimeStamp)]
Set Field [CMP::CreatedBy; Get(UserName)]
Commit Records

# STEP 4: Create ORCHESTRATION in ETY (NO BUSINESS DATA!)
Go to Layout ["ETY"]
New Record/Request
Set Field [ETY::EtyID; $NewDNA & "_ORCH"]
Set Field [ETY::ProcessType; $ProcessType & "_EXECUTION"]
Set Field [ETY::Status; "TO_DO"]
Set Field [ETY::LinkedCMP; $NewDNA]  # Points to CMP instance!
Set Field [ETY::JSONData;
   "{" &
   "\"orchestration_type\":\"" & $ProcessType & "_execution\"," &
   "\"cmp_instance\":\"" & $NewDNA & "\"," &
```

```
    "\"workflow_status\":\"initiated\"," &
    "\"created_from_template\":\"" & $TemplateID & "\"" &
    "}"
]  # ORCHESTRATION ONLY!
Set Field [ETY::CreatedAt; Get(CurrentTimeStamp)]
Set Field [ETY::CreatedBy; Get(UserName)]
Commit Records

# STEP 5: Create automatic LOG
Perform Script ["Create_LOG_Entry"; Parameter:
    "CREATE|CMP|" & $NewDNA & "|" & $ProcessType & "|Instance created from template " & $TemplateID
]

# STEP 6: Return DNA for subsequent use
Exit Script [Text Result: $NewDNA]
```

Script 2: Update_Entity_Status - CORRECT

```
filemaker
# INPUT: $CmpID, $NewStatus, $JSONUpdates
# OUTPUT: Updated ETY (status) + Updated CMP (if business data) + LOG

# STEP 1: Find ETY orchestration
Go to Layout ["ETY"]
Enter Find Mode []
Set Field [ETY::LinkedCMP; $CmpID]
Perform Find []
If [Get(FoundCount) = 0]
    Exit Script [Text Result: "ERROR_ORCHESTRATION_NOT_FOUND"]
End If
```

```
# STEP 2: Save previous values for LOG
Set Variable [$OldStatus; Value: ETY::Status]
Set Variable [$EtyID; Value: ETY::EtyID]

# STEP 3: Update STATUS in ETY (orchestration)
Set Field [ETY::Status; $NewStatus]
Set Field [ETY::UpdatedAt; Get(CurrentTimeStamp)]

# Update orchestration metadata
Set Variable [$OldJSONData; Value: ETY::JSONData]
Set Variable [$NewJSONData; Value: JSONSetElement($OldJSONData;
   ["workflow_status"; $NewStatus; JSONString];
   ["last_update"; Get(CurrentTimeStamp); JSONString]
)]
Set Field [ETY::JSONData; $NewJSONData]

If [$NewStatus = "DONE"]
   Set Field [ETY::CompletedAt; Get(CurrentTimeStamp)]
End If
Commit Records

# STEP 4: If there are business updates, they go in CMP!
If [Not IsEmpty($JSONUpdates)]
   Go to Layout ["CMP"]
   Enter Find Mode []
   Set Field [CMP::CmpID; $CmpID]
   Set Field [CMP::IsTemplate; FALSE]
   Perform Find []
   If [Get(FoundCount) > 0]
      Set Variable [$OldBusinessData; Value: CMP::JSONStructure]
```

```
    Set Variable [$NewBusinessData; Value: JSONMerge($OldBusinessData;
$JSONUpdates)]
    Set Field [CMP::JSONStructure; $NewBusinessData]
    Commit Records
  End If
End If

# STEP 5: Create LOG of modification
Perform Script ["Create_LOG_Entry"; Parameter:
  "UPDATE|ETY|" & $EtyID & "|" & ETY::ProcessType & "|Status: " & $OldStatus & " → " &
$NewStatus
]

Exit Script [Text Result: "SUCCESS"]
```

7.2 Performance: JSON Optimizations and Queries

Performance Calculated Indexes - CORRECT (on CMP)

```
filemaker
# CMP Calculated Field: idx_CallerName
FORMULA: If(IsTemplate = FALSE and ProcessType = "PHO";
    JSONGetElement(JSONStructure; "caller_name"); "")
INDEXED: Yes, Stored, Non Nulls
USE: Fast queries on caller name

# CMP Calculated Field: idx_Outcome
FORMULA: If(IsTemplate = FALSE and ProcessType = "PHO";
    JSONGetElement(JSONStructure; "outcome"); "")
INDEXED: Yes, Stored, All Values
USE: Aggregations by outcome type
```

```
# CMP Calculated Field: idx_Duration
FORMULA: If(IsTemplate = FALSE and ProcessType = "PHO";
        GetAsNumber(JSONGetElement(JSONStructure; "duration_minutes")); "")
INDEXED: Yes, Stored, Non Nulls
USE: Statistical calculations and range queries
```

CORRECT Optimized Queries (on CMP for business data)

```
sql
-- CORRECT Daily Phone Call Dashboard
ExecuteSQL(
   "SELECT
      idx_Outcome as Outcome,
      COUNT(*) as Count,
      AVG(idx_Duration) as AvgDuration
    FROM CMP
    WHERE ProcessType = 'PHO'
     AND IsTemplate = FALSE
     AND DATE(CreatedAt) = CURDATE()
    GROUP BY idx_Outcome";
   ""; ""
)
```

7.3 Interfaces: Dynamic vs Specific (80/20 Strategy)

Universal Dynamic Layout - CORRECT

```
filemaker
LAYOUT: "CMP_ETY_Form_Dynamic"
LOGIC: Reads attributes from CMP template and generates runtime interface

SCRIPT OnLayoutEnter:
```

```
# INPUT: $ProcessType (e.g. "PHO", "BOM")

# STEP 1: Read template from CMP
Go to Layout ["CMP"]
Enter Find Mode []
Set Field [CMP::ProcessType; $ProcessType]
Set Field [CMP::CmpType; "PROCESS"]
Set Field [CMP::IsTemplate; TRUE]
Perform Find []

# STEP 2: Generate fields for business data (from CMP instance)
# STEP 3: Generate controls for workflow (from ETY orchestration)
# STEP 4: Synchronize CMP ↔ ETY during editing
```

CHAPTER 8: CONCLUSIONS AND IMPLEMENTATION ROADMAP

8.1 Achieved Ontological Validation

The Concept of Proof 1P=1E=1C concretely demonstrates that:

1. **3P3 Ontology is Implementable:** The PEOPLE-PROCESSES-PRODUCTS trinity manifests perfectly in CMP-ETY-LOG architecture with correct separation between business data (CMP) and orchestration (ETY)
2. **1-1-1 Relationship is Maintainable:** Every action generates instance in CMP + orchestration in ETY + trace in LOG
3. **The System is Scalable:** From simple process (phone call) to complex process (BOM) using identical ontological principles
4. **Performance is Sustainable:** 10,000 test records demonstrate that JSON in CMP handles complexity without degradation
5. **Integration is Possible:** Coexistence with legacy systems and gradual migration is feasible

8.2 Fundamental Implementation Principles

GOLDEN RULE OF 3P3 SYSTEM:

CMP = EVERYTHING (Templates with IsTemplate=TRUE + Instances with IsTemplate=FALSE)
ETY = ONLY Orchestration and Workflow (NEVER business data)
LOG = ONLY Traceability and Audit

UNIVERSAL PATTERN FOR EVERY PROCESS:

1. DEFINITION → CMP template (IsTemplate=TRUE)
2. INSTANTIATION → CMP instance (IsTemplate=FALSE) with real data
3. ORCHESTRATION → ETY manages instance workflow
4. TRACEABILITY → LOG records every action

8.3 Detailed Implementation Roadmap

MILESTONE 1: Validated Proof of Concept (Step 1-2)

CYRIL DELIVERABLES:
- ├── FileMaker database with 3 functional tables
- ├── Correct Create_Universal_Entity script (CMP+ETY+LOG)
- ├── Operational CMP_Form_Phone layout
- ├── Test 1,000 phone calls in CMP (not ETY!)
- ├── Performance and data integrity report
- └── Live demo with clear CMP/ETY separation

8.4 Final Notes for Implementers

CRITICAL ATTENTION:

This document version 2.0 corrects a fundamental ontological error from v1.0 where instances were erroneously created in ETY instead of CMP.

ALWAYS Follow the pattern:

- **Business data** → CMP::JSONStructure (IsTemplate=FALSE)
- **Orchestration** → ETY::JSONData
- **Traceability** → LOG

Validation test for every implementation:

1. Is the instance with real data in CMP?
2. Does ETY contain only orchestration?
3. Do business queries interrogate CMP?
4. Are workflows managed in ETY?

APPENDIX A: CORRECT Quick Reference

CORRECT Instance Creation Pattern

```
sql
-- 1. Create instance in CMP
INSERT INTO CMP (CmpID, CmpType, IsTemplate, JSONStructure)
VALUES ('PHO25001', 'PROCESS_INSTANCE', FALSE, '{business_data}');

-- 2. Create orchestration in ETY
INSERT INTO ETY (EtyID, LinkedCMP, JSONData)
VALUES ('PHO25001_ORCH', 'PHO25001', '{orchestration}');

-- 3. Trace in LOG
INSERT INTO LOG (TargetTable, TargetID, ActionType)
VALUES ('CMP', 'PHO25001', 'CREATE');
```

CORRECT Business Data Query

```
sql
-- ALWAYS from CMP for business data
SELECT JSONStructure FROM CMP
WHERE IsTemplate = FALSE AND ProcessType = ?;

-- NEVER from ETY for business data!
```

CORRECT Workflow Query

```
sql
-- From ETY for status and orchestration
SELECT Status, JSONData FROM ETY
```

```
WHERE LinkedCMP = ?;
```

FINAL CONCLUSION

This document v2.0 represents the **ontologically correct version** of the 3P3 system, with the fundamental separation between:

- **CMP:** Universal repository of everything (templates and instances with data)
- **ETY:** Pure orchestration and workflow
- **LOG:** Pure traceability

By implementing EXACTLY this pattern, the 3P3 system will demonstrate its ontological and practical superiority over traditional fragmented systems.

For the happiness of all beings and for the harmonious development of a peaceful and conscious society

Luca Meggiolaro