Software Efímero

Nueva forma de software

Tomás de Camino Beck,Ph.D
Universidad CENFOTEC

Nota. Esta presentación se basa en el preprint publicado: de-Camino-Beck, T.; Nuñez-Corrales, S.; Naranjo-Zeledon, L. C.; y Trejos-Zelaya, I. (2026). *Ephemeral Software: a New Software Paradigm* (v1). Zenodo. DOI: 10.5281/zenodo.18727596.

00

Algunas observaciones

≫ Un panorama en transformación

- La IA redefinió lo que esperamos de las computadoras.
- Pasamos de interfaces WIMP a Conversacionales
- Poder del lenguaje natural
- IA Generativa no es herramienta, es amplificación

≫ Un asunto de lenguajes

- La especificación formal no captura intención.

≫ Un asunto de lenguajes

- La especificación formal no captura intención.
- El lenguaje natural incorpora intención y contexto antes de formalizar.

≫Un asunto de lenguajes

- La especificación formal no captura intención.
- El lenguaje natural incorpora intención y contexto antes de formalizar.
- Formalizar elimina algo de ambigüedad de intención humana, pero pierde información.(Characteristica Universalis)

≫Un asunto de lenguajes

- La especificación formal no captura intención.
- El lenguaje natural incorpora intención y contexto antes de formalizar.
- Formalizar elimina algo de ambigüedad de intención humana, pero pierde información.(Characteristica Universalis)
- LLMs muestran computación a partir de intención.

≫ Un asunto de lenguajes

- La especificación formal no captura intención.
- El lenguaje natural incorpora intención y contexto antes de formalizar.
- Formalizar elimina algo de ambigüedad de intención humana, pero pierde información.(Characteristica Universalis)
- LLMs muestran computación a partir de intención.
- Cambio fundamental, pasar de procedimientos a manifestar la intención y verificar.

01

Introducción

≫Software tradicional

SUPUESTO CLÁSICO

- El código es un **activo durable**.
- Se almacena, versiona y mantiene.
- Favorece continuidad y reutilización.

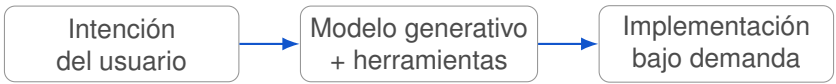
EFECTO ACUMULATIVO

- Más historia que intención.
- Deuda y rigidez arquitectónica.
- Deriva semántica y costo de gobierno.

El problema

Algunas soluciones caducan antes de que mantener su código produzca valor.

≫Sistemas generativos



- Los LLM sintetizan código, diseño y arquitectura.
- Conservar una realización pasa a ser una **decisión**.
- Persisten objetivos, restricciones e invariantes.

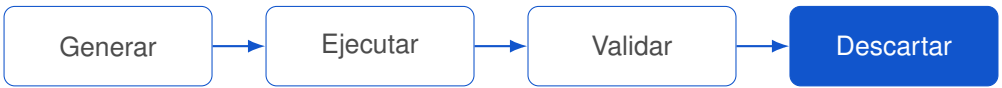
La intención ejecutable pasa a ser la unidad de diseño y responsabilidad.

El software pasa a ser efímero...

≫ Definición

Software efímero es una realización computacional generada bajo demanda para una intención y un contexto específicos, validada durante su uso y descartada después.

≫Software efímero



. . . centrado en intención

02

Paradigmas centrados en intención o intuición

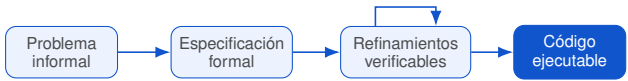
≫Cuatro paradigmas

End-User Software Engineering (EUSE)

≫Cuatro paradigmas

End-User Software Engineering (EUSE)

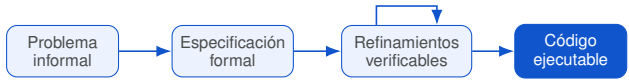
Programación basada en intuición



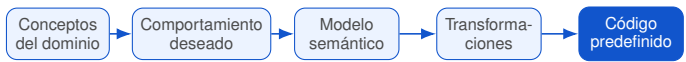
≫Cuatro paradigmas

End-User Software Engineering (EUSE)

Programación basada en intuición



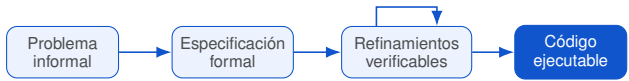
Programación intencional



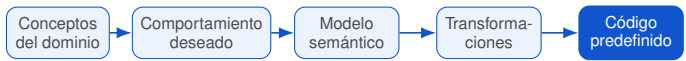
≫Cuatro paradigmas

End-User Software Engineering (EUSE)

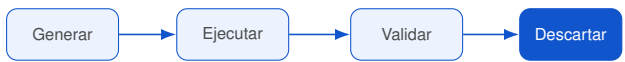
Programación basada en intuición



Programación intencional



Software efímero



≫Comparación

Dimensión	EUSE	Basada en intuición	Intencional	Software efímero
Actor central	Usuario final	Desarrollador	Diseñador de dominio	Usuario + IA
Intención	Implícita	Implícita, evolutiva	Explícita y estructurada	Inmediata y situada
Relación con código	Código accesible al usuario	Intuición refinada como código	Código generado desde intención	Código transitorio
Rigor	Baja–media	Progresiva	Alta	Según el contexto
Duración	Persistente	Persistente	Persistente	Efímera
Verificación	En uso	En desarrollo	Antes y durante	Al generar y usar
Aporte	Empoderamiento	Realismo cognitivo	Separar intención y código	Ejecutar intención situada

03

Software efímero

Definición un poco más formal

≫ Definición formal

$$\mathcal{S}_e = (I, C, G, E)$$

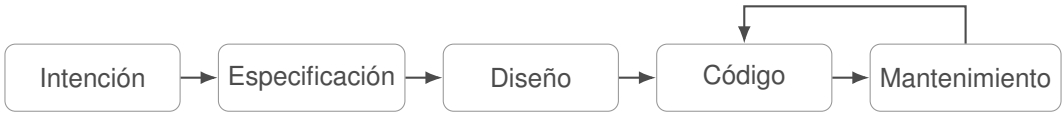
- I intención explícita o implícita;
- C restricciones técnicas, temporales, organizacionales o éticas;
- G mecanismo generativo;
- E entorno de ejecución.

Inversión ontológica

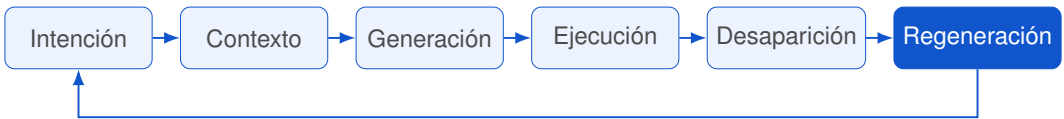
Código y arquitectura pasan de activos primarios a resultados de $G(I, C)$ en E.

≫ Dos ciclos de vida

SOFTWARE TRADICIONAL



SOFTWARE EFÍMERO



≫ ¿Continuidad?

SOFTWARE TRADICIONAL

La continuidad depende de conservar y modificar la implementación.

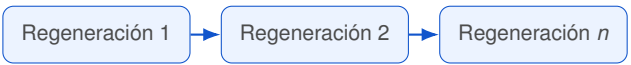
SOFTWARE EFÍMERO

La continuidad reside en poder generar otra realización desde la intención.

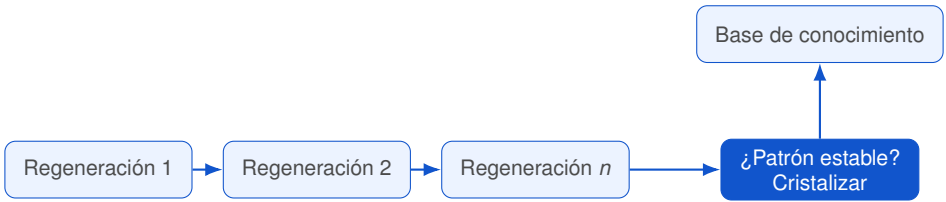
≫ Regeneración

Regeneración 1

≫ Regeneración

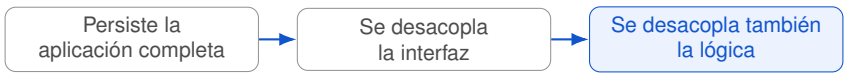


≫ Regeneración



≫ Persistencia

Arquitectura	¿Qué permanece?
Monolítica (tradicional)	Interfaz de usuario + lógica + datos
Headless	Lógica + datos
Software efímero	Intención + políticas + datos; la lógica se genera cuando se necesita



Evolucionar reduce aquello que debe conservarse como software.

04

Intención ejecutable

Una fuente de verdad computable y gobernable.

≫Intención computacional

Definición operacional

*/** permite generar, ejecutar y evaluar comportamiento sin fijar una implementación.

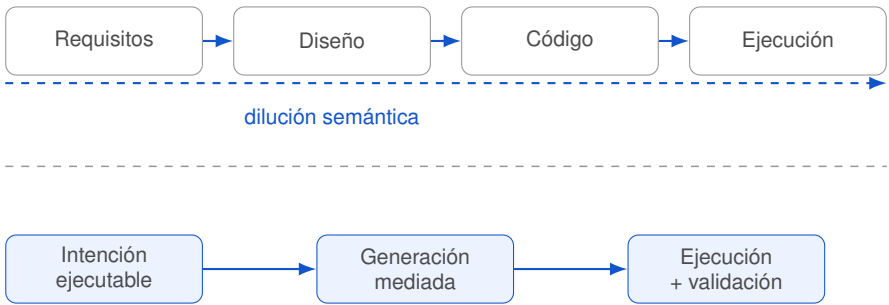
Objetivos
Resultados deseados y aceptables.

Límites
Restringe el espacio de implementaciones.

Interpretabilidad
Puede ser procesada por un mecanismo generativo.

Ejecutar es realizar una intención mediante generación mediada.

≫ Ejecución


Requisitos
Diseño
Código
Ejecución
dilución semántica
Intención ejecutable
Generación mediada
Ejecución + validación

≫Conserva / Genera

LO QUE SE VERSIONA

- Metas y restricciones
- Reglas de transformación
- Criterios de satisfacción
- Contexto y trazabilidad

LO QUE SE GENERA

- Lenguaje y framework
- Arquitectura y datos
- Estrategia algorítmica
- Despliegue compatible

Se estabiliza el significado; la realización queda abierta.

≫ Artefactos de valor

	Artefacto privilegiado	Rasgo distintivo
Declarativo	Programa persistente	El lenguaje fija la estrategia de ejecución
Programación intencional	Intención estructurada	Genera representaciones estables
Intención ejecutable	Intención + contexto + validación	Repetible sin continuidad de implementación

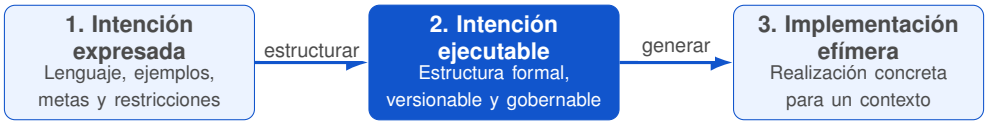
La diferencia está en qué se trata como fuente de verdad.

05

Operacionalización

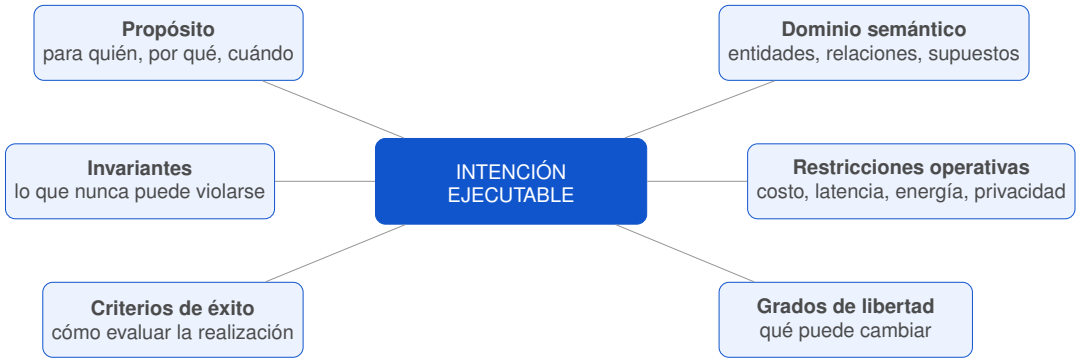
De expresión humana a realización validada.

≫ Niveles de representación



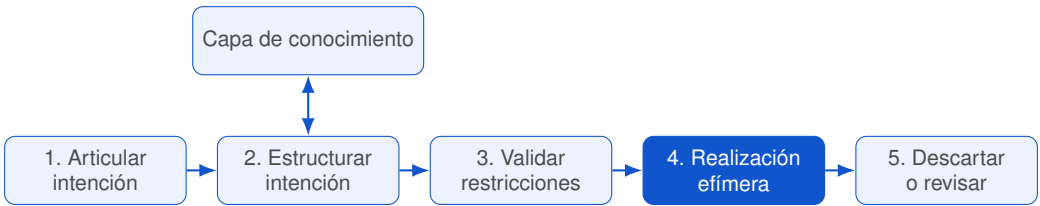
- Expresión humana: parcial y situada.
- Capa ejecutable: explícita y gobernable.
- Implementación: concreta y descartable.

≫Intención ejecutable



La plataforma estructura y verifica estos elementos antes de generar.

≫ Pipeline software efímero



06

Ejemplos

y potenciales aplicaciones

≫ Ejemplo mínimo

Construcción de un perceptrón para enseñanza

≫ Representaciones

Representación	Capacidad de regenerar la misma página	Capacidad de generar nuevas versiones válidas
Página web	Muy alta	Muy baja
Prompt técnico exacto	Muy alta	Baja
Prompt natural exacto	Alta	Media
Prompt intencional	Media o baja	Muy alta

≫ Banca (conepto)

PROCEDIMIENTO TRADICIONAL

$$P = \{e_1, e_2, e_3, \ldots, e_n\}$$

Una secuencia previamente programada de operaciones.

≫ Banca (conepto)

SOFTWARE EFÍMERO

$$P_t = G(I, C_t)$$

donde I expresa qué debe conseguir el banco, C_t contiene políticas, regulación, estado del cliente, riesgo y demás condiciones vigentes, y G construye el procedimiento apropiado para ese momento.

≫ERP/CRM

Aspecto	ERP/CRM tradicional	Software efímero
Activo central	Código y aplicación	Intención, conocimiento y reglas
Desarrollo y arquitectura	Pantallas, reportes, módulos y servicios persistentes	Dominio, políticas, primitivas y aplicaciones generadas
Persistencia	Aplicación permanente	Datos y conocimiento permanentes
Mantenimiento y crecimiento	Modificar código, interfaces y agregar módulos	Actualizar conocimiento, reglas y capacidades
Interfaz e interacción	Interfaz fija; menús y módulos	Interfaz situada; conversaciones y objetivos
Procesos y productos	Programados y predefinidos	Sintetizados y compuestos para la situación
Personalización e IA	Configuración; IA como asistente	Generación contextual; IA como motor de compilación
Ciclo de vida	Interfaces durables y nuevas versiones	Interfaces temporales y regeneración continua

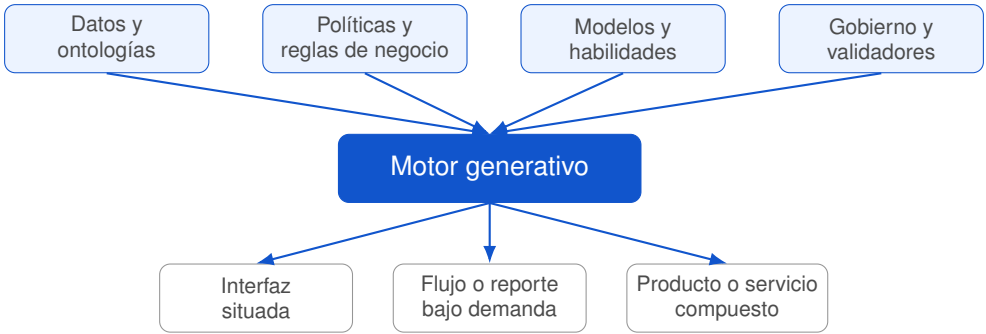
La fuente de verdad migra de la aplicación al conocimiento organizacional.

≫ Giro organizacional

ANTES	DESPUÉS
El software representa la organización.	La organización se representa como conocimiento.
El código implementa los procesos.	Las reglas describen los procesos.
Los usuarios aprenden la aplicación.	La aplicación se adapta a los usuarios.
Se distribuyen aplicaciones.	Se distribuyen modelos organizacionales e intenciones ejecutables.
El software es el producto.	El software es una manifestación temporal del conocimiento organizacional.

Explicitar la regla evita acumular pantallas y flujos.

≫ Núcleo y efemeridad



PERSISTE

Datos, semántica, políticas, capacidades y evidencia.

CADUCA

Pantallas, consultas, reportes y flujos de la situación.

≫ Paradigmas complementarios

Dimensión	Software tradicional	Software efímero
Ontología	Artefacto que continúa	Evento que materializa intención
Optimiza	Longevidad, escala, compatibilidad	Inmediatez, adaptación y contexto
Evolución	Mantener, refactorizar, migrar	Revisar intención y regenerar
Responsabilidad	Custodia del código	Gobierno de la intención
Ámbitos	Infraestructura y sistemas críticos	Exploración, decisiones, creatividad y automatización puntual

Persistir cuando aporta valor; diseñar discontinuidad cuando es una carga.

≫Otras ideas de uso

- Analítica de datos
- Ofimática
- Operaciones semi-automáticas
- Operaciones esporádicas
- Pruebas
- Sistemas agenticos
- Plataformas y productos

≫Endless Polyend

≫ Endless Polyend

08

Correctitud, responsabilidad y verificación

Intención y su validación.

≫Correctitud y responsabilidad

Correcto $\iff$ la realización satisface I^* bajo el contexto C
y supera los criterios de validación definidos al generarse

INTENCIÓN

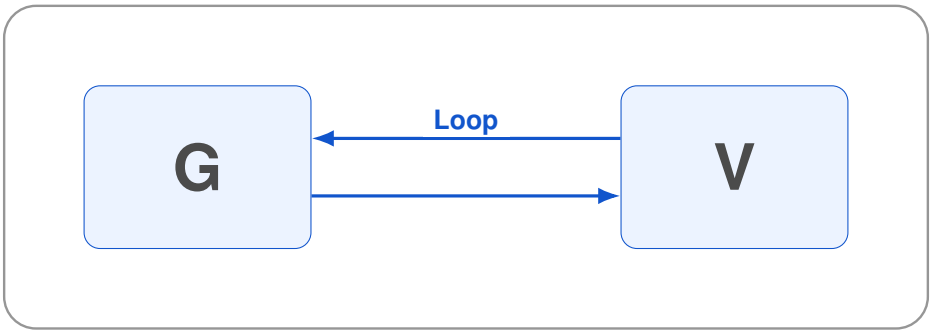
Aprobar y versionar I^*; definir supuestos y criterios de éxito.

INFRAESTRUCTURA

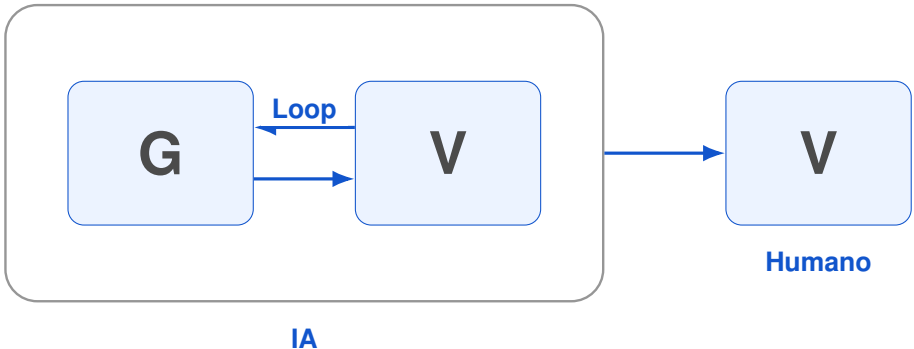
Gobernar modelos, herramientas y datos; trazar generación y ejecución.

La responsabilidad recae en la intención y en el proceso que autoriza la generación.

≫ Verificación



≫ Verificación IA



09

Conclusión

≫Conclusión

- Software como **algo temporal**.
- La **intención ejecutable** preserva significado y responsabilidad.
- Generar, validar y descartar evita mantenimiento innecesario.
- Persistir y formalizar cuando riesgo, escala o reutilización lo exigen.

De codificar a generar conocimiento.