

Ontología filosófica y modelo operacional en software

Genealogía conceptual, autores y uso contemporáneo

Jesús Pérez Lorenzo

OntoRef · 2026

Contenidos

1. ¿Son los invariantes el equivalente a los universales filosóficos?	3
1.1. Los universales en ontología filosófica	3
1.2. Los invariantes como propiedades esenciales de un singular	3
1.3. Dónde sí hay resonancia con los universales	3
1.4. El estatus ontológico: hechos institucionales	4
2. Genealogía filosófica del modelo operacional	5
2.1. Matices que la tabla no captura	6
3. La ontología operacional en software: origen, autores y uso actual	7
3.1. El problema que había que resolver	7
3.2. Los fundadores	7
3.3. Línea temporal condensada	8
3.4. Uso actual	9
3.5. Conclusión	10
4. Taoísmo, Yin-Yang y ontología operacional	11
4.1. Lo que la tradición occidental no vio	11
4.2. Yin-Yang — polaridad constitutiva, no oposición	11
4.3. Correspondencias con el modelo operacional	11
4.4. Las cuatro correspondencias profundas	12
4.5. La crítica taoísta a la ontología clasificatoria	13
5. DAGs — genealogía y función en ontología operacional	14
5.1. El problema que fuerza el grafo	14
5.2. Por qué la aciclicidad — el problema de la justificación circular	14
5.3. Línea genealógica	14

5.4. Las cuatro funciones del DAG en ontología operacional	15
5.5. La tensión con el Taoísmo — DAG vs. ciclo	16
6. Física cuántica, computación cuántica y ontología operacional	17
6.1. El golpe más profundo — Aristóteles estaba equivocado en el nivel fundamental	17
6.2. Las tres conexiones genuinas	17
6.3. La interpretación más relevante — Mecánica Cuántica Relacional	18
6.4. La computación cuántica — honestidad sobre sus límites	18
6.5. La conexión no obvia — LLMs como geometría cuántica aplicada	19
7. Por qué es relevante en software e infraestructuras digitales	20
7.1. El problema real no es técnico — es epistémico	20
7.2. Lo que la ontología operacional hace que nada más hace	20
7.3. Por qué los DAGs — especificidad técnica	20
7.4. Por qué la reflection — la ontología sin operación es un museo	21
7.5. Las tres presiones que hacen esto urgente ahora	21
7.6. La síntesis	22
8. Referencias y lecturas recomendadas	23
8.1. Filosofía clásica y ontología	23
8.2. Inteligencia artificial y representación del conocimiento	24
8.3. Ingeniería de software y praxis	25
8.4. Perspectivas adicionales	25
8.5. Física cuántica y fundamentos	26
8.6. Ingeniería de software e infraestructura	27

1. ¿Son los invariantes el equivalente a los universales filosóficos?

La pregunta es legítima pero la correspondencia es más fina de lo que parece a primera vista.

1.1. Los universales en ontología filosófica

Los **universales** responden a una pregunta específica: *¿qué comparten múltiples particulares que los hace del mismo tipo?* Son propiedades, relaciones o cualidades que pueden instanciarse en varios individuos distintos — «rojez», «humanidad», «ser-mayor-que». El debate clásico gira en torno a su estatus ontológico:

- **Realismo platónico:** los universales existen independientemente de sus instancias — las Formas habitan un reino separado del mundo sensible.
- **Realismo moderado** (Aristóteles): los universales existen *en* las cosas, no aparte de ellas. No hay humanidad sin seres humanos concretos.
- **Nominalismo** (Ockham, siglo XIV): solo existen los particulares. Los universales son nombres (*nomina*), convenciones lingüísticas sin referente ontológico propio.
- **Conceptualismo** (posición media): los universales existen en la mente como conceptos, no en el mundo ni como meros nombres.

1.2. Los invariantes como propiedades esenciales de un singular

Los invariantes en un modelo operacional de software no son universales en ninguno de los sentidos anteriores. No preguntan qué comparten múltiples proyectos — preguntan *qué no puede cambiar en este proyecto concreto sin que deje de ser este proyecto*.

La analogía filosófica precisa es la distinción aristotélica entre **propiedades esenciales y accidentales**: la esencia es aquello sin lo cual una cosa no sería lo que es; el accidente es lo que puede cambiar sin alterar su naturaleza. Un invariante operacional declara: «esta propiedad es esencial para la identidad de este proyecto.»

Kripke formalizó esto de manera más rigurosa en *Naming and Necessity* (1980): ciertas propiedades son **rígidas** — verdaderas en todos los mundos posibles accesibles al individuo. El agua es necesariamente H₂O aunque alguien la llame «phlogiston». Un invariante operacional tiene este carácter modal: no puede ser falso en ningún estado posible del proyecto sin que el proyecto haya dejado de ser ese proyecto. Los ADRs que lo modifican no *violan* el invariante — constituyen un nuevo individuo.

1.3. Dónde sí hay resonancia con los universales

La familia semántica es compartida: ambos capturan *lo que no varía*. El universal no varía a través de la instanciación (la rojez es la misma en esta manzana y en esa). El invariante no varía a través de la evolución temporal del proyecto. El eje difiere — instanciación vs. tiempo — pero el gesto filosófico es análogo.

1.4. El estatus ontológico: hechos institucionales

Hay una diferencia de fondo que ninguna de las categorías anteriores captura. Los universales platónicos existen independientemente de que alguien los declare. Los invariantes operacionales son **hechos institucionales** en el sentido de John Searle (*La construcción de la realidad social*, 1995): son verdad porque un colectivo los declaró y formalizó. Como el dinero, los contratos o las reglas del ajedrez — su existencia depende de actos declarativos sostenidos por una práctica social. Un invariante que el equipo no reconoce como tal no es un invariante real; es teatro.

Esto los hace más frágiles que los universales metafísicos, pero también más honestos sobre su naturaleza.

2. Genealogía filosófica del modelo operacional

La siguiente tabla mapea los elementos básicos de la ontología filosófica a sus equivalentes en modelos operacionales de software, con autores y origen temporal.

Concepto filosófico	Autor / Origen	Equivalente operacional	Implementación
Universales	Platón (428–348 aC) <i>República, Parménides</i>	Tipos / Schemas	Contratos Nickel: String, ['Accepted, 'Proposed]
Esencia / Accidente	Aristóteles (384–322 aC) <i>Metafísica</i>	Invariante / Estado mutable	invariante = true en core.ncl vs. dimensiones en state.ncl
Potencia / Acto	Aristóteles <i>Metafísica IX</i>	Estado actual → deseado + Gates	current_state / desired_state / gate.ncl
Categorías (10 predicamentos)	Aristóteles <i>Categorías</i>	Tipos de nodo ontológico	invariants, tensions, practices, dimensions, gates
Sustancia primera / segunda	Aristóteles <i>Categorías</i>	Instancia / Tipo	Proyecto concreto vs. schema de proyecto
Identidad a través del tiempo	Locke (1689) <i>Ensayo</i> Parfit (1984) <i>Reasons</i>	Versioning + ADR lifecycle	status = Superseded, superseded_by — identidad que evoluciona formalmente
Dialéctica	Hegel (1807) <i>Fenomenología del Espíritu</i>	Tensiones activas	tensions en core.ncl — sin síntesis, gestionadas permanentemente
Logos como tensión de opuestos	Heráclito (535–475 aC)	Tensión constitutiva irreducible	Dualidad ontología / reflection como tensión de diseño
Mundos posibles	Leibniz (1710) <i>Teodicea</i> Kripke (1980) <i>Naming and Necessity</i>	FSM — transiciones permitidas	ADRs como restricciones de accesibilidad entre estados
Propiedades rígidas / esenciales	Kripke (1980) <i>Naming and Necessity</i>	Invariantes como necesidad, no contingencia	Un invariante es verdad en todos los estados posibles del proyecto
Límites del lenguaje = límites del mundo	Wittgenstein (1921) <i>Tractatus Logico-Philosophicus</i>	Schema como frontera de lo expresable	Lo que no tipea en Nickel no puede existir en el sistema
Juegos de lenguaje	Wittgenstein (1953) <i>Investigaciones filosóficas</i>	Modos operacionales	Un modo define las reglas del juego para un contexto operacional

Concepto filosófico	Autor / Origen	Equivalente operacional	Implementación
Verdad como correspondencia	Tarski (1933) <i>El concepto de verdad</i>	Typecheck / validación formal	nickel export como verificación de correspondencia con el schema
Hechos institucionales	Searle (1995) <i>La construcción de la realidad social</i>	ADRs como decisiones declaradas	Una constraint existe porque el equipo la declaró y comprometió
Ontología formal vs. regional	Husserl (1913) <i>Ideas I</i>	Schema formal vs. ontología del proyecto	adrs/schema.ncl (formal) vs. .ontology/core.ncl (regional)
Filosofía del proceso	Whitehead (1929) <i>Process and Reality</i>	Ontología (ser) vs. Reflection (devenir)	La dualidad estructural del protocolo — capas con ritmos distintos
Realismo estructural	Ladyman & Ross (2007) <i>Every Thing Must Go</i>	Grafo como lo real, no las entidades	El DAG captura relaciones; los nodos son secundarios
Lógica de predicados	Frege (1879) <i>Begriffsschrift</i>	Queries sobre el grafo	where severity == "Hard" select id claim

2.1. Matices que la tabla no captura

Aristóteles es el arquitecto base. La esencia/accidente es exactamente invariante/estado-mutable. La potencia/acto es exactamente `current_state/desired_state` con la gate como umbral de actualización. Que esto haya tardado 2.300 años en aparecer explícitamente en software es indicativo de cuánto tiempo estuvo implícito en las decisiones de diseño.

Whitehead contra Aristóteles es la tensión interna del modelo. Aristóteles ve la realidad como sustancias con propiedades. Whitehead la ve como procesos — no hay cosas, hay eventos. Un modelo operacional de software vive en esa grieta: la ontología es aristotélica (qué *es* el proyecto), la capa de reflection es whiteheadiana (cómo *deviene*). Mantenerlas separadas es un reconocimiento implícito de que ninguna basta sola.

Kripke resuelve lo que Aristóteles dejó ambiguo. Aristóteles distingue esencial de accidental pero no formaliza la modalidad. Kripke lo hace: una propiedad esencial es rígida — verdadera en todos los mundos posibles accesibles desde el individuo. Los invariantes operacionales tienen exactamente este carácter modal. Los ADRs que los modifican no violan los invariantes — constituyen un nuevo individuo.

La ausencia notable: Heidegger. El *ser-en-el-mundo* como condición de posibilidad de cualquier ontología explícita no aparece en ningún modelo operacional de software. Lo que Heidegger llamaría el trasfondo pre-temático — el contexto que no se puede formalizar porque es la condición de posibilidad de la formalización — es exactamente lo que los READMEs intentan capturar y fracasan. Un modelo operacional no lo resuelve; lo que hace es reducir la superficie de ese trasfondo haciendo explícito lo que puede serlo.

3. La ontología operacional en software: origen, autores y uso actual

3.1. El problema que había que resolver

A finales de los años 1970, los sistemas de inteligencia artificial enfrentaban un problema conocido como **el cuello de botella del conocimiento**: los sistemas expertos eran eficaces en dominios muy acotados pero no podían compartir conocimiento entre sí ni razonar fuera de su dominio específico. Cada sistema construía su representación del mundo de forma ad hoc, incompatible con el resto.

La pregunta que emergió fue: ¿es posible especificar formalmente el conocimiento de un dominio de manera que múltiples sistemas puedan compartirlo y razonar sobre él?

3.2. Los fundadores

3.2.1. John McCarthy — el precursor (1958–1980s)

John McCarthy, fundador del término «inteligencia artificial» y creador de LISP, fue el primero en proponer que los sistemas de IA necesitaban *representaciones explícitas del mundo* para razonar sobre él. Su noción de **situational calculus** (1969, con Patrick Hayes) — un lenguaje formal para describir el mundo y sus cambios — es el antecedente directo de la ontología operacional.

McCarthy adoptó el término «ontología» de la filosofía para referirse a los compromisos sobre qué tipos de cosas existen en el dominio modelado. No era una metáfora — era un préstamo deliberado de la tradición aristotélica.

3.2.2. Marvin Minsky — frames y estructuras de conocimiento (1974)

En su influyente artículo *A Framework for Representing Knowledge* (MIT AI Lab, 1974), Minsky propuso los **frames** como estructuras para organizar el conocimiento: cada frame representa un concepto con slots (atributos) y valores por defecto. Es la primera implementación computacional de algo cercano a la sustancia aristotélica con propiedades.

Los frames son el antecedente directo de los esquemas orientados a objetos y, más tarde, de las ontologías formales.

3.2.3. Tom Gruber — la definición canónica (1993)

Tom Gruber, en el Knowledge Systems Laboratory de Stanford, formuló la definición que se convirtió en estándar: «*An ontology is a specification of a conceptualization*» (*A translation approach to portable ontology specifications*, Knowledge Acquisition, 1993).

La definición es deliberadamente minimalista: una ontología no captura la realidad — captura un *punto de vista compartido* sobre un dominio. Esto la aleja del realismo metafísico y la sitúa en el territorio de los hechos institucionales de Searle.

Gruber también desarrolló ONTOLINGUA, el primer sistema formal para escribir y compartir ontologías entre sistemas de IA heterogéneos.

3.2.4. Nicola Guarino — ontología formal en sistemas de información (1990s)

Nicola Guarino, del Instituto de Ciencias y Tecnologías Cognitivas (ISTC-CNR) de Italia, desarrolló la distinción entre **ontología de alto nivel** (conceptos universales como tiempo, espacio, proceso, objeto) y **ontología de dominio** (conceptos específicos de un campo).

Fundó la conferencia *Formal Ontology in Information Systems* (FOIS, 1998), aún activa, que establece el puente entre la filosofía analítica y la ingeniería del conocimiento. Su trabajo sobre la distinción entre ontologías **descriptivas** (cómo los agentes conceptualizan el mundo) y **prescriptivas** (cómo deben conceptualizarlo para interoperar) es directamente aplicable a los modelos operacionales de software.

3.2.5. Doug Lenat — Cyc: el proyecto más ambicioso (1984–2017)

Doug Lenat inició el proyecto **Cyc** en 1984 en MCC (Microelectronics and Computer Technology Corporation), con el objetivo de codificar todo el conocimiento de sentido común humano como una ontología operacional. Fue el proyecto de representación de conocimiento más grande y longevo de la historia.

Cyc alcanzó más de 600.000 conceptos y 5 millones de aserciones. Demostró tanto el potencial como los límites del enfoque: la escala del conocimiento implícito humano es prácticamente inabarcable mediante codificación manual.

3.2.6. Tim Berners-Lee, James Hendler y Ora Lassila — la Web Semántica (2001)

En su artículo seminal en *Scientific American* (2001), propusieron extender la Web con significado semántico: páginas que no solo humanos sino también máquinas pudieran «entender». La ontología era el mecanismo de interoperabilidad semántica.

Esto produjo las tecnologías RDF (Resource Description Framework), OWL (Web Ontology Language) y SPARQL — el ecosistema de la Web Semántica — que popularizaron el uso de ontologías en ingeniería de software más allá de la IA.

3.3. Línea temporal condensada

Año	Hito
1958	McCarthy propone representación explícita del conocimiento para IA
1969	McCarthy & Hayes — situational calculus: lógica de cambios en el mundo
1974	Minsky — frames: primera estructura computacional análoga a la sustancia aristotélica
1984	Lenat inicia Cyc — codificación masiva de conocimiento de sentido común
1991	Primeras ontologías de alto nivel: DOLCE, SUMO — conceptos universales compartidos

Año	Hito
1993	Gruber — definición canónica: «ontología = especificación de una conceptualización»
1998	Primera conferencia FOIS (Formal Ontology in Information Systems)
2001	Berners-Lee et al. — Web Semántica: RDF, OWL, SPARQL
2004	OWL se convierte en recomendación W3C — estándar industrial
2006–2010	Wikidata, DBpedia — ontologías abiertas a escala masiva
2012	Google Knowledge Graph — ontología operacional en producción para miles de millones de usuarios
2020s	LLMs entrenados sobre grafos de conocimiento ontológicos; ontologías como contexto estructurado para agentes de IA

3.4. Uso actual

3.4.1. Biomedicina — el dominio más maduro

La ontología operacional tiene su mayor penetración en ciencias biomédicas:

- **Gene Ontology (GO)**: iniciada en 1998, describe funciones génicas de forma estandarizada entre organismos. Usada en todos los laboratorios de biología molecular del mundo.
- **SNOMED CT**: más de 350.000 conceptos médicos con relaciones formales. Estándar en sistemas de historia clínica electrónica en Europa, EEUU, Australia.
- **ICD-11 (OMS)**: clasificación internacional de enfermedades en formato ontológico desde 2018.
- **Human Phenotype Ontology (HPO)**: diagnóstico de enfermedades raras mediante matching ontológico entre síntomas del paciente y fenotipos conocidos.

3.4.2. Grafos de conocimiento empresariales

- **Google Knowledge Graph (2012)**: más de 500.000 millones de hechos estructurados que alimentan los resultados de búsqueda y el panel de conocimiento.
- **Wikidata**: ontología abierta con más de 100 millones de elementos, mantenida colaborativamente, usada como fuente de conocimiento por Wikimedia, investigación académica y sistemas de IA.
- **schema.org**: vocabulario ontológico estándar para marcado semántico de páginas web, creado por Google, Microsoft, Yahoo y Yandex. Presente en la mayoría de sitios web modernos.

3.4.3. Industria y manufactura

- **ISO 15926** (petróleo y gas): ontología para integración de datos a lo largo del ciclo de vida de plantas industriales. Usada por Shell, BP, Statoil.
- **IFC** (Industry Foundation Classes): ontología para información de construcción. Base del estándar BIM (Building Information Modeling) en arquitectura e ingeniería civil.

- **Industry 4.0 ontologies:** RAMI 4.0, Asset Administration Shell — representación formal de activos industriales en fábricas conectadas.

3.4.4. Inteligencia artificial y agentes

- **ConceptNet:** grafo de conocimiento de sentido común con 21 millones de aserciones, usado en entrenamiento de modelos de lenguaje.
- **Entrenamiento de LLMs:** Wikidata y DBpedia son fuentes estructuradas en los corpus de entrenamiento de GPT, LLaMA y modelos similares.
- **Contexto para agentes:** el uso emergente más relevante — ontologías como contexto estructurado que se inyecta en agentes de IA para que operen con comprensión del dominio en lugar de inferirlo del lenguaje natural.

3.4.5. Software de proyecto — la frontera actual

El uso de ontologías operacionales *dentro* de proyectos de software — no para describir dominios externos sino para describir el propio proyecto — es el desarrollo más reciente y menos formalizado.

Los precursores son los ADRs (Architecture Decision Records, propuestos por Michael Nygard en 2011) y los modelos C4 (Simon Brown). Pero estos son documentación, no grafos consultables.

La propuesta de **ontoref** pertenece a esta frontera: aplicar las estructuras de la ontología formal — invariantes tipados, tensiones explícitas, estados formalizados, gates verificables — al proyecto de software como entidad que se describe a sí misma. El proyecto *es* el dominio, y la ontología *es* el modelo operacional de ese dominio.

Lo que distingue este enfoque de la documentación tradicional es que el modelo es:

- **Consultable:** mediante pipelines estructuradas, no búsqueda de texto
- **Verificable:** el typecheck valida la coherencia en cada commit
- **Ejecutable:** los modos operacionales son grafos acíclicos que el sistema puede ejecutar
- **Machine-readable:** los agentes de IA pueden consumirlo directamente como contexto estructurado

3.5. Conclusión

La ontología operacional en software no es una importación reciente de la filosofía — es una convergencia que lleva setenta años ocurriendo, desde los frames de Minsky hasta los grafos de conocimiento de Google. Lo que ha cambiado con la proliferación de agentes de IA es la urgencia: cuando un agente puede modificar código, infraestructura o configuración de forma autónoma, la ausencia de un modelo explícito del proyecto no es una deuda técnica. Es una pérdida de control.

La ontología no es la solución a ese problema. Es el nombre del problema que hay que resolver.

4. Taoísmo, Yin-Yang y ontología operacional

4.1. Lo que la tradición occidental no vio

La ontología occidental desde Parménides privilegia el **ser** sobre el **no-ser**, la **presencia** sobre la **ausencia**, la **identidad** sobre la **diferencia**. Toda la tradición aristotélica-escolástica-analítica trabaja desde ese punto de partida.

El Taoísmo invierte la pregunta. El Tao (道) es previo a la distinción ser/no-ser. El primer capítulo del *Tao Te Ching* (Laozi, siglo VI-IV aC):

道可道，非常道。名可名，非常名。

El Tao que puede nombrarse no es el Tao eterno.

El nombre que puede pronunciarse no es el nombre eterno.

Lo que no puede formalizarse no es una limitación del modelo — es la condición de posibilidad de todo lo que sí puede formalizarse. La ontología operacional que puede expresarse completamente en un schema no captura la realidad operacional completa. Heidegger llegó al mismo lugar por camino diferente: el **trasfondo pre-temático** que hace posible cualquier acto de comprensión explícita es precisamente lo que escapa a la formalización.

4.2. Yin-Yang — polaridad constitutiva, no oposición

La confusión más frecuente: yin y yang no son **contrarios** sino **complementarios que se definen mutuamente**. Las propiedades estructurales del par son:

- Cada polo **contiene una semilla del opuesto** — representada por el punto de color contrario en el símbolo
- El exceso de un polo **genera su contrario** (極則反 — la extremidad se invierte)
- La tensión entre ellos **nunca se resuelve** en síntesis — solo se gestiona dinámicamente
- Uno **no existe sin el otro** — yin sin yang es vacío inerte, yang sin yin es movimiento sin forma

Esto es estructuralmente distinto de la dialéctica hegeliana, que busca síntesis — resolver la contradicción en una unidad superior. El yin-yang no resuelve. La tensión es constitutiva, permanente. Hegel usa la contradicción como **palanca**; el Taoísmo la habita como **condición**.

Heráclito (535-475 aC) es el occidental más cercano: su logos como tensión dinámica de opuestos — «el camino hacia arriba y hacia abajo son el mismo» — es estructuralmente idéntico al yin-yang. La diferencia: Heráclito vive en la tensión, Hegel la usa para superarla.

4.3. Correspondencias con el modelo operacional

Concepto taoísta	Texto / Época	Equivalente occidental más cercano	Correspondencia operacional
Tao (道) — principio previo a toda distinción	Laozi, <i>Tao Te Ching</i> (6-4 aC)	Ser de Heidegger — previo a los entes	Trasfondo pre-temático que ningún schema captura completamente

Concepto taoísta	Texto / Época	Equivalente occidental más cercano	Correspondencia operacional
Yin / Yang — polaridad que no se resuelve	Tradición Zhou (1100 aC) sistematizada en <i>I Ching</i>	Heráclito — logos como tensión de opuestos	Ontología (ser) / Reflection (devenir) — capas que se necesitan mutuamente
Wu wei (無為) — acción sin forzar	Laozi, <i>Tao Te Ching</i>	Sin equivalente exacto en Occidente	«No enforcement» — coherencia que emerge de adopción justificada, no impuesta
Ji (機) — el momento exacto de transformación	Zhuangzi, <i>Zhuangzi</i> (4 aC)	Kripke — mundos posibles accesibles	Gates — umbral cualitativo entre estados, no condición binaria
Wu ji → Tai ji — de lo indiferenciado a la polaridad	Zhou Dunyi (1017-1073) <i>Taijitu shuo</i>	Husserl — ontología formal vs. regional	schema.ncl (vacío universal) → core.ncl (polaridad concreta del proyecto)
Relatividad perspectival	Zhuangzi — la rana del pozo	Wittgenstein tardío — juegos de lenguaje	Toda ontología es una conceptualización, no una captura de la realidad (Gruber)

4.4. Las cuatro correspondencias profundas

Yin/Yang ↔ Ontología/Reflection. La dualidad ya está en el diseño del protocolo. La ontología (core.ncl) captura lo que ES — invariantes, estructura, ser; capa yang: formal, dura, definida. La reflection captura lo que DEVIENE — operaciones, modos, movimiento; capa yin: fluida, procesual. Como en el símbolo: cada una contiene una semilla de la otra. La ontología describe estados **futuros** (desired_state) — contiene movimiento. La reflection puede **actualizar** la ontología vía ADRs — contiene forma. Separarlas es necesario porque tienen ritmos de cambio distintos; fundirlas colapsaría el yin en el yang.

Wu wei ↔ «No enforcement». Uno de los invariantes del protocolo: coherencia voluntaria que emerge de la adopción justificada, sin mecanismo de enforcement. Esto es wu wei aplicado a diseño de sistemas. No impones — creas las condiciones para que el alineamiento emerja naturalmente. Un sistema de tipos fuertes fuerza; una ontología operacional orienta. La diferencia es exactamente la diferencia entre imponer el Tao y fluir con él.

Tensiones ↔ Yin-Yang permanente. Las tensiones en un modelo operacional («formalización vs. fricción de adopción», «ontología vs. reflection») son explícitamente irresolubles. No se busca síntesis — se gestionan. Si eliminas la tensión entre formalización y adopción maximizando uno de los polos, el sistema muere: o es un schema perfecto que nadie adopta, o es caos operacional que todos adoptan pero nadie puede razonar sobre él.

Gates ↔ Ji (機). El concepto taoísta de ji es el punto de inflexión donde el potencial se actualiza — no un estado sino el umbral entre estados. Dinámico, cualitativo, contextual. Las gates en el modelo operacional son exactamente esto: no checkboxes — son condiciones que marcan el momento en que un estado puede transformarse en otro.

4.5. La crítica taoísta a la ontología clasificatoria

Los sistemas como OWL intentan eliminar la ambigüedad mediante clasificación precisa. El *Tao Te Ching* advierte: cuanto más preciso el nombre, más excluyes. La realidad vive en el espacio **entre** las categorías, no en las categorías mismas.

Esto explica por qué las ontologías biomédicas — las más maduras del campo — sufren continuamente con casos límite, dependencia del contexto y la necesidad de múltiples ejes de clasificación simultáneos. La vista taoísta diría: estáis intentando resolver una tensión constitutiva. La precisión y la cobertura son yin y yang — no se resuelven, se equilibran.

Zhuangzi va más lejos que Laozi. Toda distinción ontológica es perspectival. Esto no lleva al nihilismo sino a la **apropiada contextualidad**: la ontología correcta para un proyecto no es la más formalmente precisa sino la que el equipo puede sostener y evolucionar. Gruber llegó al mismo lugar en 1993 sin saberlo: una ontología es «una especificación de una conceptualización» — un punto de vista compartido, no una verdad metafísica.

5. DAGs — genealogía y función en ontología operacional

5.1. El problema que fuerza el grafo

Antes del grafo hay un problema: ¿cómo representar conocimiento estructurado de forma que se pueda razonar sobre él? La prosa no es consultable ni verificable. Las listas planas pierden las relaciones. Las jerarquías de árbol pierden los nodos con múltiples padres. El grafo emerge como la estructura mínima que captura relaciones arbitrarias entre conceptos.

Pero el grafo general tiene un problema: puede tener ciclos. Y los ciclos en ontología son epistemológicamente problemáticos.

5.2. Por qué la aciclicidad — el problema de la justificación circular

Dos tradiciones independientes llegan al mismo resultado:

Aristóteles (*Analíticos Posteriores*): una definición válida procede de género más diferencia específica. El género es más general que lo definido — la jerarquía es estrictamente descendente. Si A se define en términos de B y B en términos de A, ninguno está definido. La justificación circular es un defecto lógico.

Russell (1901): la paradoja del conjunto de todos los conjuntos que no se contienen a sí mismos. La solución en *Principia Mathematica* (1910, con Whitehead) es la **teoría de tipos**: los tipos forman una jerarquía estricta — un tipo de nivel N solo puede referirse a tipos de nivel N-1 o inferior. Esto es estructuralmente un DAG. La aciclicidad no es una restricción arbitraria — es la solución formal al colapso por autorreferencia.

La convergencia: en lógica, ontología formal y teoría de tipos, la aciclicidad es el requisito de que las definiciones sean **bien fundadas**. No puede haber descenso infinito ni circularidad.

5.3. Línea genealógica

Año	Autor / Hito	Contribución al DAG en ontología
384-322 aC	Aristóteles — <i>Analíticos Posteriores</i>	Definición bien fundada: jerarquía sin circularidad. Género → diferencia → especie.
1879	Frege — <i>Begriffsschrift</i>	Lógica de predicados: relaciones formales entre conceptos. Base para representar grafos de conocimiento.
1901-1910	Russell & Whitehead — <i>Principia Mathematica</i>	Teoría de tipos como DAG: aciclicidad como solución al colapso por autorreferencia.
1933	Tarski — «El concepto de verdad»	Verdad como correspondencia entre proposiciones y estructuras (grafos). Base de la validación formal.
1958	US Navy — PERT charts (programa Polaris)	DAG para planificación de proyectos: camino crítico como recorrido del grafo de dependencias temporales.

Año	Autor / Hito	Contribución al DAG en ontología
1969	Collins & Quillian — «Retrieval time from semantic memory»	Primera red semántica computacional: is-a como DAG. La topología del grafo predice tiempos de reacción humanos.
1974	Minsky — «A Framework for Representing Knowledge»	Frames con herencia múltiple: DAG sobre árbol. Primer sistema con nodos de múltiples padres.
1976	Feldman — Make	Build system como DAG de dependencias. Orden topológico para compilación incremental. Ciclos = error explícito.
1985	Brachman & Schmolze — KL-ONE	Description Logic: subsunción formal como DAG. Antecedente directo de OWL.
1988	Pearl — <i>Probabilistic Reasoning in Intelligent Systems</i>	Causalidad como DAG: la asimetría temporal hace la causalidad inherentemente acíclica.
2000	Pearl — <i>Causality</i>	Do-calculus: intervención como operación sobre el DAG causal. Separación entre correlación y causa.
2001	Berners-Lee et al. — Web Semántica	RDF/OWL: grafos de conocimiento como estándar industrial. DAG de subsunción en jerarquías de clases.
2005	Torvalds — Git	Historia de commits como DAG: irreversibilidad del tiempo en control de versiones. Merges = nodos con múltiples padres.

5.4. Las cuatro funciones del DAG en ontología operacional

① **Traversal determinista para análisis de impacto.** Si cambias un nodo, el DAG define exactamente qué nodos son alcanzables desde él (descendientes) y qué nodos lo alcanzan (ancestros). Esto es análisis de impacto: «si modifico este invariante, ¿qué tensiones, prácticas y gates se ven afectadas?» En un grafo cíclico esto es indecidible — el cierre transitivo de un nodo puede ser todo el grafo.

② **Orden topológico para ejecución.** Los modos operacionales son DAGs de pasos con dependencias. El ordenamiento topológico da el orden de ejecución válido sin necesidad de un scheduler externo. Si el paso B depende del paso A, A siempre precede a B en cualquier topological sort.

③ **Justificación bien fundada — sin circularidad epistémica.** El debate clásico en epistemología enfrenta fundacionalismo (el conocimiento descansa en fundamentos no circulares — DAG con nodos raíz) contra coherentismo (red de creencias mutuamente sustentadas — puede ser cíclica). La ontología operacional es fundacionalista por diseño: los invariantes son nodos raíz justificados por ADR explícito, no por otros nodos del grafo. Un ciclo significaría: «este invariante existe porque esta práctica existe, y esta práctica existe porque este invariante existe» — dogma circular, no arquitectura.

④ **Rollback con orden definido.** Deshacer cambios requiere un orden inverso bien definido. En un grafo cíclico no existe ese orden. El DAG garantiza que el rollback siempre tiene una secuencia válida.

5.5. La tensión con el Taoísmo — DAG vs. ciclo

El DAG es **yang** en su estructura: direccional, jerárquico, asimétrico, irreversible. Codifica la flecha del tiempo — las causas preceden a los efectos, las definiciones preceden a lo definido.

El yin-yang taoísta es **cíclico**: yin genera yang, yang genera yin. Las estaciones, el día y la noche, vida y muerte — todo retorna.

La resolución está en separar dos temporalidades:

- **Sincrónica** (snapshot): el DAG es válido dentro de un momento dado. La ontología en t_0 es un grafo acíclico.
- **Diacrónica** (evolución): la secuencia de snapshots puede tener patrones cíclicos. Lo que era tensión puede convertirse en invariante; lo que era práctica puede ser abandonado y redescubierto.

El DAG captura la **estructura del ser en un momento**. El ciclo taoísta captura el **proceso de transformación a través del tiempo**. Son complementarios, no contradictorios — exactamente como ontología y reflexión son complementarios en el protocolo. El **I Ching** — el libro de los cambios — es un sistema para razonar sobre las transformaciones de un estado a otro: 64 hexagramas como estados (nodos), transformaciones entre ellos como aristas. El sistema completo es un grafo. No un DAG — un grafo con ciclos, porque la realidad vuelve.

6. Física cuántica, computación cuántica y ontología operacional

6.1. El golpe más profundo — Aristóteles estaba equivocado en el nivel fundamental

La ontología aristotélica asume que las sustancias tienen propiedades definidas. La mecánica cuántica demuestra que, en el nivel fundamental, las propiedades no son definidas hasta que se miden. Un electrón no **tiene** posición — la posición emerge del acto de medición.

Esto no es ignorancia del observador sobre una propiedad preexistente. El **teorema de Kochen-Specker** (1967) lo prueba formalmente: es imposible asignar valores definidos a todos los observables cuánticos simultáneamente, incluso antes de la medición. La indeterminación es **ontológica**, no epistémica. El suelo sobre el que se construyeron 2.300 años de ontología occidental se mueve.

6.2. Las tres conexiones genuinas

6.2.1. Complementariedad de Bohr \leftrightarrow Yin-Yang y la dualidad Ontología/Reflection

Niels Bohr (1885-1962) formuló el **principio de complementariedad**: onda y partícula son dos descripciones mutuamente excluyentes pero igualmente necesarias del mismo fenómeno. No puedes observar ambas simultáneamente — el aparato de medición determina qué aspecto revelas.

Bohr lo conectó explícitamente con el Taoísmo: cuando fue ennoblecido en 1947 adoptó el símbolo yin-yang como parte de su escudo de armas con la inscripción *Contraria sunt complementa* — los contrarios son complementarios.

La dualidad ontología/reflection del modelo operacional tiene la misma estructura. El «aparato de medición» — qué preguntas haces al sistema — determina qué aspecto revelas. La capa que uses (core.ncl o los modos de reflection) determina qué ves. Ninguna es más verdadera — son complementarias.

6.2.2. Contextualidad cuántica \leftrightarrow Ontología perspectival

El teorema de Kochen-Specker prueba algo más radical que la incertidumbre de Heisenberg: el resultado de medir un observable cuántico depende de **qué otros observables se miden simultáneamente**. Las propiedades son **contextuales** — no existen independientemente del contexto de medición.

Esto convierte en hecho físico lo que tres tradiciones de esta discusión habían sostenido filosóficamente:

- **Zhuangzi**: toda distinción ontológica es perspectival
- **Wittgenstein tardío**: el significado depende del juego de lenguaje

- **Gruber:** una ontología es una especificación de una conceptualización, no una captura de la realidad

Para la ontología operacional: el «estado» de un proyecto es siempre relativo al actor y al contexto de consulta. Un agente de IA y un desarrollador humano consultando el mismo `core.ncl` están haciendo mediciones con aparatos distintos. Lo que ven es complementario, no idéntico.

6.2.3. El problema del observador ↔ El trasfondo que no puede formalizarse

En mecánica cuántica, la medición colapsa la función de onda — el observador es parte del sistema. No hay *view from nowhere* (Nagel, 1986). Todo conocimiento es conocimiento **desde** una posición.

Esto es exactamente lo que Heidegger llamó el trasfondo pre-temático — la condición de posibilidad de cualquier formalización que ella misma no puede ser formalizada. Y lo que el *Tao Te Ching* afirma: el Tao que puede nombrarse no es el Tao eterno.

La ontología operacional que crees que describes objetivamente está siendo constituida por el acto de escribirla. Esto no es un defecto del modelo — es un límite estructural de toda representación.

6.3. La interpretación más relevante — Mecánica Cuántica Relacional

Carlo Rovelli propuso en 1996 (*Relational Quantum Mechanics*, International Journal of Theoretical Physics) que no existen estados absolutos — solo estados **relativos a otros sistemas**. El estado de una partícula no es una propiedad intrínseca — es una propiedad relacional respecto al sistema con el que interactúa.

Esto es la ontología más radical posible y conecta directamente con Ladyman y Ross (realismo estructural): no hay propiedades, solo relaciones. Lo real son las estructuras relacionales, no las entidades.

Para el modelo operacional: el «estado» de un proyecto (`current_state` en `state.ncl`) no es una propiedad intrínseca del proyecto — es relacional respecto al actor que consulta, los objetivos perseguidos, el momento del ciclo de vida. Dos actores con marcos de referencia distintos pueden tener mediciones del estado coherentes pero distintas. No hay contradicción — hay contextualidad relacional.

6.4. La computación cuántica — honestidad sobre sus límites

Aspecto	Situación real (2026)	Relevancia para ontología operacional
Tamaño de grafos	La ventaja cuántica aparece a escala masiva	Los grafos de ontología de proyecto son pequeños — sin ventaja cuántica
Algoritmos relevantes	Grover: $O(\sqrt{N})$ búsqueda. Shor: factorización polinómica	No mapean a consultas típicas sobre ontologías de proyecto
Decoherencia	Cualquier interacción colapsa la superposición	El uso práctico requiere aislamiento que contradice operación continua

Aspecto	Situación real (2026)	Relevancia para ontología operacional
Redes semánticas cuánticas	Investigación activa — relaciones con amplitudes de probabilidad	Captura gradualidad que OWL binario pierde — relevante a mediano plazo
Bayesian networks cuánticas	Extensión de Pearl con amplitudes complejas	Modelar dependencias con comportamiento emergente no reducible
LLMs y espacios de Hilbert	Los embeddings son productos internos en espacio vectorial de alta dimensión	Directamente relevante: la misma matemática cuántica, aplicada a semántica

6.5. La conexión no obvia — LLMs como geometría cuántica aplicada

Los Large Language Models operan en espacios vectoriales de alta dimensión donde el significado es relacional y contextual. No es metáfora cuántica — es la misma matemática de espacios de Hilbert que usa la mecánica cuántica, aplicada a representación semántica.

La ontología operacional (core.ncl, invariantes, tensiones) es, en este contexto, una **base** en ese espacio — un conjunto de vectores que fijan el marco de referencia en el que el agente opera. Inyectar el contexto ontológico al inicio de una sesión es, en la geometría de estos modelos, fijar el marco de referencia antes de que el agente empiece a computar.

La física cuántica importa aquí no porque vayamos a correr ontologías en computadores cuánticos — sino porque demuestra que las asunciones ontológicas clásicas (propiedades definidas, estados absolutos, observador separado del sistema) son físicamente incorrectas en el nivel fundamental. La ontología operacional es contextual, relacional y perspectival **por diseño correcto**, no por limitación.

7. Por qué es relevante en software e infraestructuras digitales

7.1. El problema real no es técnico — es epistémico

El software moderno tiene un déficit estructural que no se resuelve con mejores herramientas:

```
Lo que el sistema hace ≠ Lo que el equipo cree que hace
Lo que la infra tiene desplegado ≠ Lo que Terraform declara
Lo que el ADR decidió ≠ Lo que el código implementa
Lo que el agente asume ≠ Lo que el contexto real requiere
```

Cada una de esas brechas es silenciosa, acumulativa, y solo se descubre cuando falla en producción. No son fallos de las personas — son consecuencias inevitables de operar un sistema complejo sin un modelo del propio sistema.

7.2. Lo que la ontología operacional hace que nada más hace

Documentación tradicional	Ontología operacional
Prosa — no consultable	Grafo tipado — consultable con queries estructuradas
Estática — se desactualiza sin señal	Verificada en cada commit mediante typecheck
Describe lo que hubo	Formaliza lo que ES y lo que debe ser
Solo legible por humanos	Legible por humanos, agentes de IA y pipelines de CI
Captura el qué	Captura el qué, el por qué, y lo que no puede cambiar
Nadie la actualiza bajo presión	El proceso la actualiza — es el path, no documentación paralela

Formaliza lo que no puede romperse. Los invariantes no son comentarios — son constraints tipadas que el sistema verifica antes del runtime, no después.

Captura las tensiones activas. «Coste vs. disponibilidad», «velocidad de despliegue vs. estabilidad» viven en la ontología, no en la cabeza del miembro más veterano del equipo. Cuando ese alguien se va, los trade-offs permanecen.

Da al agente de IA el contexto que no puede inferir del código. Un agente puede leer todo el código de un sistema y no saber que hay un acuerdo que impide escalar horizontalmente cierto servicio, o que la decisión de usar gRPC interno fue consecuencia de una tensión específica de latencia que no está en ningún archivo. La ontología operacional hace eso explícito y machine-readable.

7.3. Por qué los DAGs — especificidad técnica

Los DAGs no son una elección estética. Son la única estructura que garantiza cuatro propiedades simultáneamente en sistemas operacionales:

Análisis de impacto decidible. Antes de cambiar un invariante o un nodo de infra — recorre el DAG y sabes exactamente qué se ve afectado. Concreto: en Terraform, el grafo

de recursos es un DAG. `terraform plan` recorre ese grafo para determinar qué debe crearse, modificarse o destruirse en qué orden. Si hubiera ciclos, el plan sería indecidible.

Orden topológico para ejecución fiable. Cualquier operación con dependencias — despliegue, migración, rollback, onboarding — tiene un orden correcto. El DAG lo da sin necesidad de un coordinador central. Este es el principio detrás de Make (1976), Bazel, Gradle, y cualquier build system no trivial.

Justificación sin circularidad. Si el invariante A existe porque la práctica B lo requiere, y la práctica B existe porque el invariante A lo define — tienes dogma, no arquitectura. El DAG hace esa circularidad visible e inaceptable.

Rollback determinista. El inverso topológico del DAG es siempre computable. Sin DAG, el rollback es un proceso manual que depende de que alguien recuerde el orden correcto.

7.4. Por qué la reflection — la ontología sin operación es un museo

La ontología captura el **ser**. Pero los sistemas de software no solo son — **operan**. La reflection es la capa que hace la ontología ejecutable:

Modos como DAGs operacionales. Un procedimiento sin DAG es una lista de pasos que alguien puede ejecutar en cualquier orden, saltar, o ignorar. Un modo como DAG tiene dependencias explícitas — el sistema puede verificar que se ejecutó correctamente antes de avanzar.

ADRs con constraints tipadas. Una decisión sin constraints es prosa. Con constraints tipadas es un contrato que el CI puede verificar en cada commit. La pregunta «¿estamos violando alguna decisión arquitectural?» pasa de ser reflexión manual a ser comprobación automática.

Detección de drift. La diferencia entre el estado sellado (lo que declaraste que era verdad) y el estado actual (lo que realmente hay desplegado) es medible si y solo si formalizaste el estado sellado. Sin eso, la pregunta «¿ha derivado esto?» no tiene respuesta objetiva.

7.5. Las tres presiones que hacen esto urgente ahora

Estos conceptos no son nuevos. Lo nuevo es que tres presiones convergen simultáneamente haciendo que la ausencia del modelo operacional sea costosa de una forma que antes era tolerable:

Presión 1: Los sistemas superan la capacidad de comprensión individual. Un monolito de 2005 podía vivir en la cabeza de dos personas. Un sistema cloud-native de 2026 con decenas de microservicios, múltiples regiones, pipelines automatizados y SLOs por servicio no cabe en ninguna cabeza. El modelo mental individual ha dejado de ser suficiente como mecanismo de coordinación.

Presión 2: Los agentes de IA tienen acceso de escritura. Hasta hace poco, una herramienta de IA que generaba código incorrecto producía texto que un humano revisaba antes de ejecutar. Hoy los agentes pueden abrir PRs, modificar configuración, ejecutar

Terraform, y desplegar. El ciclo humano-en-el-loop se comprime. Un agente sin contexto ontológico en ese entorno no es un asistente — es un actor autónomo con comprensión incompleta del sistema que modifica.

Presión 3: Los equipos son multi-actor y distribuidos. Humanos en zonas horarias distintas, agentes de IA con distintos contextos de sesión, pipelines de CI ejecutando en paralelo — todos escribiendo sobre el mismo sistema sin coordinador central. Sin un protocolo compartido y machine-readable para «qué puede cambiarse, bajo qué condiciones, con qué justificación», la coordinación depende de que todo el mundo lea el mismo Confluence actualizado. Eso nunca funcionó.

7.6. La síntesis

Lo que la ontología operacional, reflection y DAGs aportan en conjunto es algo que ninguna de sus partes aporta sola:

Un sistema que sabe lo que es, puede verificar que sigue siendo lo que dice ser, y puede comunicarlo a cualquier actor — humano, agente o CI — de forma estructurada y sin ambigüedad.

Eso no es documentación. Es un modelo operacional vivo.

La ausencia de ese modelo no es un problema de disciplina del equipo. Es una propiedad emergente de sistemas complejos operados sin representación explícita. El drift, la amnesia de decisiones, la ilusión de control en infraestructura declarativa — no son fallos de las personas. Son consecuencias inevitables.

La ontología operacional es la respuesta a la pregunta: **¿qué estructura mínima necesita un sistema para conocerse a sí mismo de forma que ese conocimiento pueda ser usado, verificado y compartido?**

8. Referencias y lecturas recomendadas

8.1. Filosofía clásica y ontología

Aristóteles. *Metafísica*. Traducción de Tomás Calvo Martínez. Madrid: Gredos, 1994. [330 aC]

El libro fundamental para la distinción esencia/accidente, potencia/acto y las categorías. El libro IX sobre potencia es la base del modelo estado-actual/estado-deseado.

Aristóteles. *Analíticos Posteriores*. Traducción de Miguel Candel Sanmartín. Madrid: Gredos, 1988. [350 aC]

Formaliza el requisito de definiciones bien fundadas — antecedente directo de la aciclicidad en DAGs.

Laozi. *Tao Te Ching*. Traducción y comentarios de Stephen Mitchell. Nueva York: Harper Perennial, 1992. [siglo VI-IV aC]

Texto fundacional del Taoísmo. Los primeros y últimos capítulos son los más relevantes para la relación con ontología operacional.

Zhuangzi. *Zhuangzi: The Complete Writings*. Traducción de Brook Ziporyn. Indianapolis: Hackett, 2020. [siglo IV aC]

Especialmente los capítulos internos (1-7). La relatividad perspectival de Zhuangzi como crítica de toda ontología clasificatoria.

Frege, G. *Begriffsschrift*. Halle: Nebert, 1879.

Primera formulación de la lógica de predicados. El antecedente formal de toda representación de conocimiento estructurado.

Russell, B. & Whitehead, A.N. *Principia Mathematica*. Cambridge: Cambridge University Press, 1910-1913.

La teoría de tipos como DAG para evitar la autorreferencia circular. El capítulo introductorio explica el principio del círculo vicioso.

Wittgenstein, L. *Tractatus Logico-Philosophicus*. Londres: Kegan Paul, 1922.

«Los límites de mi lenguaje son los límites de mi mundo.» El fundamento de por qué el schema Nickel define la frontera de lo representable en el sistema.

Wittgenstein, L. *Investigaciones filosóficas*. Oxford: Blackwell, 1953.

Juegos de lenguaje como práctica. El significado como uso — los modos operacionales como juegos de lenguaje de un proyecto.

Hegel, G.W.F. *Fenomenología del Espíritu*. Traducción de Manuel Jiménez Redondo. Valencia: Pre-Textos, 2006. [1807]

Dialéctica tesis-antítesis-síntesis. Contrástese con el yin-yang: la diferencia entre síntesis y tensión gestionada permanentemente.

Husserl, E. *Ideas relativas a una fenomenología pura y una filosofía fenomenológica*. México: FCE, 1993. [1913]

La distinción ontología formal/regional — estructura vacía universal vs. conceptos de dominio específico.

Whitehead, A.N. *Process and Reality*. Nueva York: Free Press, 1978. [1929]

La realidad como proceso, no sustancia. El fundamento filosófico de la capa reflection como devenir distinto del ser ontológico.

Kripke, S. *Naming and Necessity*. Cambridge: Harvard University Press, 1980.

Propiedades esenciales rígidas y semántica de mundos posibles. El fundamento modal de los invariantes operacionales.

Searle, J. *The Construction of Social Reality*. Nueva York: Free Press, 1995.

Hechos institucionales como fundamento de las decisiones declaradas. El ADR como acto de habla colectivo que crea realidad operacional.

Parfit, D. *Reasons and Persons*. Oxford: Oxford University Press, 1984.

Identidad a través del tiempo. La pregunta «¿es el mismo proyecto?» cuando sus invariantes cambian vía ADR.

Ladyman, J. & Ross, D. *Every Thing Must Go: Metaphysics Naturalized*. Oxford: Oxford University Press, 2007.

Realismo estructural: las relaciones son más reales que las entidades. El DAG como estructura fundamental, los nodos como secundarios.

Quine, W.V.O. «On What There Is.» *Review of Metaphysics*, 2(5), 1948.

El concepto de compromiso ontológico: toda teoría asume la existencia de ciertas entidades. Todo schema asume una ontología aunque no lo declare.

Tarski, A. «The Concept of Truth in Formalized Languages.» En *Logic, Semantics, Metamathematics*. Oxford: Clarendon Press, 1956. [1933]

Verdad como correspondencia. El fundamento de la validación formal — nickel typecheck como verificación tarsquiana.

8.2. Inteligencia artificial y representación del conocimiento

Minsky, M. «A Framework for Representing Knowledge.» MIT AI Lab Memo 306, 1974.

Los frames como primera estructura computacional análoga a la sustancia aristotélica con propiedades heredables.

Collins, A.M. & Quillian, M.R. «Retrieval time from semantic memory.» *Journal of Verbal Learning and Verbal Behavior*, 8(2), 240-247, 1969.

Primera red semántica computacional. Demuestra que la topología del grafo de conocimiento predice el comportamiento cognitivo.

Brachman, R.J. & Schmolze, J.G. «An overview of the KL-ONE knowledge representation system.» *Cognitive Science*, 9(2), 171-216, 1985.

Description Logic: la subsunción como relación formal que genera el DAG de conceptos. Antecedente teórico de OWL.

Gruber, T. «A translation approach to portable ontology specifications.» *Knowledge Acquisition*, 5(2), 199-220, 1993.

La definición canónica: ontología como especificación de una conceptualización. Referencia obligatoria para cualquier trabajo en ontología de software.

Lenat, D.B. & Guha, R.V. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Reading: Addison-Wesley, 1990.

El proyecto más ambicioso de ontología operacional. Sus fracasos son tan informativos como sus logros.

Pearl, J. *Probabilistic Reasoning in Intelligent Systems*. San Mateo: Morgan Kaufmann, 1988.

Bayesian networks como DAGs causales. La formalización de por qué la causalidad es inherentemente acíclica.

Pearl, J. *Causality: Models, Reasoning, and Inference*. Cambridge: Cambridge University Press, 2000.

El do-calculus. Distinción entre correlación y causa como operaciones sobre el DAG. Segunda edición (2009) incluye el modelo causal completo.

Guarino, N. (ed.) *Formal Ontology in Information Systems*. Amsterdam: IOS Press, 1998. Actas de FOIS'98. La conferencia fundacional del campo. Especialmente el artículo introductorio de Guarino sobre ontología formal vs. de dominio.

Berners-Lee, T., Hendler, J. & Lassila, O. «The Semantic Web.» *Scientific American*, 284(5), 34-43, 2001.

El artículo que popularizó las ontologías en ingeniería de software. La visión de la web como grafo de conocimiento estructurado.

8.3. Ingeniería de software y praxis

Nygard, M. «Documenting Architecture Decisions.» Cognitect Blog, 2011. Disponible en: cognitect.com/blog

La propuesta original de ADRs como documentación ligera de decisiones. El precursor directo de los ADRs con constraints tipadas.

Brown, S. *Software Architecture for Developers*. Leanpub, 2014.

El modelo C4 como alternativa a los diagramas UML. Introduce la idea de múltiples niveles de abstracción para describir un sistema.

Hohpe, G. & Woolf, B. *Enterprise Integration Patterns*. Boston: Addison-Wesley, 2003. Patrones como vocabulario compartido — la idea de que nombrar los patrones operacionales es un acto ontológico.

8.4. Perspectivas adicionales

Zhou Dunyi. *Taijitu shuo* (Explicación del Diagrama del Supremo Último). [1017-1073]

La cosmología taoísta sistematizada: wu ji → tai ji → yin-yang. La transición de lo indiferenciado a la polaridad como modelo de emergencia estructural.

Heidegger, M. *Ser y Tiempo*. Traducción de Jorge Eduardo Rivera. Santiago: Editorial Universitaria, 1997. [1927]

El trasfondo pre-temático que ninguna ontología captura completamente. La ausencia notable en todos los modelos operacionales de software.

Heráclito. *Fragmentos*. Edición de Rodolfo Mondolfo. Buenos Aires: Losada, 2007. [500 aC]

El único pensador presocrático estructuralmente equivalente al yin-yang. El fragmento DK22B60 — «el camino hacia arriba y hacia abajo son el mismo» — es el punto de partida.

Bateson, G. *Steps to an Ecology of Mind*. Nueva York: Ballantine, 1972.

La diferencia que hace una diferencia. Epistemología sistémica que conecta Taoísmo, cibernética y representación del conocimiento. Perspectiva poco citada en CS pero fundamentalmente relevante.

Winograd, T. & Flores, F. *Understanding Computers and Cognition*. Norwood: Ablex, 1986.

La crítica más influyente a la IA simbólica desde Heidegger. Argumenta que el trasfondo de comprensión no puede formalizarse — límite estructural de toda ontología operacional.

8.5. Física cuántica y fundamentos

Bohr, N. «Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?» *Physical Review*, 48, 696-702, 1935.

La respuesta de Bohr al artículo EPR de Einstein-Podolsky-Rosen. Formula el principio de complementariedad como posición filosófica irrenunciable.

Bohr, N. «Discussion with Einstein on Epistemological Problems in Atomic Physics.» En Schilpp (ed.), *Albert Einstein: Philosopher-Scientist*. Evanston: Library of Living Philosophers, 1949.

El debate más importante de la física del siglo XX sobre interpretación cuántica. Bohr y Einstein representan dos ontologías irreconciliables.

Heisenberg, W. *Physics and Philosophy: The Revolution in Modern Science*. Nueva York: Harper & Row, 1958.

La reflexión filosófica más accesible de uno de los fundadores. El capítulo sobre el papel del lenguaje en física cuántica conecta directamente con Wittgenstein.

Schrödinger, E. «Die gegenwärtige Situation in der Quantenmechanik.» *Naturwissenschaften*, 23(48-49-50), 1935.

El artículo del gato. Introduce el concepto de entrelazamiento (*Verschränkung*) y demuestra que la superposición cuántica no puede interpretarse clásicamente.

Kochen, S. & Specker, E.P. «The Problem of Hidden Variables in Quantum Mechanics.» *Journal of Mathematics and Mechanics*, 17(1), 59-87, 1967.

Prueba formal de que es imposible asignar valores definidos a todos los observables cuánticos simultáneamente. La contextualidad como hecho matemático, no filosófico.

Bell, J.S. *Speakable and Unspeakable in Quantum Mechanics*. Cambridge: Cambridge University Press, 1987.

Colección de ensayos del autor del teorema de Bell (1964). El capítulo «Against measurement» es una crítica filosófica fundamental al vocabulario de la física cuántica.

Aspect, A., Grangier, P. & Roger, G. «Experimental Realization of Einstein-Podolsky-Rosen-Bohm Gedankenexperiment.» *Physical Review Letters*, 49(2), 91-94, 1982.

La confirmación experimental de que las desigualdades de Bell se violan. Prueba que la naturaleza es no-local — no hay variables ocultas locales.

Everett, H. «Relative State Formulation of Quantum Mechanics.» *Reviews of Modern Physics*, 29(3), 454-462, 1957.

La interpretación de muchos mundos. Relevante porque ofrece una ontología donde todos los estados posibles son reales — el antípoda de la ontología operacional que colapsa posibilidades en decisiones.

Rovelli, C. «Relational Quantum Mechanics.» *International Journal of Theoretical Physics*, 35(8), 1637-1678, 1996.

No existen estados absolutos — solo estados relativos a otros sistemas. La ontología más radical de la lista y la más relevante para entender por qué el estado de un proyecto es siempre relacional.

Rovelli, C. *Helgoland*. Milano: Adelphi, 2020.

La versión divulgativa de la mecánica cuántica relacional. Accesible y filosóficamente riguroso. El capítulo sobre relaciones y el budismo es relevante para la conexión con el Taoísmo.

Nagel, T. *The View from Nowhere*. Oxford: Oxford University Press, 1986.

El argumento de que toda objetividad requiere un punto de vista. En tensión directa con el problema del observador cuántico y la ontología perspectival de Zhuangzi.

Nielsen, M.A. & Chuang, I.L. *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press, 2000.

El texto estándar de computación cuántica. Los capítulos 2-3 sobre el formalismo matemático (espacios de Hilbert, operadores, entrelazamiento) son la base para entender la conexión con LLMs.

Deutsch, D. *The Fabric of Reality*. Londres: Allen Lane, 1997.

La defensa más ambiciosa de la interpretación de muchos mundos. El capítulo sobre epistemología cuántica conecta con la cuestión de qué puede conocerse sobre un sistema complejo.

8.6. Ingeniería de software e infraestructura

Evans, E. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston: Addison-Wesley, 2003.

El lenguaje ubicuo como ontología operacional informal. El bounded context es estructuralmente equivalente a una ontología regional de Husserl aplicada a dominios de negocio.

Kleppmann, M. *Designing Data-Intensive Applications*. Sebastopol: O'Reilly, 2017.

El capítulo sobre modelos de datos es el más relevante: la elección de modelo (relacional, documental, grafo) es una decisión ontológica — define qué relaciones pueden existir.

Morris, K. *Infrastructure as Code*. Sebastopol: O'Reilly, 2016.

Formaliza la declaratividad en infraestructura. El capítulo sobre principios hace explícita la distinción entre estado deseado y estado actual — el núcleo del modelo state.ncl.

Humble, J. & Farley, D. *Continuous Delivery*. Boston: Addison-Wesley, 2010.

El pipeline de despliegue como DAG de verificaciones. La idea de que cada etapa del pipeline es un gate que el sistema debe cruzar formalmente.

Kim, G., Humble, J., Debois, P. & Willis, J. *The DevOps Handbook*. Portland: IT Revolution Press, 2016.

El flujo de trabajo DevOps como protocolo operacional. La sección sobre feedback loops es estructuralmente equivalente al ciclo ontología/reflection.

Dehghani, Z. «How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh.» Martin Fowler Blog, 2019.

Data mesh como ontología operacional distribuida para datos. Cada dominio es responsable de su propia ontología de datos — la aplicación más cercana al modelo ontorep en infraestructura de datos.

Fowler, M. «Who Needs an Architect?» *IEEE Software*, 20(5), 11-13, 2003.

Define arquitectura como «las decisiones que son difíciles de cambiar» — equivalente exacto de los invariantes en ontología operacional.

Richardson, C. *Microservices Patterns*. Shelter Island: Manning, 2018.

El patrón Saga como DAG de transacciones distribuidas. Cada paso del saga tiene depends_on implícito — el DAG operacional aplicado a consistencia eventual. La crítica más influyente a la IA simbólica desde

Heidegger. Argumenta que el trasfondo de comprensión no puede formalizarse — límite de toda ontología operacional.