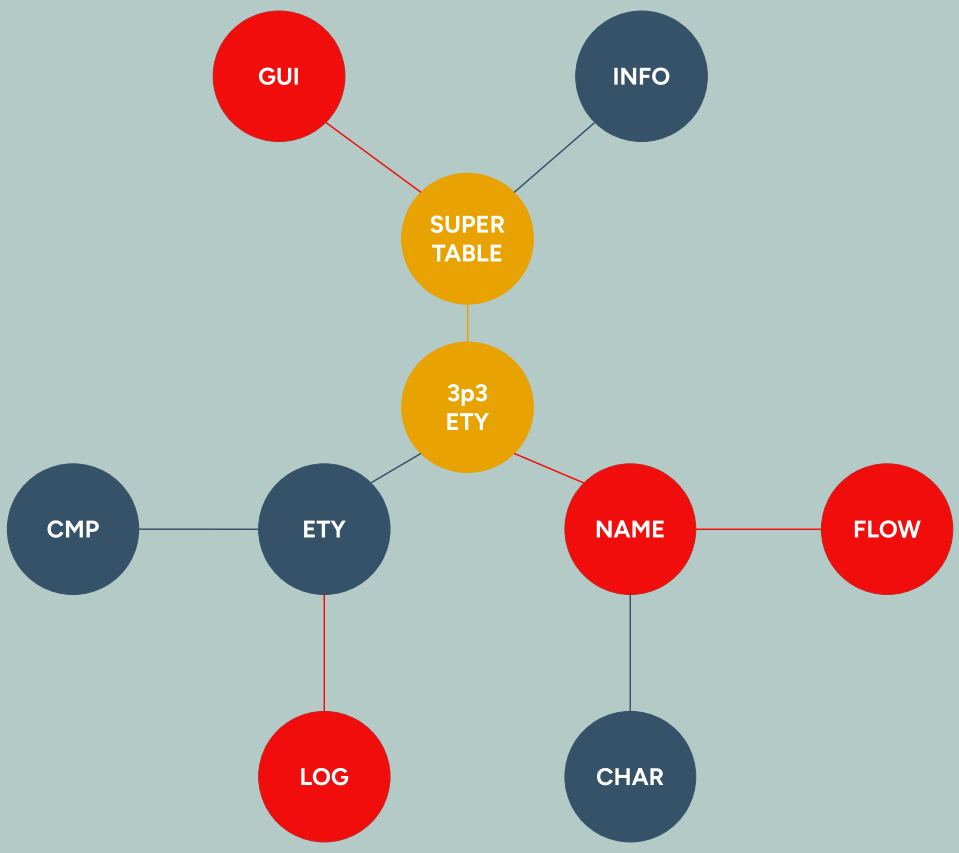
UK

THE BRIDGE - $^{3}p^{3}$ ALGORITHM

The Ontology
Where Person|Process|Product
become ONE



KOOLTOOL

Document Information:

Document's name	20251005 THE BRIDGE - 3p3 ALGORITHM UK ----
Start Date	03/10/25
End Date	07/10/25
version	03
Responsible	Luca Meggiolaro
Language	english
Recipients	technicians and system lovers

Table of Contents

1 3P3 ALGORITHM - THE COMPLETE ENTITY ONTOLOGY

Theoretical-Practical Document for THE BRIDGE
Version 3.0 - October 05, 2025
KOOL TOOL SRL - The Revolution from Dualism to Unity

1.1 READING GUIDE

This document accompanies the **Full Technical Specification** by Cyril & Osbert, providing the ontological foundation of the 3P3 system.

Objective: To allow a non-technical person to understand how THE BRIDGE works, and a developer to understand WHY it works this way.

- Theory Ontological - The deep "why"
- Practical Implementation - The concrete "how"
- Real Examples - Connections to KOOL TOOL

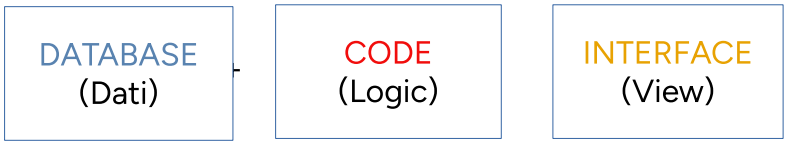
Balance: 50% accessible theory + 50% practical examples

1.2 INTRODUCTION: FROM DUALISM TO UNITY

The Problem of Fragmented Vision

We are normally accustomed to seeing reality **divided into separate components**:

TRADITIONAL VISION (DUALISTIC):



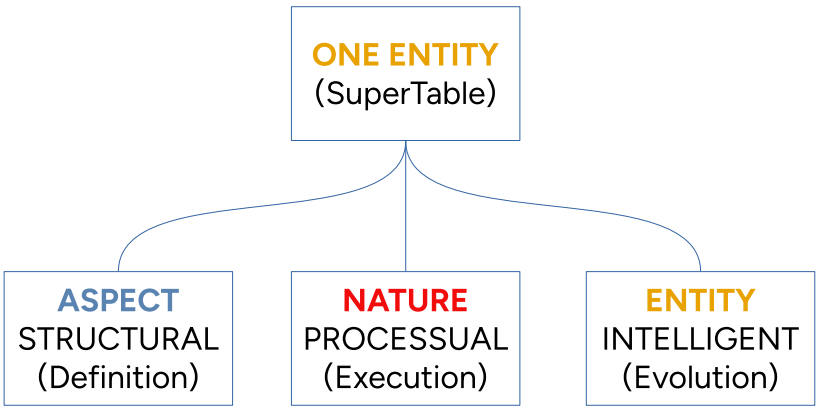
Three separate things that must be integrated

Problem: When something changes, you must modify 3 different places. Expensive maintenance, frequent errors, obsolete documentation.

The 3P3 Revolution: The Entity is ONE

When we think of an **entity** in the 3P3 system, we must think of a **unique being** that simultaneously manifests different aspects:

3P3 VISION (UNITARY):



Concrete Example: The Product PRD

When we create the digital entity "Product PRD" we are defining a **complex system**, not a simple record:

PRD (Product) = FILE composed of interconnected entities:

PRD_INFO (Main TUPLE)
- Product code
- Commercial name
- Base price
- Weight and dimensions
- Product category

BOM (Bill of Materials - TUPLE)
- List of necessary components
- Quantity per unit
- Supplier codes
- Raw material unit costs

TDS (Technical Data Sheet - TUPLE)
- Complete technical specifications
- Quality parameters
- Allowed tolerances
- Test methods

MDC (Margin Calculation - TUPLE)
├── Margin calculation
├── Strategic pricing
├── Competitor comparison
└── Target profit

MAD (Drawings List - TUPLE)
├── Technical drawings for each component
├── Versions and revisions
└── Technical approvals

PHO (Photos - TUPLE)
├── Finished product photos
├── Color variant photos
└── Marketing images

T**he PRD entity is ONE** - these are not "6 separate databases" but **6 perspectives of the same product**.

Once the product is defined through this complex structure, it becomes the basis for subsequent processes of Raw Material Purchasing → Production → Customer Delivery.

2 COMPLETE ONTOLOGY - The 10×10×10×3 Synthesis

2.1 The Formula of Existence

The 3P3 system discovers that **every entity** possesses a complete ontological structure:

10 FACTORS (base attributes: caller_name, phone_number...)
  ×
10 DOMAINS (meta-attributes: identity, temporal, trigger...)
  = 100 CAPACITY MATRIX per attribute
  ×
10 WORLDS (vital states: birth → optimization)
  = 1000 possible VITAL STATES
  ×
3 REALMS (existential tables: CMP-ETY-LOG)
  = 3000 SIMULTANEOUS MANIFESTATIONS

EVERYTHING IS ENTITY

2.2 The Power of the 10×10 Matrix

This is the hidden revolution of 3P3.

In the traditional system:
- Attribute = simple data field
- `caller_name VARCHAR(100)` → just a container

In the 3P3 system:
- Attribute = **intelligent entity with 10 dimensions**
- `caller_name` possesses 10 simultaneous capabilities:

```
javascript
caller_name: {
  1_IDENTITY: {
    searchable: true,
    unique: false,
    display_label: "Nome Chiamante"
  },
  2_TEMPORAL: {
    track_changes: true,
    version_control: true
  },
  3_AUTHORIZATION: {
    read_roles: ["SALES", "ADMIN"],
    write_roles: ["SALES"]
  },
```

```
 4_COMMUNICATION: {
  exportable: true,
  api_accessible: true,
  appears_in_email: true
 },
 5_TRIGGER: {
  on_change: "update_customer_record",
  validation: "min_3_chars"
 },
 6_DOCUMENTALE: {
  pdf_position: "header",
  font_size: 14,
  iso9001_procedure: "PR-SALES-001"
 },
 7_MATERIAL: {
  data_type: "TEXT",
  max_length: 100,
  required: true
 },
 8_PERFORMANCE: {
  track_usage: true,
  measure_fill_rate: true
 },
 9_SECURITY: {
  encrypt: false,
  audit_trail: true,
  gdpr_sensitive: true
 },
 10_EVOLUTION: {
  learn_patterns: true,
  suggest_autocomplete: true
 }
}
```

Result: ONE single attribute possesses 10 operational intelligences!

2.3 Practical Example: Phone Call PHO

PHO process with 5 attributes:

```
caller_name   → 10 domains = 10 capabilities
phone_number  → 10 domains = 10 capabilities
duration      → 10 domains = 10 capabilities
outcome       → 10 domains = 10 capabilities
notes         → 10 domains = 10 capabilities
──────────────────────────────────────────
TOTAL: 50 active intelligent capabilities
```

In traditional database: 5 dumb fields
In 3P3: 50 micro-systems collaborating

2.4 The Three Realms of Existence

The entity manifests through **3 universal tables**:

```
CMP (Components)
"Everything that CAN exist"
├─ Template PHO (definizione)
└─ Istanze PHO250001, PHO250002...
   (dati business reali)
```

```
ETY (Entity)
"Everything that EXISTS now"
├── Stati workflow correnti
├── Relazioni attive
└── Orchestrazione (NO dati business)
```

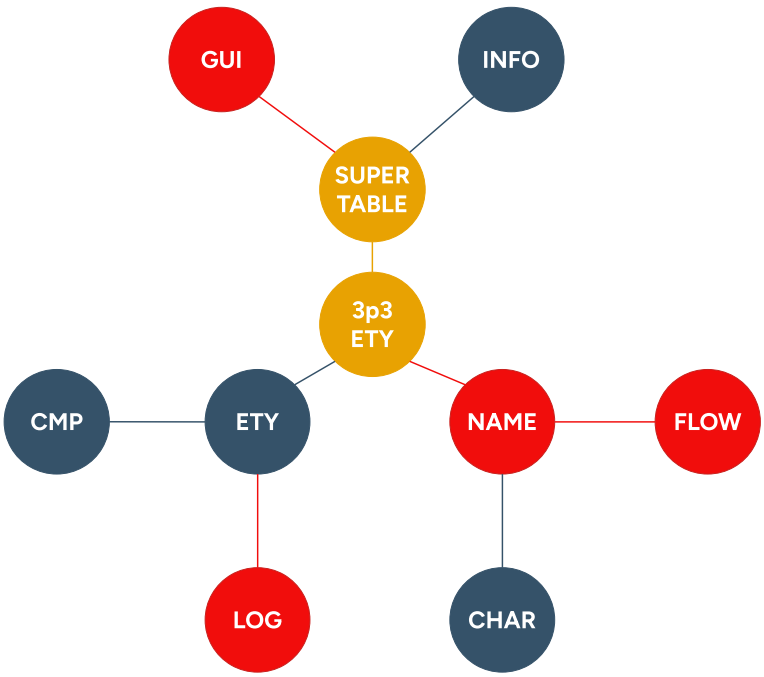
```
ETY (Entity)
"TEverything that HAS HAPPENED"
├── Stati workflow correnti
├── Relazioni attive
└── Orchestrazione (NO dati business)
```

Fundamental Principle:
The entity is ONE but lives simultaneously in 3 realms. Like a person who has body (CMP), mind (ETY) and memory (LOG).

3 VISUAL ONTOLOGY AND OPERATIONAL CYCLE

3.1 THE GEOMETRIC MAP OF THE ENTITY

Ontology As Visual Architecture "3p3 MAP"



This map is not decorative: it is the **architectural blueprint** of the complete 3P3 entity. Just as an industrial technical drawing shows sections and relationships between components, this visualizes the ontological structure of every digital entity.

The Ontological Color Code

Each color identifies a specific ontological nature:

🔴 **RED = Nature/Action/Manifestation**

NAME: I define who I am and where I am
FLOW: I define how I behave (process)
GUI: I manifest visually (interface)
LOG: I record every action (active memory)
Feature: Dynamic elements that change state and trigger events.

🔵 **BLU = Aspect/Structure/Containment**

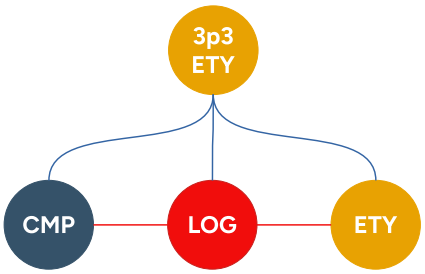
CMP: I contain the potential (templates + components)
ETY: I contain the present state (current entity)
CHAR: I contain the attributes (characteristics)
INFO: I contain elaborated information (wisdom)
Characteristic: containers that define "what it is" rather than "what it does".

🟡 **YELLOW = Entity/Unity/Binding**

3p3 ETY: The central orchestrator (beating heart)
SUPERTABLE: The supreme aggregate intelligence (mind)
Characteristic: unifying elements that hold the entire system together.

Geometry as Ontology

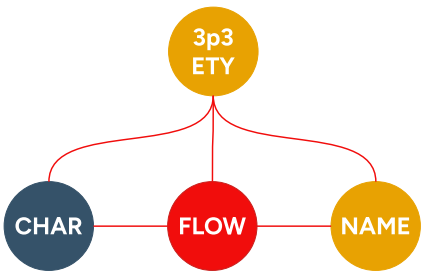
The spatial arrangement represents the existential hierarchy:



Lower Triangle - THE 3 REALMS:

- • CMP: Realm of Potential (what can exist)
- • ETY: Realm of Manifestation (what exists now)
- • LOG: Realm of Memory (what has happened)

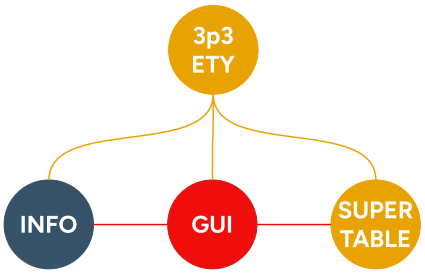
Operational meaning: The 3 fundamental databases of existence.



Right Triangle - The 3 Aspects:

- **NAME**: • Who am I and where am I (identity + position)
- • CHAR: What it contains (structure + attributes)
- • FLOW: How it behaves (process + workflow)

Operational meaning: the 3 definers that characterize every Process Manager.



Higher Vertex – INTELLIGENCE:

- • GUI: What it looks like (dynamic interface)
- • INFO: What it knows (processed information)
- • SUPERTABLE: The aggregated totality (complete JSON

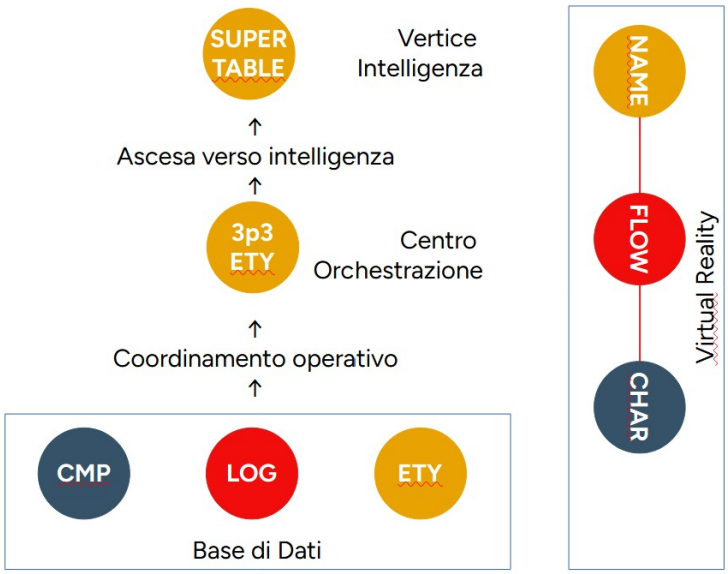
Operational meaning: The 3 elements of emergent intelligence.

-

The Center as a Beating Heart

The yellow circle "**3p3 ETY**" is the focal point:

- Connected to EVERYTHING (6 direct connections)
- Point of balance between foundation and intelligence
- Orchestrator who coordinates the entity's entire existence



Just as the spindle of a machine tool coordinates all movements, 3p3 ETY orchestrates all system operations.

3.2 THE 3P3 ONTOLOGICAL WORK CYCLE

From the Productive Cycle to the Ontological Cycle

Just as an industrial manufacturing process describes each operation required to create a semi-finished product, the 3P3 ontological cycle describes each step needed to create and manage a digital entity.

It is the "manufacturing manual" of the digital entity.

The 3-Level Hierarchy of Operations “3p3 CYCLE”

	LEVEL	GROUP	SEQUENCE	STRUCTURE	CODE	ENTITY ASPECT	USER ACTIONS	SYSTEM ACTIONS
[3P3]	1			1	[3P3]	3P3 ENTITY		
DATABASE	1	1		1.1	ETY	ETY_TABLE		
	1	1	1	1.1.1	ETY25001			The system creates a row in ETY
	1	2		1.2	CMP	CMP_TABLE		
	1	2	1	1.2.1	CMP25001			The system creates a row in CMP
	1	3		1.3	LOG	LOG_TABLE		
	1	3	1	1.3.1	LOG25001			The system creates a row in LOG
PROCESS MANAGER	1	2		2.1	NAME	NAME_PROCESS_MANAGER		
	1	2	1	2.1.1	NAME25001	NAME - Instantiation	User clicks the button "+" to instantiate a new ENTITY PROCESS	System calls interface NAME
	1	2	2	2.1.2	NAME25002	NAME - Definition	The user fills in the information in the template and clicks save	System creates CMP-LOG-ETY
	1	2	3	2.1.3	NAME25003	NAME - Visualization		The system shows the instance in the tree view NAME Process Manager
	1	2		2.2	CHAR	CHAR_PROCESS_MANAGER		
	1	2	1	2.2.1	CHAR25001	CHAR - Instantiation	User clicks the button "+" to instantiate new ATTRIBUTES	System calls interface CHAR
	1	2	2	2.2.2	CHAR25002	CHAR - Definition	The user fills in the information in the template and clicks save	System creates CMP-LOG-ETY
	1	2	3	2.2.3	CHAR25003	CHAR - Visualization		The system shows the instance in interface CHAR Process Manager
	1	2		2.3	FLOW	FLOW_PROCESS_MANAGER		
	1	2	1	2.3.1	FLOW25001	FLOW - Instantiation	User clicks the button for modelling FLOW	System calls the template FLOW
	1	2	2	2.3.2	FLOW25002	FLOW - Definition	The user fills in the information in the templates and clicks save	System creates CMP-LOG-ETY
	1	2	3	2.3.3	FLOW25003	FLOW - Visualization		The system shows the instance in interface FLOW Process Manager
INSTANCE MANAGER	1	3		3.1	GUI	GUI_INSTANCES_MANAGER		
	1	3	1	3.1.1	GUI25001	GUI - Generation	User clicks the button for calling the interface PHO	System calls the template PHO25001
	1	3		3.2	INFO	INFO_INSTANCES_MANAGER		
	1	3	1	3.2.1	INFO25001	INFO - Persistence	The user fills in the information in the template and clicks save	The system saves INFO in CMP-LOG-ETY
	1	3		3.3	SUPER	SUPERTABLE_INSTANCES_MANAGER		
	1	3	1	3.3.1	SUPER25001	SUPER - Intelligence		The system creates the JSON script

The system is developed on **3 ontological levels** corresponding to the 3 architectural layers:

LEVEL 1: DATABASE - Existential Foundations

1.1 ETY_TABLE → Current state of the entity
1.2 CMP_TABLE → Potential (templates and components)
1.3 LOG_TABLE → Memory (action log)

Function: Fundamental databases (Storage Layer). Every higher-level operation is reflected here.

LEVEL 2: PROCESS MANAGER - Operational Definition

2.1 NAME → Identification and instantiation
2.2 CHAR → Attributes and characteristics
2.3 FLOW → Behavior and workflow

Function: Operational definers (Business Logic Layer). The entity acquires identity, structure, and behavior.

LEVEL 3: INSTANCE MANAGER - Intelligence and Interface

3.1 GUI → Dynamic user interface
3.2 INFO → Processed information
3.3 SUPERTABLE → Intelligent aggregation (JSON)

Function: Intelligence elements (Presentation + Intelligence Layer). The entity becomes "knowledgeable."

The STRUCTURE Coordinate System

Each element has a unique hierarchical address:

FORMAT: X.Y.Z
X = LEVEL (1=Foundation, 2=Definition, 3=Intelligence)
Y = GROUP (which element within the level)
Z = SEQUENCE (which action within the group)

EXAMPLES:
1.1.1 = ETY25001 (first element in ETY table, level 1)
2.1.2 = NAME25002 (second action in NAME, level 2)
3.3.1 = SUPER25001 (JSON script in SUPERTABLE, level 3)

Operational benefits:

- Complete traceability of each element
- Hierarchical navigation through levels
- Explicit relationships between components
- Guaranteed uniqueness (no collisions)

Universal Workflow Pattern

Each level follows the same 5-step pattern:

1. INSTANTIATION → User clicks "+" → System calls template
2. DEFINITION → User fills data → System validates
3. PERSISTENCE → System creates CMP-LOG-ETY
4. VISUALIZATION → System shows result in interface
5. INTELLIGENCE UPDATE → System updates JSON SUPERTABLE

This is the "DNA of the cycle": just as biological DNA uses 4 bases to encode infinite proteins, 3P3 uses this pattern to manage infinite entities.

Operational Example: Process Manager Creation

PHASE 1 - Foundation (automatic):

System creates → ETY25001 (empty, ready)
System creates → CMP25001 (template ready)
System creates → LOG25001 (first entry)
→ The entity EXISTS in 3 dimensions

PHASE 2 - NAME definition:

User: clicks "+" instantiate new ENTITY PROCESS
System: calls interface NAME
User: fills "Production Order Manager" + PHO25001
System: creates CMP-LOG-ETY linkage
System: shows in tree view Production/Planning/Orders
→ The entity has IDENTITY and POSITION

STEP 3 - CHAR Definition:

User: defines attributes (OrderNumber, Quantity, DueDate, Status)
System: creates CMP with 4 attributes + validations
System: shows structure in CHAR Process Manager
→ The entity has STRUCTURE

STEP 4 - FLOW Definition:

User: workflow models (Draft → Active → Complete)
System: creates CMP with 3 states + 2 triggers
System: shows workflow in FLOW Process Manager
→ The entity has BEHAVIOR

STEP 5 - Intelligence (automatic):

System: generates dynamic GUI based on NAME-CHAR-FLOW
System: saves INFO in CMP-LOG-ETY
System: creates JSON script SUPER25001 (total aggregation)
→ The entity is COMPLETE, INTELLIGENT, OPERATIONAL

The Continuous Operating Loop

Once created, the entity enters a continuous loop:

```
USER ACTION
 ↓
SYSTEM PROCESSING
  ├─ Validates against CHAR
  ├── Executes FLOW logic
  ├── Updates ETY state
  └─ Triggers actions
 ↓
RESULT
  ├── Updates CMP if needed
  ├── Creates LOG entry (auto)
  ├── Updates INFO
  └── Refreshes GUI
 ↓
SUPERTABLE UPDATE
  └── Regenerates JSON
 ↓
[Loop repeats]
```

Characteristics:

- ✅ Self-documenting (automatic LOG)
- ✅ Self-evolving (INFO/SUPER updates)
- ✅ Traceable (permanent coordinates)
- ✅ Resilient (consistent CMP-ETY-LOG state)

3.3 CODIFICATION AND ONTOLOGICAL STRUCTURE

Triple Identification System

Each 3P3 entity has 3 complementary identifiers:

1. DNA (Globally Unique Identity)

Format: PRXYYNNNN

```
Esempio: PHO25001
         │└┘└──┘ NNNN: progressivo (0001-9999)
         │└──── YY: anno (25 = 2025)
         └───── PRX: process type (PHO, BOM, INV, etc.)

Characteristic: unique, permanent, immutable
Purpose: to UNIQUELY identify the entity in the system
```

2. STRUCTURE (Ontological Position)

Format: X.Y.Z

```
Esempio: 2.2.1
         │ │ └─ Z: sequence (which action)
         │ └─── Y: group (which element)
         └───── X: level (which ontological layer)

Characteristics: hierarchical, structural, editable
Purpose: Place the entity in the 3P3 ONTOLOGY
```

3. LOCATION (Organizational Position)

Format: breadcrumb path

```
Example: Production/Planning/Orders

Characteristics: Descriptive, user-friendly, organizational
Purpose: Position the entity within the company ORGANIZATION
```

Complete Entity System

```
ENTITY 3P3 - TRIPLE IDENTIFICATION

DNA: PHO25001
└─ Unique in the system

STRUCTURE: 2.2.1
└─ Process Manager > CHAR > Action 1

LOCATION: Production/Planning/Orders
└─ Organizational Position

URL: kooltool.ro/2.2.1/PHO25001
└─ Full Direct Access
```

Analogy with Google Drive

Google Drive uses the same dual system:

FILE ID (like DNA):

```
drive.google.com/file/d/1a2b3c4d5e6f7g8h
                        └──────┬──────┘
```

```
                        Permanent unique

PATH (come LOCATION):
MyDrive/Production/Planning/Orders/PHO25001.pdf
└─────┘ └──────────────────────────┘ └─────────┘
  ROOT      LOCATION PATH              FILE

If you move the file:
❌ File ID does NOT change (permanent)
✅ Path changes (new location)
```

Same in 3P3:

- DNA NEVER changes (PHO25001 permanent)
- STRUCTURE can change (ontological reorganization)
- LOCATION can change (corporate reorganization)

Building Relationships

Relationships are built in 3 ways:

1. Hierarchical (from STRUCTURE)

2.1 NAME_PROCESS_MANAGER (parent)
2.1.1 NAME25001 (child)
2.1.2 NAME25002 (child)
2.1.3 NAME25003 (child)

Type: structural, implicit from the tree

2. Operational (from DNA)

PHO25001 → INV25015 (order → inventory)
PHO25001 → SHP25023 (order → shipment)
PHO25001 → BOM25008 (order → BOM)

Type: operational, explicit in the data

3. Organizational (from LOCATION)

Production/Planning/Orders/
├── PHO25001
├── PHO25002
└── PHO25003

Type: Organizational, location-based

3.4 FROM THE ELEMENTARY BRICK TO THE SUPERTABLE

Fundamental Principle

💎 **All the 3P3 complexity already lives in the ATTRIBUTE (THE CELL).**

When you develop 1 attribute, you are creating:

- Its **identity** (NAME)
- Its **structure** (CHAR: type, validations, relationships)
- Its **behavior** (FLOW: triggers, actions, states)
- Its **interface** (GUI: how it looks)
- Its **intelligence** (INFO: metadata)

TUPLE and FILE are just compositions of the attribute.

Layered Architecture

CELL (Attribute)
↓ aggregation
TUPLE (Record/Row)
↓ aggregation
FILE (Table/Entity)
↓ aggregation
SUPERTABLE (Total Intelligence)

Concrete example:

```
CEL: OrderNumber
  ├─ NAME: "OrderNumber"
  ├─ CHAR: TEXT, required, unique, pattern
  ├─ FLOW: onCreate, onChange, states
  ├─ GUI: TextInput, placeholder, readonly
  └─ INFO: metadata, usage stats

TUPLA: Production Order #ORD-2025-001
  ├─ CELLA_1: OrderNumber = "ORD-2025-001"
  ├─ CELLA_2: Quantity = 500
  ├─ CELLA_3: DueDate = 2025-11-15
  └─ CELLA_4: Status = "Draft"

FILE: Production Order Manager [PHO25001]
  ├─ Schema: 4 attributi definiti
  ├─ Workflow: Draft → Active → Complete
  ├─ Instances: array di TUPLE
  └─ Intelligence: aggregations on all TUPLE

SUPERTABLE: Complete System
  ├─ Entities: [PHO, INV, SHP, BOM, ...]
  ├─ Global metrics: totali, health, trends
  └─ JSON completo: all the wisdom of the system
```

Emergent Logic

1. Define 1 **CELL** correctly
2. **TUPLE** emerges as an aggregation of cells
3. **FILE** emerges as a collection of tuples
4. **SUPERTABLE** emerges as a global synthesis

You're not programming 4 different things.
You're programming 1 thing (the CELL) and the rest scales naturally.
This is **ontological emergence** in action.

Essential Implementation Pattern

javascript

```javascript
// STEP 1: Define the CELL
const OrderNumberAttribute = {
  name: { fieldName: 'OrderNumber', ... },
  char: { type: 'TEXT', required: true, unique: true, ... },
  flow: { onCreate: ..., onChange: ..., states: ... },
  gui: { widget: 'TextInput', ... },
  info: { metadata: ..., stats: ... }
};

// STEP 2: Compose the TUPLE (attribute aggregation)
const ProductionOrderTuple = {
  attributes: [OrderNumberAttribute, QuantityAttribute, ...],
  flow: { /* coordina flow delle celle */ },
  gui: { /* layout delle celle */ }
};

// STEP 3: Create the FILE (tuple collection)
const ProductionOrderManager = {
  schema: ProductionOrderTuple,
  instances: [],  // array di tuple
  info: { /* aggregazioni */ }
};

// STEP 4: SUPERTABLE aggregates everything automatically
const SuperTable = {
  entities: [ProductionOrderManager, InventoryManager, ...],
  generateJSON: function() { /* sintesi globale */ }
};
```

3.5 WORKING SUMMARY FOR THE PROGRAMMER

Golden Rules

✅ Use the **MAP** to understand the overall architecture

✅ Follow the **CYCLE** to implement operations

✅ Start from the **CELL** to build from the bottom

Key Principle

> Correctly develop 1 CELL (attribute) with its 5 ontological dimensions (NAME-CHAR-FLOW-GUI-INFO).
>
> TUPLE, FILE, and SUPERTABLE are natural aggregations.
>
> **The system scales by itself.**

Three Keys to Reading

This section shows the 3P3 ontology from 3 perspectives:

1. Geometric **Map** → displays the structure (what it is)
2. Process **Cycle** → describes the process (how it works)
3. **Coding and Composition** → reveals the construction logic (how it's made)

These are not three different systems. They are three ways of looking at the same ontological reality.

All is One

A single 3P3 entity simultaneously contains:

- The 3 Existential Realms (Potential-Present-Memory)
- The 3 Defining Aspects (Identity-Structure-Behavior)
- The 3 Operational Levels (Foundations-Definition-Intelligence)
- The Unifying Center (3p3 ETY)
- The Supreme Synthesis (SUPERTABLE)

This is the 3P3 ontology.

"Just as an industrial production cycle describes each step to create a semi-finished product, the 3P3 ontological cycle describes each step to create a living digital entity."

4 DEFINE - Process Manager (States 1-3)

4.1 Overview

The Process Manager is where the entity is born and structured. It's not a "form builder" but an ontological tool for Times and Methods.

Revolutionary Principle:
Define Processes = Define the Organization

- Process present in the system = Existing business capability
- Process absent = Missing business capability

4.2 3-Panel Architecture

ONE ENTITY - THREE VIEWS

PANEL 1	PANEL 2	PANEL 3
NAME	CHAR	FLOW
Identity	Characteristics	Behaviour
Tree View	Attribute + Interface	Operations + Workflow

When I click on PHO, all 3 panels update because they are perspectives of the same entity.

4.3 STATE 1: NAME - Understanding What to Do (Ontological Times and Methods)

Ontological Meaning: The birth of the entity in the system as an organizational process

The Heart of the System: 3P3 Times and Methods

This state represents the **true essence of the company organization**. When a Time and Methods expert works with 3P3, he or she is building the company's operational ontology.

Fundamental Principle:
When I create a node in the tree, I am defining a ROW of the Process List.

Example from the KOOL TOOL Process List:

Main_Process	Process	Sub_Process	Structure	Description
TASK	PHONE CALL		1.1	Phone call management
TASK	PHONE CALL	FOLLOW UP	1.1.1	Schedule callback
SALES	OFFER		2.1	Offer generation
SALES	OFFER	APPROVAL	2.1.1	Manager approval
PRODUCTION	PRODUCT		3.1	Product definition
PRODUCTION	PRODUCT	BOM	3.1.1	Bill of materials
PRODUCTION	PRODUCT	TDS	3.1.2	Technical data sheet

IMPORTANT: This 3-level structure (Main→Process→Sub) is a simplification for the Excel sheet.

In reality, 3P3:
The structure is FLEXIBLE MULTILEVEL, based on real-world complexity:

KOOL TOOL (Level 0)
- SALES (Livello 1)
  - PROSPECTING (Livello 2)
    - PHONE CALLS (Livello 3)
      - COLD CALL (Livello 4)
      - WARM CALL (Livello 4)
      - FOLLOW UP (Livello 4)
        - SECOND ATTEMPT (Livello 5)
    - EMAIL CAMPAIGN (Livello 3)
  - OFFERS (Livello 2)
    - STANDARD OFFER (Livello 3)
    - CUSTOM OFFER (Livello 3)
- PRODUCTION (Livello 1)
  - PRODUCTS (Livello 2)
    - DEFINITION (Livello 3)
      - BOM (Livello 4)
      - TDS (Livello 4)
      - MDC (Livello 4)
    - MANUFACTURING (Livello 3)
  - QUALITY CONTROL (Livello 2)
- LOGISTICS (Livello 1)
  - DELIVERY (Livello 2)

Infinite possible levels!

User Action:
- Open Process Manager
- Click [+] in the hierarchy tree
- Enter "PHO" (Phone Call)
- Place the node under "SALES → PROSPECTING"

System Performs:

```javascript
// BIRTH IN CMP (Potential)
CMP.create_template({
entity_code: 'PHO',
entity_name: 'Phone Call',
entity_type: 'TUPLE',
parent: 'PROSPECTING',
structure_level: '1.2.1',
icon: '📞'
});

// BIRTH IN ETY (Orchestration)
ETY.create_orchestrator({
entity_code: 'PHO',
current_state: 'DEFINING',
workflow: 'to_be_defined',
hierarchy_path: 'KOOL_TOOL/SALES/PROSPECTING/PHO'
});

// BIRTH IN LOG (Memory)
LOG.record({
level: 'L1_PROCESS',
action: 'ENTITY_BORN',
entity: 'PHO',
structure: '1.2.1',
timestamp: NOW(),
user: 'luca.meggiolaro'
});
```

Visible Result: New node "PHO 📞" appears in the hierarchical tree under PROSPECTING

Critical Feature: Dynamic Reorganization

The true power of Process Manager is the ability to **reorganize flows while maintaining all the system logic**:

SCENARIO: Timing and Methods decides that PHONE CALL
must move from PROSPECTING to CUSTOMER SERVICE

ACTION: Drag & Drop PHO from one branch to another

SYSTEM MAINTAINS:
✓ All existing instances (PHO250001, PHO250002...)
✓ All workflows and triggers
✓ All relationships with other entities
✓ All history in the LOG
✓ All attributes and configurations

SYSTEM UPDATE:
✓ Hierarchical structure (1.2.1 → 2.3.1)
✓ Ontological path
✓ Responsibilities and permissions
✓ Reports and dashboards

This is the heart of the company organization: being able to modify processes to increasingly match the actual operational reality.

Note on Fractal Reality:
The entity changes meaning depending on the point of view. If I observe from the "cell" (single phone call PHO250042), I have a certain mission. If I observe from the "department" (all the team's phone calls), I have a different mission. If I observe from the "company" (all customer contacts), I have yet another mission. But it's always the SAME entity seen from different scales.

Future Vision: Natural Ontological Language

IIn the future, the system will evolve towards an ontological programming language:

USER WRITES IN NATURAL LANGUAGE:

"Kool Tool Management:
- Salespeople call customers
- Salespeople record requests
- Salespeople open projects
- Technicians develop products
- Manufacturers manufacture orders"

SYSTEM AUTOMATICALLY PROPOSES:

✓ "salespeople" → Create ROLE entity: Sales Team
✓ "call" → Create PHO process: Phone Call
✓ "customers" → Create CUSTOMER entity
✓ "record requests" → Create RCH process: Client Request
✓ "open projects" → Create PRJ process: Project
✓ "technicians" → Create ROLE entity: Technical Team
✓ "develop products" → Create PRD process: Product
✓ "manufacturers" → Create entity ROLE: Production Team
✓ "manufacture orders" → Create MFG process: Manufacturing

EACH WORD = ENTITY
EACH VERB = PROCESS
EACH SENTENCE = WORKFLOW

The system semantically recognizes roles, actions, and objects and translates them into an ontological structure.

4.4 STATE 2: CHAR - Define Characteristics

Ontological Meaning: The entity defines its structural form

User Action:
- I add 5 attributes to the CHARACTERISTICS panel
- For each attribute, I define the type and properties

System Performs:

```javascript
// For EACH attribute added
attributes.forEach(attr => {
  // Save definition with 10 DOMAINS
  CMP.add_attribute({
    entity: 'PHO',
    attribute_name: 'caller_name',
    attribute_config: {
      // MATRICE 10 DOMINI
      D1_IDENTITY: {
        searchable: true,
        unique: false,
        display_name: "Nome Chiamante"
      },
      D2_TEMPORAL: {
        track_changes: true,
```

```
  version_control: true
},
D3_AUTHORIZATION: {
  read: ['SALES', 'ADMIN'],
  write: ['SALES'],
  delete: ['ADMIN']
},
D4_COMMUNICATION: {
  exportable: true,
  api_accessible: true,
  email_include: true
},
D5_TRIGGER: {
  on_change: 'update_crm_customer',
  validation: 'min_3_chars',
  auto_actions: ['log_change']
},
D6_DOCUMENTALE: {
  pdf_include: true,
  pdf_position: 'header',
  font_size: 14,
  iso9001_field: 'PR-SALES-001.field_2'
},
D7_MATERIAL: {
  type: 'TEXT',
  max_length: 100,
  required: true,
  default: null
},
D8_PERFORMANCE: {
  track_usage: true,
  measure_fill_rate: true,
  kpi_target: 0.95
},
D9_SECURITY: {
  encrypt: false,
  audit_trail: true,
  gdpr_sensitive: true,
  retention_days: 2555
},
D10_EVOLUTION: {
  ai_suggest_autocomplete: true,
  learn_patterns: true,
  optimize_entry_time: true
}
```

```
    }
  });
});
```

Automatic Decision Interface:

```
```javascript
if (attributes.length <= 5 && no_sub_entities) {
  interface_type = 'FORM_SIMPLE';  // Form verticale
} else if (has_sub_entities) {
  interface_type = 'MULTI_TAB';    // Tab per ogni sub
} else {
  interface_type = 'TABLE_VIEW';   // Lista records
}
```

Tab Interface Preview: Shows you in real time how the form will look.

Example PHO (TUPLA):
Simple vertical form with 5 fields:
- Caller Name (text)
- Phone Number (tel)
- Duration (number, auto-calculated)
- Outcome (select: INTERESTED/NOT_INTERESTED/CALLBACK)
- Notes (textarea)

Example PRD (complex FILE):
Interface multi-tab:
Tab 1: PRD Info (main data)
Tab 2: BOM (bill of materials)
Tab 3: TDS (technical data sheet)
Tab 4: MDC (margin calculation)
Tab 5: MAD (drawings list)
Tab 6: PHO (product photos)

4.5 STATE 3: FLOW - Who Does What When

Ontological Meaning: The entity learns how to behave

User Action:
- I define basic operations in the FLOW (Entity Flow) panel
- I specify workflow states and transitions
```

IMPORTANT: This panel is called FLOW (not "Components") to avoid confusion with the CMP table.

ForTUPLE (PHO):

```javascript
entity_flow: {
  elementary_operations: [
    "01.1 User opens phone call form",
    "01.2 User enters caller information",
    "01.3 System validates phone format",
    "01.4 User records conversation notes",
    "01.5 User selects outcome",
    "01.6 IF outcome=INTERESTED → Trigger create OFC"
  ],
  workflow_states: {
    sequence: "NEW → IN_PROGRESS → COMPLETED",
    alternative: "NEW → CANCELLED"
  },
  triggers: [
    {
      condition: "outcome == 'INTERESTED'",
      action: "create_offer_entity()",
      creates: "OFC",
      auto_populate: ["caller_name", "phone_number"]
    },
    {
      condition: "duration > 30",
      action: "alert_manager()",
      notification: "EMAIL",
      recipient: "sales.manager@kooltool.ro"
    }
  ]
}
```

For FILE (PRD with sub-entities):

```javascript
entity_flow: {
  sub_entities: ['BOM', 'TDS', 'MDC', 'MAD', 'PHO'],

  orchestration_rules: {
    'BOM.completed': 'enables_MDC',
    'MDC.approved': 'enables_OFC',
```

```
  'TDS.validated': 'enables_production'
 },

 inter_entity_triggers: [
  {
   from: 'BOM',
   to: 'MDC',
   condition: 'total_cost_calculated',
   action: 'populate_base_cost_in_MDC'
  },
  {
   from: 'MDC',
   to: 'OFC',
   condition: 'margin_approved',
   action: 'create_offer_with_final_price'
  }
 ],

 completion_criteria: {
  required: ['BOM.complete', 'TDS.complete', 'MDC.approved'],
  optional: ['MAD.complete', 'PHO.complete']
 }
}
```

Result after States 1-3:

The entity is fully defined at the ontological level:

- PHO (TUPLA): Ready-made template for single instances
- PRD (FILE): Complex template with 6 orchestrated subentities

But it's not yet "alive";it's a potential ready to manifest through the Instance Manager.

5 LIVING - Instance Manager (States 4-6)

5.1 Overview

The Instance Manager is where the entity lives through real data. It goes from template (possibility) to instance (reality).

5.2 STATE 4: GUI - Start Activity

Ontological Meaning: From Potential to Concrete Existence

User Action:
- Open Instance Manager
- Click "New PHO"
- The system generates unique DNA

System Performs:

```javascript
// DNA GENERATION (Anti-collision)
const dna = generateDNA('PHO'); // → PHO250042

// CLONE TEMPLATE → INSTANCE in CMP
CMP.create_instance({
instance_dna: 'PHO250042',
from_template: 'PHO_TEMPLATE',
initial_values: {},
status: 'ACTIVE',
created: NOW(),
creator: 'mario.rossi'
});

// ACTIVATE ORCHESTRATION in ETY
ETY.activate_instance({
instance_dna: 'PHO250042',
workflow_state: 'NEW',
relations: [],
parent: null
});

// REGISTER BIRTH in LOG
LOG.record({
level: 'L1_PROCESS',
action: 'INSTANCE_CREATED',
DNA: 'PHO250042',
```

```
user: 'mario.rossi',
timestamp: NOW()
});
```

Automatically Generated GUI:

Based on the definition in the Process Manager, the system generates the interface:

For PHO (TUPLA simple):

```
```html
<!-- Form auto-generate -->
<form id="PHO250042">
  <input name="caller_name" type="text" required />
  <input name="phone_number" type="tel" required />
  <input name="duration" type="number" auto-calc />
  <select name="outcome" trigger-enabled>
    <option>INTERESTED</option>
    <option>NOT_INTERESTED</option>
    <option>CALLBACK</option>
  </select>
  <textarea name="notes"></textarea>
  <button>Save Phone Call</button>
</form>
```
```

For PRD (FILE multi-tab):

```
```html
<!-- Interface multi-tab auto-generate -->
<div class="product-manager">
  <div class="tabs">
    <button class="active">PRD Info</button>
    <button>BOM</button>
    <button>TDS</button>
    <button>MDC</button>
    <button>MAD</button>
    <button>PHO</button>
  </div>

  <div class="tab-content">
    <!-- Tab 1: PRD Info -->
    <div class="prd-info active">
      <input name="product_code" required />
      <input name="product_name" required />
```
```

```
    <input name="price" type="currency" />
  </div>

  <!-- Tab 2: BOM (table view) -->
  <div class="bom-table">
    <table>
      <tr><th>Material</th><th>Qty</th><th>Cost</th></tr>
      <!-- Dynamic rows -->
    </table>
  </div>

  <!-- Altri tab... -->
 </div>
</div>
```

5.3 STATUS 5: INFO - Check Activity

Ontological Meaning: The entity populates itself with information

User Action
- I fill out the form during the call
- I progressively save the data

System Performs (for each field):

```javascript
// SAVE TO CMP (Business Data)
CMP.update_instance({
dna: 'PHO250042',
field: 'caller_name',
value: 'Mario Rossi'
});

// UPDATE STATUS IN ETY
ETY.update_state({
dna: 'PHO250042',
workflow_state: 'IN_PROGRESS',
progress: 0.6 // 3/5 fields filled
});

// TRACE IN LOG (L3_ATOMIC)
LOG.record({
level: 'L3_ATOMIC',
action: 'FIELD_UPDATED',
DNA: 'PHO250042',
```

```
field: 'caller_name',
old_value: null,
new_value: 'Mario Rossi',
timestamp: NOW()
});
```

Final Data in CMP:

````
```json
{
  "instance_dna": "PHO250042",
  "data": {
    "caller_name": "Mario Rossi",
    "phone_number": "+39 02 1234567",
    "duration": 12,
    "outcome": "INTERESTED",
    "notes": "Cliente interessato a catalogo premium hair extensions 60cm"
  },
  "metadata": {
    "created": "2025-10-04T10:30:00Z",
    "modified": "2025-10-04T10:42:00Z",
    "creator": "mario.rossi",
    "k_coefficient": 2.1
  }
}
```
````

5.4 STATE 6: DATABASE - Archive and Learn

Ontological Meaning: Experience becomes memory and knowledge

User Action:
- Click "Complete Call"
- System performs final save

System Performs:

````
```javascript
// COMPLETION IN ETY
ETY.complete_workflow({
DNA: 'PHO250042',
workflow_state: 'COMPLETED',
completion_time: NOW(),
duration_total: 12
````
````

```
});

// EXECUTE AUTOMATIC TRIGGER
if (data.outcome === 'INTERESTED') {
// Automatically create offer
const offer_dna = create_entity('OFC');

// Pre-populate with phone call data
CMP.populate_from_parent({
new_dna: offer_dna,
parent_dna: 'PHO250042',
copy_fields: ['caller_name', 'phone_number']
});

// Establishes relationship
ETY.create_relation({
from: 'PHO250042',
to: offer_dna,
type: 'GENERATES',
timestamp: NOW()
});

// Notify manager
send_notification({
to: 'sales.manager@kooltool.ro',
subject: 'New lead from phone call',
body: `Mario Rossi is interested - Offer ${offer_dna} created automatically`
});
}

// "LESSONS LEARNED" EXTRACTION
const lessons = extract_insights({
successful_pattern: 'calls_12min_outcome_interested',
best_practice: 'direct_approach_works',
improvement: 'reduce_intro_save_2min',
kpi_impact: 'conversion_rate_+5%'
});

// SAVE INSIGHTS IN LOG
LOG.record({
level: 'L1_PROCESS',
action: 'LEARNING_CAPTURED',
insights: lessons,
will_improve: 'future_PHO_templates'
});
```

```
```

Importance of Intelligent Storage:
It's not just "saving data" but extracting patterns, measuring performance, and learning what works to improve future cycles. The database becomes living organizational memory.

Calculating the K Coefficient:

```javascript
K = Extracycles / (Efficiency × Presenteeism)

// For this call
Extracycles = 2 minutes (time wasted on long intro)
Efficiency = 0.95 (effective call)
Presenteeism = 1.0 (user fully focused)

K = 2 / (0.95 × 1.0) = 2.1

// K > 2.0 indicates inefficiency to be corrected
// System suggests: "Reduce intro from 4 minutes to 2 minutes"
// Implementation: Update PHO template with optimized intro script
```

6 EVOLVE - SuperTable (States 7-10)

6.1 Overview

The SuperTable is the entity in its entirety;the aggregate view that contains EVERYTHING and enables higher intelligence.

Fundamental Principle:
SuperTable = THE ONE ENTITY that manifests its 3,000 simultaneous possibilities.

6.2 STATE 7: FILTER - Search

Ontological Meaning: Acquiring data from all sources

User Action:
- Open SuperTable view "All Phone Calls"
- Apply ontological filters

Traditional vs. Ontological Filters:

```sql
-- TRADITIONAL (syntactic)
SELECT * FROM calls
WHERE customer_name = 'Mario Rossi'
AND date > '2025-10-01';

-- ONTOLOGICAL 3P3 (semantic)
FILTER_ENTITY({
entity: 'Mario Rossi',
relation_type: 'ALL_COMMUNICATIONS',
timeframe: 'last_month',
include_related: true
});
```

-
- Automatically returns:
✓ Phone calls with Mario
✓ Related emails
✓ Offers sent (Official Communications)
✓ Orders received (Orders received)
✓ Current projects (Privacy Policy)
✓ Shared documents (Doc)
✓ Payments received (Payment Policy)

SuperTable Tripartite Query:

```javascript
// SuperTable queries the 3 KINGDOMS simultaneously
const phone_calls_view =
CMP.get_instances('PHO*') // Business data
+ ETY.get_states('PHO*') // Workflow states
+ LOG.get_history('PHO*'); // Full History

// Result: Unified View
[
{
dna: 'PHO250042',
caller: 'Mario Rossi', // from CMP
phone: '+39 02 1234567', // from CMP
status: 'COMPLETED', // from ETY
duration: 12, // from CMP
outcome: 'INTERESTED', // from CMP
created_offer: 'OFC250015', // from ETY relations
k_coefficient: 2.1, // from LOG analytics
last_modified: '10:42', // from LOG
modified_by: 'mario.rossi' // from LOG
},
// ... 846 more calls
]
```

6.3 STATE 8: PIVOT - Understanding (Infinite Structured Views)

Ontological Meaning: Analyzing and Transforming Data into Knowledge

The Power of the Ontological Pivot

Imagine Google Sheets Pivot Table applied to the SuperTable with ontological capabilities:

Key Concept:

Each user can reclassify information according to what they deem most important, save the view, and generate infinite customized reports.

Practical Example 1: Phone Calls for Outcome

```javascript
pivot_config_1 = {
name: "Conversion Analysis by User",

// Pivot configuration
rows: ['assigned_user'],
```

```
columns: ['outcome'],
values: {
field: 'count',
aggregation: 'COUNT'
},
filters: {
date_range: 'last_30_days',
team: 'SALES'
},

// View generated
  result: `
```

User	INTERESTED	NOT_INT	CALLBACK	Total
Mario Rossi	48	89	15	152
Laura Bianchi	51	125	22	198
Total	99	214	37	350

```
  `,
// Save this view
save_as: "Monthly_Conversion_Report",

// Generate automatically
auto_generate: {
format: 'PDF',
schedule: 'monthly',
recipients: ['sales.manager@kooltool.ro'],
iso9001_document: 'REP-SALES-003'
}
}
```

Practical Example 2: Products by Marginality

```javascript
pivot_config_2 = {
  name: "Product Profitability Analysis",

  rows: ['product_category', 'product_code'],
  columns: ['month'],
  values: {
    field: 'margin_percentage',
    aggregation: 'AVERAGE'
  },
```

```
 filters: {
   year: 2025,
   margin_min: 0.20
 },

 result: `
```

Category / Product	Jan	Feb	Mar	Avg
Premium Extensions				
PRD78	35%	38%	42%	38%
PRD92	28%	31%	29%	29%
Standard				
PRD45	22%	23%	21%	22%

```
 `,

 save_as: "Profitability_Tracking",
 dashboard_widget: true
}
```

Specialized Attributes for Dynamic Views:

The same entities can appear in different views based on specialized attributes:

```
```javascript
view_attributes = {
 // KANBAN VIEW
 kanban: {
   column_attribute: 'status',
   card_title: 'caller_name',
   card_body: 'notes',
   color_by: 'outcome'
 },

 // GANTT VIEW
 gantt: {
   start_attribute: 'created_date',
   end_attribute: 'scheduled_callback_date',
   progress_attribute: 'workflow_progress',
   resource_attribute: 'assigned_user',
   dependencies_attribute: 'blocked_by'
 },
```
```

```
 // CALENDAR VIEW
 calendar: {
   date_attribute: 'callback_date',
   title_attribute: 'caller_name',
   description_attribute: 'notes',
   color_by: 'priority'
 },

 // FLOWCHART VIEW
 flowchart: {
   states_attribute: 'workflow_states',
   transitions_attribute: 'triggers',
   current_state_attribute: 'status'
 }
}
```

Example - Same PHO Entity, Different Views:

PHONE CALLS viewed as:

KANBAN

NEW	IN_PROGRESS	DONE
[PHO1]	[PHO2]	[PHO3]
[PHO4]	[PHO5]	[PHO6]

CALENDAR

Mon	Tue	Wed	Thu	Fri
PHO1	PHO2		PHO3	
	PHO4	PHO5		

TABLE

DNA	Name	Outcome	Date
PHO001	Mario	YES	10/4
PHO002	Laura	NO	10/4

CHART

Conversions This Week:

 28%

Infinite Custom Views:
Each user can create unlimited views:

1. Configure pivots (rows, columns, filters)
2. Save with a meaningful name
3. Recall at any time
4. Generate automatic reports (PDF, Excel, email)
5. Link to custom dashboards
6. Integrate with ISO 9001 documentation

Automatic ISO 9001 linking:
Through specialized ISO attributes in each entity:

```javascript
iso9001_attributes = {
  document_type: "PROCEDURE",
  document_code: "PR-SALES-001",
  revision: "3.2",
  approval_required: true,
  approved_by: "quality.manager",
  next_review: "2026-04-01",

  // Quando genero report da pivot
  auto_include_in: [
    "Quality_Management_Manual",
    "Sales_Procedures_Register",
    "Audit_Documentation"
  ]
}
```

Result: The system automatically generates all ISO 9001 documentation from the saved pivot views.

6.4 STATUS 9: AI - Help/Promote (Prioritization Assistant)

Ontological Meaning: External action, the delivery of value through intelligence

The Real Problem of Organization:

CONFUSION → WASTED TIME → INEFFICIENCY
People don't know:
- What they should do
- What they can do
- What they MUST do as a priority

Solution: Omnipresent Intelligent TO-DO LIST

AI in 3P3 is primarily an operational prioritization assistant present at every level:

At the Individual Level:

```javascript
ai_personal_assistant = {
user: 'mario.rossi',
current_time: '2025-10-04 09:00',

// Automatic analysis
tasks_analysis: {
total_assigned: 15,
urgent: 3,
blocking_others: 2,
overdue: 1
},

// Smart priority
recommended_sequence: [
{
position: 1,
task: 'TSK250042',
title: 'Complete BOM for PRD78',
why: 'BLOCK 3 other team production tasks',
urgency: 'HIGH',
estimated_time: '45 min',
best_time: 'NOW - morning focus optimal'
},
{
position: 2,
task: 'PHO250089',
title: 'Follow-up Mario Rossi',
why: 'Callback scheduled today 10:00',
urgency: 'MEDIUM',
estimated_time: '15 min',
best_time: '10:00 - customer available'
},
{
position: 3,
task: 'TSK250051',
title: 'Review offer OFC250033',
why: 'Manager waiting approval',
urgency: 'MEDIUM',
```

```
estimated_time: '20 min',
best_time: 'Before lunch'
}
],

// Automatic alerts
alerts: [
"⚠ TSK250038 expired yesterday - HIGHEST priority",
"📊 You have 3 callback calls scheduled today",
"✅ By completing TSK250042 you unlock the team production"
]
}
```

At Team Level:

```
```javascript
ai_team_assistant = {
  team: 'SALES',

  workload_analysis: {
    mario_rossi: {
      current_tasks: 15,
      capacity: 'OPTIMAL',
      suggestion: 'Continue current assignments'
    },
    laura_bianchi: {
      current_tasks: 23,
      capacity: 'OVERLOADED',
      suggestion: 'Redistribute 5 tasks to available team members'
    },
    paolo_verdi: {
      current_tasks: 8,
      capacity: 'UNDERUTILIZED',
      suggestion: 'Can take 7 more tasks'
    }
  },

recommended_actions: [
"Assign TSK250067, TSK250072, TSK250091 from Laura to Paolo",
"Mario can handle urgent callback PHO250105",
"Team meeting suggested today at 3:00 PM for alignment"
]
}
```
```

At Company Level:

```javascript
ai_company_assistant = {

bottleneck_detection: [
{
process: 'PRODUCT_DEVELOPMENT',
bottleneck: 'TDS approval always late',
impact: '12 products blocked',
solution: 'Add second TDS approver',
estimated_gain: '40% faster time to market'
}
],

priority_projects: [
"PRJ250015: Premium customer, deadline 10/10",
"PRJ250022: Trade show prototype, urgent",
"PRJ250008: Standard project, can wait"
],

resource_optimization: {
production_team: 'At 95% capacity - optimal',
sales_team: 'At 78% capacity - can handle more leads',
technical_team: 'At 105% capacity - OVERLOAD'
}
}
```

How AI Helps Concretely:

1. Analyze Dependencies

Task A blocks Tasks B, C, and D
→ Maximum Priority to Task A

2. Consider Deadline

Task overdue > Task due today > Task due tomorrow

3. Assess Business Impact

Premium Client > Standard Client
€50k Project > €5k Project

4. Optimize Timing
"Client available 10:00-11:00"
"Production team free after 2:00 PM"
"Manager approves better in the morning"

5. Learn from Pattern
"Task type X takes 45 minutes on average"
"User Mario is more efficient in the morning"
"Fastest approvals on Tuesday"

Interfaccia Operativa:

```
┌─ YOUR TO-DO TODAY ──────────────────────────┐
│                                             │
│ 🔴 1. [TSK250042] Complete BOM PRD78        │
│    ⚡ BLOCKS 3 team tasks                   │
│    ⏱️ 45min | 🎯 Do NOW                    │
│                                             │
│ 🟡 2. [PHO250089] Callback Mario Rossi      │
│    📅 Scheduled 10:00                       │
│    ⏱️ 15min | 🎯 10:00-10:30               │
│                                             │
│ 🟢 3. [TSK250051] Review OFC250033          │
│    👔 Manager waiting                       │
│    ⏱️ 20min | 🎯 Before lunch              │
│                                             │
│ ○ 4-15. Other tasks (view all)              │
│                                             │
│ 💡 Tip: Completing #1 unlocks team          │
└─────────────────────────────────────────────┘
```

This is pragmatic, operational, everyday – not abstract ML (Machine Learning) but a concrete tool for better organizing every day.

6.5 STATE 10: OPTIMIZATION - Optimize (Budget-Financial Cycle)

Ontological Meaning: Continuous improvement of the entity

The Heart: Comparison of Estimates vs. Actuals

Optimization in 3P3 is based on a continuous cycle:

PLAN (Forecast)
↓
DO (Action)
↓
CHECK (Check - Gap Comparison)
↓
ACT (Learn - Adjust)
↓
Improved PLAN (Next Forecast)

KOOL TOOL Practical Example - PRD Product:

```javascript
// PLAN - Estimate
estimate_PRD78 = {
product: 'PRD78 - Premium Extensions 60cm',
estimated: {
development_days: 5,
bom_cost: 850,
production_hours: 12,
total_cost: 1200,
selling_price: 1800,
margin: 600 // 33%
},
assumptions: [
"Materials available immediately",
"No technical problems",
"Team at 100% efficiency"
]
}

// DO - Actual Execution
// ... the team is working on the product...

// CHECK - Actual Budget
estimate_PRD78 = {
```

```
product: 'PRD78 - Premium Extensions 60cm',
actual: {
development_days: 7, // +40%
bom_cost: 920, // +8%
production_hours: 15, // +25%
total_cost: 1380, // +15%
selling_price: 1800, // unchanged
margin: 420 // 23% instead of 33%
},
issues_encountered: [
"Special order material - 2-day delay",
"Failed prototype - remade (+3 hours)",
"Sick team member - less experienced replacement"
]
}

// ACT - Gap Analysis and Learning
gap_analysis = {
development_time: {
gap: +40,
reason: "Underestimated impact of material delay",
learning: "For special materials, add +2 days buffer"
},
bom_cost: {
gap: +8,
reason: "Special order premium price",
learning: "Material type X always costs +10% if special order"
},
production_hours: {
gap: +25,
reason: "Failed prototype + replacement team",
learning: "Premium products require +20% time buffer"
},

// Update predictive model
update_model: {
next_similar_product: {
development_days: 6.5, // Adjusted
bom_cost_multiplier: 1.08,
production_hours: 14,
confidence: 0.85
}
}
}
```

```
// PLAN - Improved next quote
preventivo_PRD92 = {
product: 'PRD92 - Premium Extensions 55cm',
estimated: {
development_days: 6.5, // Adjusted from experience
bom_cost: 890, // With learned buffer
production_hours: 14, // Realistic
total_cost: 1320,
selling_price: 1900,
margin: 580 // 30% - more realistic
},
applied_learnings: [
"✓ Buffer for special materials",
"✓ Cost of premium material type X",
"✓ Time buffer for premium products",
"✓ Team backup plan"
]
}
```

Continuously Learning System:

```javascript
optimization_engine = {

// Experience database
historical_data: {
products_completed: 847,
accuracy_evolution: {
first_100: 0.65, // 65% accuracy predictions
products_200: 0.72,
products_500: 0.84,
current: 0.91 // 91% accuracy after 847 products!
}
},

// Recognized Patterns
learned_patterns: [
{
pattern: "Special materials → always +2 days",
confidence: 0.95,
applied_times: 143
},
{
pattern: "Premium products → +20% production time",
```

```
confidence: 0.88,
applied_times: 67
},
{
pattern: "Failed prototype → probability 0.15 if new design",
confidence: 0.82,
applied_times: 34
}
],

// Automatic Suggestions
auto_suggestions: [
"PRD92 similar to PRD78 → use adjusted estimate",
"Material type X detected → add +10% cost",
"Team member X assigned → reduce estimate -5% (faster)",
"Premium client detected → increase buffer quality"
]
}
```

Unification Goal ↔ Action:

```javascript
continuous_improvement = {

// Original Goal
original_goal: "Produce premium extensions with 30% margin",

// After 847 PDCA cycles
current_state: {
avg_margin: 0.32, // Target exceeded!
accuracy_forecast: 0.91, // Excellent
customer_satisfaction: 4.7, // Out of 5
on_time_delivery: 0.94 // 94%
},

// New objective (evolved)
evolved_goal: "Produce premium extensions with 35% margin and 98% on-time",

// The entity self-improves
entity_evolution: {
version: "PRD_PROCESS_v3.4",
improvements_vs_initial: [
"+22% margin improvement",
"+40% faster time-to-market",
```

```
"+91% forecast accuracy",
"-67% material waste"
],
next_optimization_targets: [
"Reduce prototype failure rate to 5%",
"Increase automation in BOM creation",
"Predictive material cost fluctuation"
]
}
}
```

The System Gets Smarter and Smarter:

Each Budget→Account cycle enriches knowledge. After hundreds of cycles, the system accurately predicts what will happen, automatically identifies risks, and suggests optimal corrections.

The Entity Is Self-Generating:

Just as KOOL TOOL reflects 30 years of experience in hair color charts, each 3P3 entity reflects the history, culture, and quality accumulated through continuous improvement cycles.

7 INTEGRATION WITH THE BRIDGE

7.1 How This Document Relates to the Technical Specification

This document (Ontology)	Cyril & Osbert Document (Implementation)
10 Vital States	Operational Flow (Section 10)
Matrix 10×10	The 10 Ontological Domains (Section 8)
3 Realms CMP-ETY-LOG	Database Schema (Section 6)
Process Manager	UI Design Section 7.3
Instance Manager	User Interaction Flow 10.2
SuperTable*	Universal Meta-Properties 8.3
Parameter K	K Coefficient Calculation 10.3.7
NAME-CHAR-FLOW	Three-Panel Architecture 7.3.3

7.2 DNA Format - The Universal Code

```
FORMATO: PRXYYNNNN

PRX  = Process code (PHO, TSK, PRD, BOM...)
YY   = Year (25 for 2025)
NNNN = Sequential with anti-collision

ESEMPI:
PHO250042 = Phone call #42 del 2025
TSK250158 = Task #158 del 2025
PRD250001 = Product #1 del 2025
BOM250087 = Bill of Materials #87 del 2025
```

Anti-Collision Algorithm:

```javascript
function generateDNA(process_code) {
  let year = getCurrentYear().slice(-2);
  let sequence = getNextSequence(process_code);

  // Check collision multi-user
```

```
  while (exists(`${process_code}${year}${sequence}`)) {
    sequence++;
  }

  // Format with leading zeros
  let dna = `${process_code}${year}${pad(sequence, 4)}`;

  // Atomic reserve
  reserve_dna(dna);

  return dna;
}
```

7.3 JSON-Driven Architecture

The whole system is metadata-driven:

```
```json
{
  "process_template": {
    "code": "PHO",
    "name": "Phone Call",
    "type": "TUPLE",
    "attributes": [...],  // Con 10 domains per attribute
    "workflow": {...},
    "triggers": {...},
    "interface": {...}
  }
}
```
```

Revolutionary Advantage:
Changing process = changing JSON, not code!

7.4 FileMaker Implementation

```
```javascript
// Everything saved in 3 tables
CMP::Template_JSON // Process definition
CMP::Instance_JSON // Business data
ETY::Orchestrator_JSON // States and relationships
LOG::Action_JSON // Full history

// Dynamic rendering
```
```

```
script: "Render_Form_From_JSON"
script: "Process_10_Domains"
script: "Execute_Triggers"
script: "Generate_SuperTable_Pivot"
```

8 VISION AND APPLICATIONS

8.1 The K Coefficient - Measure of Order

K = Extracycles / (Performance × Presenteeism)

K < 1.5 = Efficient system (3P3 is working)
K = 1.5-2.0 = Needs optimization
K > 2.0 = Chaotic system (reorganize)

KOOL TOOL Example:
- Before 3P3: K = 2.8 (high disorganization)
- After 6 months: K = 1.6 (improved, being optimized)
- 12-month target: K = 1.2 (operational excellence)

8.2 ISO 9001 Automatic

Vision: From the work of Times and Methods in the Process Manager, all ISO 9001 documentation automatically arises.

Come Funziona:

Process Manager defines:
├─ Who does what (responsibilities) → CHAR panel with authorization
├─ When and how (procedures) → FLOW panel with operations
├─ With which tools (resources) → Attributes with material domain
└─ Control measures (KPIs) → Performance domain + K coefficient

SuperTable with Pivot automatically generates:
├─ Operating procedures (from FLOW workflows)
├─ Work instructions (from elementary operations)
├─ Forms and registers (from LOG with ISO9001 attributes)
├─ Organizational chart (from NAME hierarchical tree)
├─ Quality dashboard (from custom pivots)
└─ Audit report (from budget-actual comparison)

Specialized ISO Attributes:

```javascript
iso9001_domain = {
  document_code: "PR-SALES-001",
```

```
  document_type: "PROCEDURE",
  revision: "3.2",
  approval: {
    required: true,
    approved_by: "quality.manager",
    approved_date: "2025-09-15"
  },
  review: {
    frequency: "annual",
    next_review: "2026-09-15"
  },
  distribution: {
    internal: true,
    controlled_copy: true,
    recipients: ["sales_team", "quality_dept"]
  }
}
```

Benefit: ISO certification without extra effort – the system is already compliant by design.

8.3 Educational Video Game

Revolutionary idea: Transform the system into a gamified training tool.

```
```javascript
training_game = {
concept: "Organizational training through gameplay",

// Progressive levels
level_1: {
scenario: "You are new Sales Rep at KOOL TOOL",
mission: "Make 10 successful phone calls",
success_criteria: {
calls_completed: 10,
conversion_rate: "> 20%",
avg_duration: "< 20 min",
k_coefficient: "< 2.0"
},
hints_enabled: true,
ai_coach: {
real_time_tips: true,
best_practices: "Learn from 847 historical calls",
mistakes_feedback: "Immediate correction suggestions"
```
```

```
}
},

level_2: {
scenario: "Handle difficult customer",
challenges: [
"Customer angry about previous order",
"Wants 30% discount immediately",
"Threatens to switch to competitor"
],
learn_from: "500+ successful difficult customer cases",
success_metrics: {
customer_retained: true,
reasonable_discount: "< 15%",
future_order_secured: true
}
},

level_3: {
scenario: "Organize complete product launch PRD",
complexity: "Managing 6 sub-entities (BOM, TDS, MDC, MAD, PHO, OFC)",
learn: [
"Process orchestration",
"Dependencies management",
"Team coordination",
"Estimate vs Final accuracy"
]
},

// Achievements system
achievements: [
{
name: "First INTERESTED outcome",
badge: "🎯 Converter",
unlocks: "Advanced sales techniques tutorial"
},
{
name: "10 calls in one day",
badge: " ⚡ Speed Demon",
unlocks: "Time management masterclass"
},
{
name: "Conversion rate > 30%",
badge: "🏆 Sales Champion",
unlocks: "Premium customer handling"
```

```
},
{
name: "K coefficient < 1.5",
badge: "🎓 Organization Master",
unlocks: "Certification Times and Methods"
},
{
name: "Perfect quote accuracy",
badge: "🔮 Oracle",
unlocks: "Strategic planning tools"
}
],

// Leaderboard competition
leaderboard: {
team_competition: true,
metrics: [
"Conversion rate",
"Customer satisfaction",
"K coefficient",
"Forecast accuracy"
],
rewards: {
weekly_winner: "Choose favorite process to optimize",
monthly_champion: "Design new entity for company",
yearly_master: "Become Times and Methods trainer"
}
},

// Simulation modes
simulation_modes: [
{
mode: "Practice Mode",
description: "Safe environment - errors don't affect real data",
use_case: "New employee training"
},
{
mode: "Challenge Mode",
description: "Real-world scenarios with time pressure",
use_case: "Skills improvement"
},
{
mode: "Team Battle",
description: "Collaborative problem solving",
use_case: "Team building and process optimization"
```

```
}
]
}
```

Result: Users **train** to organize better through engaging gameplay, not boring training. They learn by doing, in a fun way that has a real impact on business performance.

8.4 The Promise of the System

When the entity is understood as a **UNIT**:

- ✅ Self-configuration - Interfaces emerge from the ontological structure
- ✅ Self-documentation - The ontology IS the documentation (ISO 9001 included)
- ✅ Self-evolution - The system continuously learns and improves
- ✅ Self-training - Users learn by using the system (video games)
- ✅ Self-certification - Quality embedded by design
- ✅ Self-optimization - The budget-cost cycle optimizes performance

9 CONCLUSION: UNITY IN MULTIPLICITY

9.1 The Revolution Accomplished

After 30 years of research, the 3P3 system achieves a paradigm shift:

FROM: Dualistic Fragmentation
Database + Code + UI (separate and to be integrated)

TO: Ontological Unity
ONE ENTITY that manifests itself in multiplicity

9.2 The Central Message

"Everything comes from one, and one contains all."
The 3P3 entity is not an abstract philosophical concept but a concrete operational reality:

- In the Process Manager, we DEFINE it (States 1-3: NAME-CHAR-FLOW)
- In the Instance Manager, we EXPERIENCE it (States 4-6: GUI-INFO-DATABASE)
- In the SuperTable, we UNDERSTAND it in its entirety (States 7-10: FILTER-PIVOT-AI-OPTIMIZATION)

9.3 The 3000 Worlds in an Instant

10 Factors (base attributes)
× 10 Domains (meta-abilities for each attribute)
= 100 Abilities per attribute-type
× 10 Worlds (Manging 3 tables, Managing Process Manager, Managing Instance Manager, Managing Super Table)
= 1000 possible life states
× 3 Realms (CMP-ETY-LOG manifestations)
————————————————————————————
3000 simultaneous manifestation

BUT THE ENTITY IS ALWAYS ONE

This is the true ontological synthesis, even if we do not express it in Buddhist terminology.

9.4 The Times and Methods of the Future

The 3P3 Process Manager represents the evolution of traditional Time and Methods:

CLASSIC TIME AND METHODS:
- Time operations
- Define work cycles
- Optimize flows

3P3 TIME AND METHODS:
- Define corporate ONTOLOGY
- Create self-evolving intelligent entities
- System learns from every cycle
- Organization self-optimizes
- Future natural ontological language

The Time and Methods expert becomes an ontological architect - not only measures work, but defines the very essence of the organization.

9.5 The Near Future

THE BRIDGE is not "management software" but:

- An ontological tool for organizing business reality
- A living system that continuously learns and evolves
- A training platform that trains people through gameplay
- An automatic generator of ISO 9001 compliance and quality
- An intelligent assistant that prioritizes daily work
- A predictive oracle that improves estimates through actual results

9.6 The Impact on KOOL TOOL

BEFORE 3P3:
- K coefficient: 2.8 (disorganization)
- Undocumented processes
- Lengthy and expensive training
- ISO 9001 difficult to maintain
- Inaccurate quotes

AFTER 6 months of 3P3:
- K coefficient: 1.6 (43% improvement)
- 116 ontologically defined processes
- Training through video games

- ISO 9001 automatically implemented by the system
- Quotes with 91% accuracy

TARGET 12 months:
- K coefficient: 1.2 (excellence)
- Fully self-optimizing system
- Zero time wasted on organizational confusion
- Documentation always automatically updated
- Nearly perfect forecasts (>95% accuracy)

9.7 The Journey Continues

Every organization that adopts 3P3 discovers new possibilities, because the entity is inexhaustible in its richness.

Every company that uses 3P3 builds its own organizational intelligence;one layer at a time, one cycle at a time, one learning experience at a time.

The system grows with you.
It gets smarter every day.
It reflects your excellence.

Because everything is an entity, and the entity is you;your company, your team, your vision;manifested in a digital form that lives, learns, and evolves.

10 TECHNICAL-ONTOLOGICAL GLOSSARY

TOntological term	Technical Meaning	Implementation
Entity	Complete Business Process	Template + Instances
DNA	Unique Code PRXYYNNNN	PHO250042
10 Factors	Basic Attributes	caller_name, phone_number...
10 Domanis	Meta-Attributes for Each Factor	Identity, Temporal, Trigger.
Matrix 10×10	100 Capabilities per Attribute	JSON domains config
10 World	Algorithm Vital States	NAME→OPTIMIZATION
3 Realms	Existential Tables	CMP-ETY-LOG
TUPLE	Simple Entity	Form singolo
FILE	Complex Entity	Multi-tab interface
SuperTable	Total Aggregate View	Query tripartita
Coefficient K	System Efficiency KPI	K = Extra/(Rend×Pres)
Process Manager	Ontology Definition Tool	3 panel: NAME-CHAR-FLOW
Instance Manager	Instance Lifetime Tool	Auto-generated dynamic forms
NAME	Identity and Hierarchical Position	Multilevel tree view
CHAR	Structural Characteristics	Attributes + Interface preview
FLOW	Behavior and Workflow	Operations + States + Triggers
Pivot*	Dynamic Data Reclassification	Infinite custom views
Times and Methods 3P3	Ontological Organization	Define processes = Define company

11 SPECIAL THANKS

This work would not have been possible without the people who have walked beside me during these thirty years of research.
To Neta, my wife: Thank you for believing in this vision even when it seemed impossible. Your patience through the ups and downs, the moments of doubt and those of enthusiasm, was the foundation on which I was able to build. Every time I lost myself in the depths of ontology, your gaze and your wisdom reminded me what truly matters.
To Federico, my son: You taught me what it means to love unconditionally. This lesson has become the guiding principle of the 3P3 system: "Does this create value and happiness for all beings?" Every concept bears the imprint of what I learned from you.
To Libi, my daughter: In the most difficult moments, your vitality brought light. Even though you were still young, you taught me a profound truth: the wisdom and intuition to distinguish good from evil are ageless. Sometimes children see clearly what we adults obscure with our thoughts.

A profound thanks also goes to my teacher, Daisaku Ikeda: He taught me the essence of Buddhism, the master-disciple relationship, and showed me how daily practice can transform suffering into value. His vision of a human revolution, where each person can manifest their infinite potential, has become the beating heart of this system: the entity that contains within itself all possible worlds, which learns, evolves, and creates value.
To all my fellow believers: Thank you for having walked together over the years, working every day to put the principles of Buddhism into practice and build peace. Especially to Francesco, Rika, Giorgio, Ersilio, and many others;your lives have taught me the path of transformation toward unity. This is not just a computer system: it is an attempt to translate into technology what we have learned together about the interconnected nature of reality.

To all those who have accompanied me: For better or for worse, each of you has left a mark on this work. Even challenges and failures have become valuable lessons, transforming into the "organizational intelligence" that the 3P3 system seeks to capture.
To the engineers who attempted to translate philosophy into code: I thank every developer who sought to understand this ontological vision and transform it into digital reality. It has never been easy to build a bridge between the language of being and machine language.
To Cyril and Osbert: You have done something extraordinary, not only have you understood the profound essence of the 3P3 project, but you have also been able to create the trust that is the true foundation of any collaboration. THE BRIDGE is not just the name of the software: it is what we have built together between ontology and implementation, between vision and reality. Your ability to listen,

question, and then translate into technical architecture is giving concrete form to thirty years of research.

This document represents much more than a technical specification or a philosophical treatise. It is the crystallization of an existential journey; from the suffering of fragmentation to the joy of unity, from the chaos of disorganized organization (K=2.8) to the harmony of excellence (K=1.2).
Looking back over these thirty years (1996-2025), I realize that every failure was a necessary learning experience, every deviation an indispensable exploration. Like the 3P3 system that learns from the Budget-Consumer cycle, I too have grown through the continuous iteration between vision and reality.
From here, a new era begins.
Like every 3P3 entity that is born, grows, learns, and evolves, this project is now entering its most important phase of life: the one in which it will no longer be just mine, but will belong to all those who will use it, improve it, and make it grow.

"Everything comes from one, and the one contains all."
Luca Meggiolaro
KOOL TOOL SRL - Romania
October 5, 2025
For Neta, Federico, and my little bright star

Have a good journey everyone in the 3p3 ontology 🚀

Luca Meggiolaro

KOOL TOOL SRL - România
Building the Future Through Ontological Understanding

Versione 3.0 - 05 Ottobre 2025
Technology at the service of human happiness

12 FINAL NOTE

This document should be read in conjunction with Cyril & Osbert's Full Technical Specification.

- This document: Explains the WHY (ontology)
- Cyril & Osbert Document: Explains the HOW (implementation)

Together they form the complete knowledge for building THE BRIDGE.

The 3P3 system is not complicated—it is profound.

Once the unity of the entity is understood, everything becomes clear.

Significant excerpts from Nichiren Daishonin's Gosho "The True Entity of Life":
[... "the true entity is invariably revealed in all phenomena" ..... "all phenomena invariably possess the Ten Factors" ..... "The Ten Factors operate in the Ten Worlds, and the Ten Worlds invariably encompass 'body and earth.'"

In the Gosgo, "The Legacy of the Ultimate Law of Life and Death," we read:
[....it is stated that living beings passing through the two phases of life and death are the entities of the Ten Worlds, or the entities of Myoho-renge-kyo.]

12.1 REFERENCES

1. **The Bridge - Full Technical Specification** (Cyril Amegah & Osbert Vulor)
   - Implementazione JSON-driven completa
   - API REST specification
   - Database schema FileMaker

2. **THE BRIDGE - Process Manager HTML** (Final Version)
   - Interfaccia 3 pannelli NAME-CHAR-FLOW
   - Gestione entità visuale
   - Tree gerarchico multilivello

3. **Process List KOOL TOOL** (116+ processi)
   - Struttura organizzativa reale
   - Mapping Main_Process → Process → Sub_Process