

Diseñando lo que ocurre

Una gramática de eventos para interfaces de usuario

Juan V. Navarro Leal

Diseñando lo que ocurre — Una gramática de eventos para interfaces de usuario

© 2026 Juan V. Navarro Leal

Todos los derechos reservados. Ninguna parte de esta publicación puede reproducirse, almacenarse o transmitirse por ningún medio sin permiso previo y por escrito del autor, salvo citas breves con mención de la fuente.

Primera edición: julio de 2026

Escrito, compuesto y maquetado por el autor.

Índice

Introducción.....	5
Alcance de la propuesta.....	8
Nota terminológica.....	9
Dónde se sitúa esta propuesta.....	10
Qué afirma este libro, y con cuánta seguridad.....	12
Cómo recorrer este libro.....	14
PARTE I — Fundamentos.....	16
1 · Qué entendemos por interfaz.....	17
2 · Por qué preguntamos ahora.....	32
3 · Fundamentos psicoperceptivos de la interfaz.....	49
4 · El evento como unidad básica de cambio.....	62
5 · Por qué una gramática de eventos.....	77
6 · Qué es una interacción semántica.....	91
7 · Criterios para identificar una interacción semántica.....	107
8 · De las clases de evento a las familias semánticas.....	119
9 · Semánticas valenciales y transicionales.....	131
10 · Intent: más allá del semáforo.....	142
PARTE II — Canales.....	152
11 · Los canales de expresión del evento.....	153
12 · Tiempo, latencia y sincronía.....	161
13 · Motion: la física percibida del evento.....	168
14 · Presencia y profundidad.....	173
15 · Forma: límite, affordance e iconografía.....	181
16 · Color: refuerzo semántico y valencia.....	188
17 · Sonido: urgencia, valencia y microeventos auditivos.....	194
18 · Háptica y vibración.....	202
19 · Accesibilidad perceptiva y reducción de canales.....	210

20 · Coordinación multicanal.....	215
PARTE III — Familias.....	220
21 · Cómo leer los capítulos de familia.....	221
22 · Contact.....	225
23 · Commit.....	233
24 · Signal.....	241
25 · Handle.....	249
26 · Emerge.....	256
27 · Shift.....	265
28 · Sustain.....	273
29 · Delegate.....	282
PARTE IV — Sistema.....	303
30 · Composición de eventos semánticos.....	304
31 · Matrices semánticas.....	320
32 · Rangos psicoperceptivos orientativos.....	331
33 · Accesibilidad aplicada.....	342
34 · Antipatronos globales.....	352
35 · Validación de una semántica perceptiva.....	364
36 · De la gramática al sistema.....	374
Cierre.....	384
Apéndice A · Capa semántica declarativa.....	390
Apéndice B · Firmas de expresión y matriz mínima.....	394
Apéndice C · Tres casos de estudio.....	399
Apéndice D · De la gramática al código: el caso Sema.....	402
Apéndice E · Casos difíciles: donde la gramática suda.....	414
Glosario.....	421
Índice analítico.....	424
Referencias bibliográficas.....	425
Lecturas complementarias.....	427
Sobre el autor.....	429

Introducción

La pregunta que aparece al construir interfaces

Este libro nace de una práctica, no de una pretensión.

Durante años, trabajando en front-end, me encontré una y otra vez con preguntas que las herramientas habituales no terminaban de responder del todo.

No eran preguntas espectaculares. Eran preguntas pequeñas, de esas que aparecen mientras se construye una interfaz real:

¿Por qué un botón debe responder antes de confirmar?

¿Por qué guardar no se siente igual que completar?

¿Por qué una espera necesita parecer viva?

¿Por qué una advertencia corregible no debería sentirse como una amenaza?

¿Por qué una pérdida ya producida no debería expresarse igual que un peligro todavía evitable?

¿Por qué un modal a veces cambia el contexto y otras veces solo muestra contenido?

¿Por qué el color ha cargado con tanta semántica?

¿Por qué motion, sonido, forma, presencia, háptica y accesibilidad suelen tratarse como capas separadas si el usuario las vive juntas?

No soy psicólogo. No soy neurocientífico. No pretendo dar lecciones en esos campos.

Soy un programador que, al construir interfaces, empezó a sospechar que faltaba un lenguaje para describir algo que ocurría todos los días delante de nosotros.

Las guías de diseño explicaban componentes. Los frameworks ofrecían patrones. Los sistemas de diseño ordenaban tokens. La accesibilidad establecía criterios necesarios. La psicología ofrecía fundamentos sobre percepción, atención, acción, evaluación y continuidad.

Pero entre todo eso seguía apareciendo una zona difícil de nombrar: qué ocurre cuando la interfaz responde, espera, avisa, confirma, pierde, aparece, cambia de contexto o sostiene un proceso.

Este libro intenta trabajar en esa zona.

No viene a sustituir la teoría de interfaces existente. No viene a redefinir desde cero el diseño de interacción. No viene a invadir campos que ya tienen desarrollos sólidos.

Viene a abordar una pregunta concreta desde otra perspectiva:

¿Podemos pensar la interfaz como una gramática de eventos?

La web será el campo principal de ejemplos, porque ahí trabajo y porque ahí la fragmentación se ve con mucha claridad: DOM, CSS, JavaScript, estados, eventos técnicos, tokens, componentes, motion, accesibilidad, sonido y comportamiento aparecen como capas separadas.

Pero el usuario no vive capas. Vive eventos.

Percibe que algo respondió. Que algo sigue ocurriendo. Que algo se guardó. Que algo reclama atención. Que algo puede perderse. Que algo ya se perdió. Que algo apareció. Que el contexto cambió. Que algo quedó completado. Que el sistema puede actuar por él.

Este libro propone una forma de nombrar esos eventos.

No como verdad final. No como dogma. No como framework cerrado. Sino como una herramienta para pensar mejor una parte de la interfaz que ya estaba funcionando, aunque muchas veces no tuviera nombre.

La hipótesis es sencilla:

La interfaz no solo tiene componentes, estados y efectos.

Tiene eventos.

Y esos eventos necesitan una gramática.

Hay una forma sencilla de situar esta idea. Los sistemas de diseño modernos le dieron a los equipos un vocabulario preciso para lo que las cosas son: átomos, componentes, tokens, variantes. Lo que aún falta es un vocabulario para lo que ocurre —la respuesta, la espera, la confirmación, la pérdida, la aparición, el cambio de contexto, la acción que el sistema hace por el usuario—. Este libro propone esa segunda gramática: la capa que un sistema de diseño todavía no sabe nombrar.

Alcance de la propuesta

Esta gramática no pretende agotar todos los eventos posibles de cualquier interfaz ni sustituir las teorías existentes de HCI, accesibilidad o diseño de interacción.

Propone un vocabulario operativo para describir eventos recurrentes en interfaces digitales, especialmente útil en producto digital, sistemas de diseño y desarrollo front-end.

En dominios especializados —trading, medicina, CAD, simulación, sistemas críticos— pueden aparecer eventos propios que requieran extensiones, subfamilias o reglas específicas. La gramática no se cierra ante ellos: ofrece criterios para decidir cuándo una diferencia merece entrar en el sistema y cuándo debe tratarse como verbo, fase, intent, canal de expresión o composición.

Este libro debe leerse como una propuesta práctica con fundamentos teóricos, no como una ley universal de la percepción ni como una especificación técnica cerrada.

Nota terminológica

El libro usa algunos términos ingleses porque también están pensados para funcionar en documentación técnica, diseño de sistemas y código. Las familias semánticas se nombran como `contact`, `commit`, `signal`, `handle`, `emerge`, `shift`, `sustain` y `delegate`. En el texto narrativo se explican en español; en tablas, notación y ejemplos técnicos se mantiene la etiqueta inglesa.

La palabra `intent` se usa como etiqueta del sistema. No significa «intención del usuario». Significa la valoración que declara un evento sobre cómo debe percibirse. Los seis intents se nombran, como las familias, con etiquetas inglesas: `neutral` (neutro), `affirm` (confirmatorio), `fulfill` (resolutivo), `risk` (corregible), `threat` (amenazante) y `loss` (perdido). Estas equivalencias se dan aquí una sola vez; a partir de este punto, el libro usa siempre las etiquetas inglesas.

El término `motion` se usa como canal de expresión. No equivale simplemente a animación técnica. La animación es la implementación; `motion` es el movimiento tal como participa en la lectura del evento.

Este libro usa la palabra semántica para una sola cosa: el significado que un evento adquiere para el usuario dentro de una interfaz. No pretende construir una semántica formal en sentido lógico.

Dónde se sitúa esta propuesta

Esta gramática no nace en el vacío. Dialoga con tradiciones que han pensado la interacción, la acción, la percepción, la semiótica y la accesibilidad desde otros ángulos.

La Semiotic Engineering entiende la interacción humano-computador como comunicación entre quienes diseñan el sistema y quienes lo usan. Esta propuesta comparte esa preocupación por el significado, pero trabaja en una escala más concreta: el evento que la interfaz hace perceptible.

La Activity Theory entiende la tecnología como mediación de la actividad humana. Este libro no sustituye ese marco, pero toma de él la idea de que una interfaz no puede analizarse solo como objeto visible: debe entenderse por las acciones que habilita y transforma.

El Ecological Interface Design trabaja la interfaz como soporte para actuar en sistemas complejos. Esta gramática comparte la preocupación por hacer visibles relaciones operativas, aunque se centra en producto digital, sistemas de diseño y eventos de interfaz.

WAI-ARIA ya proporciona una semántica técnica para accesibilidad web mediante roles, estados y propiedades. Esta propuesta no reemplaza ARIA; intenta complementar esa semántica técnica con una gramática de eventos perceptibles para diseño e implementación.

Los sistemas de design tokens y componentes han permitido estandarizar color, espaciado, tipografía, elevación, estados y variantes. Esta gramática no sustituye esos sistemas. Propone una capa anterior: declarar qué

evento ocurre antes de decidir qué token, componente o canal de expresión lo materializa.

Las microinteracciones, tal como las formuló Saffer, descomponen cada pequeño momento de uso en disparador, reglas, feedback y bucle. Esta gramática comparte esa atención al detalle, pero no se queda en la microinteracción aislada: ofrece un vocabulario común que va desde el acuse de un botón hasta el cambio de contexto o la acción delegada, y que permite hablar de todos ellos con las mismas categorías.

Los sistemas de motion de las grandes bibliotecas de diseño han enseñado que el movimiento puede ser significativo y no solo decorativo. Esta propuesta toma esa convicción y la generaliza: motion es uno de varios canales al servicio del evento, no un lenguaje propio. La pregunta no es «qué animación queda bien», sino «qué evento se está expresando y qué canal lo porta mejor».

Los patrones emergentes de las interfaces agénticas —indicadores de autonomía, líneas de tiempo de acción, controles de intervención— responden a la misma necesidad que aquí ocupa la familia delegate: hacer perceptible quién actúa. La diferencia es de alcance. Donde esos patrones forman catálogos sueltos, esta gramática integra esa pregunta en el mismo sistema que organiza el resto de los eventos, de modo que «el sistema actuó por ti» se describe con las mismas herramientas que «tu acción se registró».

La contribución del libro no es inventar la interacción desde cero. Es ofrecer un lenguaje operativo para que diseñadores, programadores y equipos de producto puedan hablar con precisión de lo que ocurre en una interfaz.

Qué afirma este libro, y con cuánta seguridad

Tres niveles de afirmación, declarados antes de empezar

Este libro mezcla, como casi cualquier propuesta de diseño, tres niveles de afirmación. Conviene declararlos antes de empezar, porque no merecen la misma confianza y sería deshonesto venderlos al mismo precio.

El primer nivel es ciencia perceptiva consolidada, con décadas de evidencia y réplica. La percepción de causalidad que Michotte midió en laboratorio; la atribución de agencia que Heider y Simmel documentaron; la organización en figura y fondo de la Gestalt. Las affordances de Gibson; el espacio de valencia y activación de Russell; la teoría de la valoración de Scherer. Cuando el Capítulo 3 los presenta, el libro está citando, no proponiendo. Este nivel no depende de que la gramática funcione.

El segundo nivel es extrapolación plausible: llevar esos hallazgos, obtenidos casi siempre fuera de las interfaces, al terreno de las interfaces. Que la ventana de causalidad percibida gobierne la latencia de un botón; que el espacio valencia/activación organice los intents; que la atención capturada por el movimiento explique la disciplina del canal motion. Son traslados razonables y coherentes con la evidencia — pero son traslados. Nadie ha medido la mayoría de ellos donde este libro los aplica.

El tercer nivel es propuesta de este autor, y es el grueso del libro: las ocho familias y no otras, los seis intents y no cinco, los verbos, las reglas de composición, los rangos orientativos. Nada de eso está validado empíricamente. Es una taxonomía construida sobre los dos niveles anteriores, con criterios declarados (Capítulo 7) y con un método

propuesto para ponerla a prueba (Capítulo 35) que todavía no se ha ejecutado. El Cierre lo dice y este párrafo lo adelanta: la gramática es provisional.

Queda la palabra «gramática», que invita al escrutinio lingüístico. La analogía se asume aquí una sola vez y con sus límites. Las familias funcionan como clases de palabras; los verbos, como su morfología. Las reglas de composición, como una sintaxis mínima; el intent, como la modalidad que tiñe un enunciado. Hasta ahí llega. El libro no afirma que la interfaz sea un lenguaje, ni que exista una gramática universal de los eventos, ni que estas reglas generen significado como genera oraciones una lengua. «Gramática» nombra una ambición de estructura — diferencias, relaciones, composición — y nada más.

La honestidad de este libro no está en fingir que todo es ciencia. Está en que los tres niveles queden a la vista — y en que el tercero, que es el más grande, se declare como lo que es: una propuesta esperando su prueba.

Cómo recorrer este libro

Tres itinerarios y una hora, para que cuatrocientas páginas no exijan leerse de una vez

El libro está construido para leerse de principio a fin: cuatro partes que se apoyan — fundamentos, canales, familias, sistema—, un cierre y cinco apéndices. Pero no todo lector llega con el mismo propósito, y un libro de este tamaño debe ofrecer más de una puerta. Aquí van tres itinerarios y un atajo.

El itinerario del diseñador

Empieza por los Capítulos 1 y 2: qué es mirar la interfaz desde lo que ocurre en ella — y no como una superficie—, y por qué las convenciones heredadas ya no saben justificarse. Salta a los Capítulos 8, 9 y 10 — las ocho familias, qué acepta evaluación y los seis intents: la mercancía central del libro—. Sigue con el 21, y usa los capítulos de familia (22 a 29) como consulta, empezando siempre por su ficha visual. Cierra con el lado de decisión de la Parte IV: composición (30), matrices (31), accesibilidad aplicada (33) y el catálogo de antipatronos (34).

El itinerario del developer

El mismo arranque: 1 y 2, luego 8, 9 y 10. Después, directo a lo operativo: el 21 como mapa, los capítulos de familia en modo consulta y la Parte IV entera, con el 36 — el sistema de diseño — como destino. Los apéndices son tuyos: la referencia canónica del B, la gramática hecha código del D y los casos difíciles del E. Si vienes del código, el D puede incluso ser la primera lectura: es la prueba de carga de todo lo demás.

El itinerario del investigador

Empieza por el 2 — la genealogía de las convenciones — y el 3 — lo que la percepción ya sabía—. Sigue con el 5 (por qué una gramática y no un catálogo de patrones) y el 7 (los criterios para que una diferencia se vuelva lenguaje). Los fundamentos evaluativos viven en el 9 y el 10. La frontera declarada del sistema está en el §11 del Capítulo 29 y en el Apéndice E; el método de validación, en el 35; y la posición frente a las tradiciones, en estas páginas preliminares y en el Cierre.

Si solo tienes una hora

Lee el Capítulo 2 y el Capítulo 10. Son el porqué y la mercancía: de dónde vienen las convenciones que nadie decidió, y la diferencia entre amenaza y pérdida, entre confirmar y culminar. Con eso y la referencia canónica del Apéndice B se puede empezar a trabajar mañana — y decidir si el resto del viaje merece la pena.



Fundamentos

Foundations

Qué es una interfaz cuando se mira desde lo que ocurre en ella; qué es un evento; por qué hace falta una gramática; y qué son las interacciones semánticas, las familias y los intents.

01

Qué entendemos por interfaz

Hacia una definición operativa de la interfaz

PREGUNTA DEL CAPÍTULO

¿Qué es exactamente una interfaz cuando dejamos de mirarla como una superficie y empezamos a mirarla desde lo que ocurre en ella?

La palabra interfaz parece clara hasta que uno intenta definirla con precisión.

En el trabajo cotidiano solemos usarla sin demasiadas dudas. Decimos que una interfaz tiene botones, menús, formularios, diálogos, listas, pestañas, estados de carga, mensajes de error y acciones disponibles. Decimos que una interfaz es clara, confusa, pesada, fluida, accesible, elegante o difícil de usar.

La palabra sirve para muchas cosas. Quizá por eso también oculta algunas.

Conviene, antes de seguir, escuchar a la propia palabra. *Interfaz* llegó al español desde el inglés, pero sus piezas son latinas: *inter*, «entre», y

facies, «cara» — la forma con la que algo se presenta. Literalmente: lo que hay entre dos caras. La palabra no nombra una cosa, sino un encuentro: el lugar donde dos entidades se dan la cara y, con ella, su lado perceptible.

Su primera casa fue la física del siglo XIX. Allí *interface* nombraba la superficie de contacto entre dos medios — el agua y el aire, el aceite y el agua, dos fases de la misma materia—. Y ya entonces era todo menos una superficie pasiva: en la interfaz vive la tensión superficial; en la interfaz ocurren el intercambio y la reacción. Los dos medios pueden ser inertes; su frontera es el lugar donde pasan cosas.

La ingeniería del siglo XX tomó prestada la palabra para los puntos de conexión entre sistemas, y la informática la heredó de ahí. Primero entre máquinas, luego entre programas, y al final entre el sistema y la persona. En ese último viaje la palabra se aplanó. «Interfaz de usuario» acabó significando, en la práctica, la pantalla: lo que se ve, lo que se dibuja, lo que se maqueta. La superficie.

La etimología, así leída, no es una curiosidad: es un aviso. La palabra nació nombrando la frontera activa entre dos medios — el lugar definido por lo que ocurre en él—, y este capítulo va a intentar devolvérsela.

Durante mucho tiempo, la definición más práctica ha sido suficiente: una interfaz es el punto de contacto entre una persona y un sistema. Es la superficie mediante la cual el sistema se vuelve visible, interpretable y manipulable.

Esa definición ha sido útil. Todavía lo es. Gracias a ella hemos aprendido a construir pantallas legibles, jerarquías visuales, patrones de navegación, controles reutilizables y guías de estilo.

El problema no es que esa definición sea falsa. El problema es que empieza a quedarse corta.

Porque cuando una persona usa una interfaz, no se relaciona únicamente con una superficie. También percibe respuestas, cambios, interrupciones,

confirmaciones, esperas, apariciones, pérdidas, errores, desplazamientos, manipulaciones y consecuencias.

Una interfaz no es solo aquello que está ahí. Es también el lugar donde ocurren cosas.

Si la interfaz se entiende solo como superficie, el componente tiende a convertirse en la unidad principal de análisis. Pero si la interfaz se entiende también como lugar donde ocurren transformaciones, aparece otra pregunta:

¿Qué clase de evento está percibiendo el usuario?

§ 1

La interfaz como superficie

La concepción clásica de interfaz nace de una necesidad concreta: hacer que un sistema pueda ser usado. Un sistema informático, por sí mismo, no se ofrece directamente a la percepción humana. Sus operaciones internas no son visibles. Sus estados no se presentan espontáneamente como posibilidades comprensibles.

La interfaz aparece para mediar esa distancia. Convierte estructuras internas en formas perceptibles. Traduce posibilidades abstractas en controles. Presenta estados. Organiza caminos de acción.

Desde este punto de vista, la interfaz es una superficie de traducción. No una superficie en el sentido pobre de «lo visual», sino en el sentido de una zona de contacto: el lugar donde dos realidades distintas se encuentran.

Esta tradición ha producido herramientas indispensables: jerarquía visual, contraste, espaciado, agrupación, estados de componente, affordances, patrones de navegación, consistencia, accesibilidad, sistemas de diseño y componentes reutilizables.

Nada de eso queda invalidado por la propuesta de este libro. Una gramática de eventos necesita apoyarse en esa base.

PRINCIPIO

La gramática de eventos no sustituye a la gramática visual; la prolonga.

Pero la superficie, por sí sola, no basta. Porque la experiencia de uso no está hecha solo de elementos dispuestos en el espacio. Está hecha de elementos que cambian, responden, aparecen, desaparecen, se desplazan, se fijan, se bloquean o se transforman.

La interfaz no es una imagen estática. Es un contexto dinámico en constante evolución, alimentado por la interacción del usuario y por el sistema.

§ 2

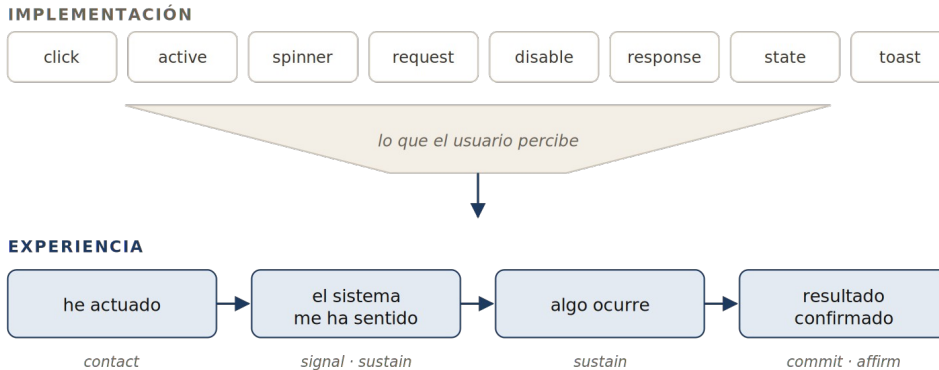
El límite de la superficie

Pensemos en algo sencillo: un usuario pulsa un botón para guardar cambios. Desde la implementación pueden ocurrir muchas cosas: se dispara un evento click, el botón entra en estado active, aparece un spinner, se envía una petición, se deshabilita el control, llega una respuesta, se muestra una confirmación, cambia un estado interno, quizá aparece un toast.

Técnicamente son muchas operaciones. Para el usuario, sin embargo, hay una secuencia más compacta: he actuado; el sistema me ha sentido; algo está ocurriendo; el resultado ha quedado confirmado.

FIGURA 1.1

De muchas operaciones técnicas a pocas unidades de sentido



La experiencia no se organiza según las capas de implementación, sino según unidades de sentido operativo.

La experiencia no se organiza según las piezas con las que construimos —componentes, estados, peticiones, estilos—. Se organiza en unidades de significado.

Lo mismo sucede con un diálogo modal. Desde la superficie, es una capa que aparece sobre otra. Pero desde la experiencia, puede significar cosas muy distintas: una confirmación ligera, una interrupción crítica, un cambio temporal de foco, una elección irreversible.

La interfaz no solo representa estados; también organiza transformaciones.

§ 3

Del objeto al cambio

El diseño de interfaces ha desarrollado una cultura poderosa alrededor del objeto. Definimos botones, inputs, selects, cards, dialogs, menus, tooltips, accordions, tabs. Cada componente tiene una anatomía, unas variantes, unos estados y unas reglas de uso.

Ese enfoque ha sido enormemente productivo. Pero el objeto no siempre es la unidad que el usuario necesita comprender. Un botón no es significativo solo por ser botón. Lo es por lo que permite hacer y por lo que ocurre cuando se actúa sobre él.

Por eso, para una gramática semántica de la interfaz, el centro debe desplazarse: del componente al evento, de la forma al cambio, del estado aislado a la transformación, de la propiedad visual al significado perceptivo.

Distinción clave

Un componente es una entidad de diseño. Un evento es una unidad de experiencia.

A partir de ahí aparece una segunda idea: una interfaz no es solo una superficie visible, sino la dimensión perceptible de un contexto operativo. Cuando usamos una interfaz, no estamos mirando figuras flotando en una pantalla. Estamos en medio de algo. Queremos hacer algo, corregir algo, elegir algo, esperar algo, confirmar algo. Hay objetivos, estados, restricciones, riesgos, expectativas y consecuencias. La interfaz es la parte de todo ese contexto que podemos percibir.

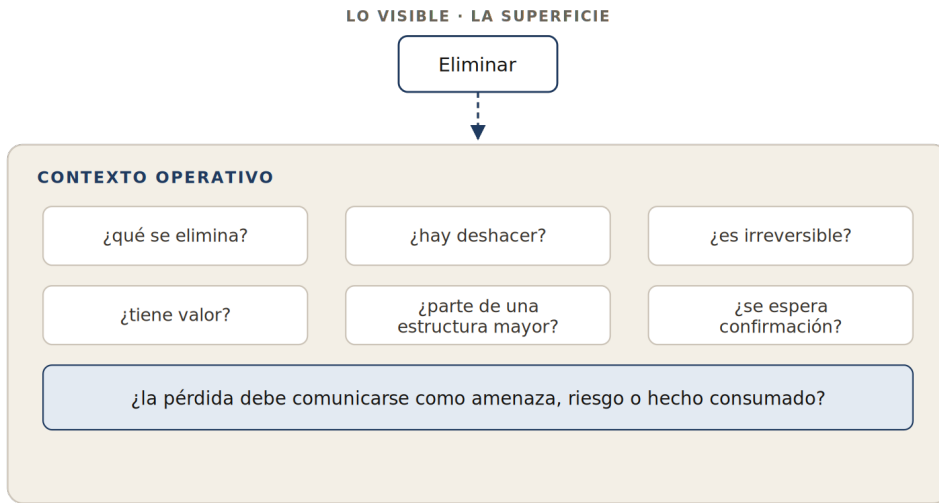
§ 4

Qué llamamos contexto operativo

Un contexto operativo es el conjunto dinámico de condiciones que hacen que una acción signifique algo. No es solo la pantalla. No es solo el estado interno del sistema.

FIGURA 1.2

El contexto operativo



El botón es solo la punta visible. El contexto operativo es todo lo que hace que la acción signifique algo.

Cuando un usuario pulsa «Eliminar», el contexto operativo no se reduce al botón rojo. Incluye qué se elimina, si hay deshacer, si la acción es irreversible, si el elemento tiene valor, si forma parte de una estructura más grande, si el usuario espera confirmación y si la pérdida debe comunicarse como amenaza, riesgo o hecho consumado.

El contexto operativo no es una capa más del sistema. Es aquello que la interfaz hace perceptible.

§ 5

La interfaz como dimensión perceptible

Si el contexto operativo es aquello en lo que estamos metidos, la interfaz es lo que nos permite percibirlo. No todo el contexto se vuelve perceptible de la misma manera. Algunas partes se expresan mediante texto. Otras mediante posición. Otras mediante color. Otras mediante movimiento.

Otras mediante sonido. Otras mediante ausencia, bloqueo, espera o silencio.

La interfaz no necesita contestar siempre con palabras. Parte de su trabajo consiste en distribuir respuestas entre señales perceptivas: posición, contraste, ritmo, continuidad, jerarquía espacial, tono, presencia, desaparición. Un campo que tiembla prepara la atención. Un botón que se comprime preserva la sensación de contacto. Un panel que aparece desde un origen hace visible una relación. Un objeto que se eleva al ser arrastrado hace sensible el cambio de régimen.

Por eso conviene hablar de dimensión perceptible. La interfaz es la zona donde el sistema se pone al alcance de un cuerpo humano: visible, audible, táctil, temporalmente inteligible y accionable.

§ 6

La interfaz no es solo visual

Uno de los supuestos más persistentes en el diseño de interfaces es que la interfaz es fundamentalmente visual. Este supuesto tiene razones históricas claras: la pantalla ha sido el medio dominante, las herramientas de diseño trabajan sobre superficies visuales, la web misma nació como documento visible.

Pero incluso dentro de lo visual, conviene distinguir canales que suelen mezclarse. Un cambio de color no comunica lo mismo que un desplazamiento. Una sombra no comunica lo mismo que un borde. Una aparición no comunica lo mismo que una vibración de alerta.

FIGURA 1.3

Dentro de lo «visual» hay muchos canales



La interfaz no es visual más algunos extras: es perceptiva, y la percepción se reparte entre canales distintos.

La interfaz no es visual más algunos extras. Es perceptiva. Y esa percepción se distribuye entre canales distintos, cada uno con mecanismos propios, reglas propias y condiciones propias de eficacia. El hecho de que algunos canales no estén disponibles en ciertos entornos no los vuelve teóricamente irrelevantes.

La pregunta de fondo es siempre la misma: ¿qué necesita percibir el usuario para entender lo que ocurre?

§ 7

Estados, eventos y consecuencias

Un estado describe una condición relativamente estable: seleccionado, desactivado, cargando, inválido. Un evento describe una transformación: seleccionar, desactivar, empezar a cargar, fallar.

La diferencia importa porque el usuario necesita entender ambas cosas. Necesita saber que algo está seleccionado, pero también necesita percibir

que la selección acaba de ocurrir. Necesita saber que algo está cargando, pero también necesita percibir cuándo empezó y si sigue vivo.

TABLA 1.1 Los cuatro planos de análisis de la interfaz

PLANO	PREGUNTA	EJEMPLO	FORMA TÍPICA
Componente	¿Qué es esto?	botón, modal, toast	Elemento del DOM
Estado	¿En qué condición está?	seleccionado, abierto, cargando	Clase CSS, propiedad
Evento	¿Qué cambio acaba de ocurrir?	seleccionar, abrir, iniciar carga	Transición, señal perceptiva
Consecuencia	¿Qué implica ahora?	quedó fijado, hay que esperar	Interpretación del usuario

Una interfaz centrada solo en estados puede mostrar bien el punto de llegada y, aun así, dejar ilegible el camino: qué cambió, cuándo y por qué. Por eso este libro insistirá en que una interfaz no debería limitarse a mostrar estados finales. Debe hacer legibles las transformaciones relevantes.

§ 8

Qué cuenta como evento

No cualquier cambio perceptible merece el mismo rango. Para hablar de evento en sentido fuerte hacen falta tres condiciones.

Primero, debe haber una modificación perceptible. Algo debe cambiar en el campo perceptivo del usuario.

Segundo, esa modificación debe tener relevancia operativa. Debe afectar a la acción, la atención o la interpretación del estado.

Tercero, debe poder inscribirse en una clase reconocible. Una gramática necesita identificar diferencias recurrentes.

Definición

Un evento, en el sentido que irá precisando este libro, es un cambio perceptible que altera lo que el usuario puede hacer, atender o entender.

Esta definición excluye tanto la pura lógica interna como la ornamentación sin función. Un callback técnico no es necesariamente un evento. Una animación decorativa tampoco. Pero cuando una modificación perceptible ayuda al usuario a entender qué está ocurriendo, empieza a tener estatuto de evento significativo.

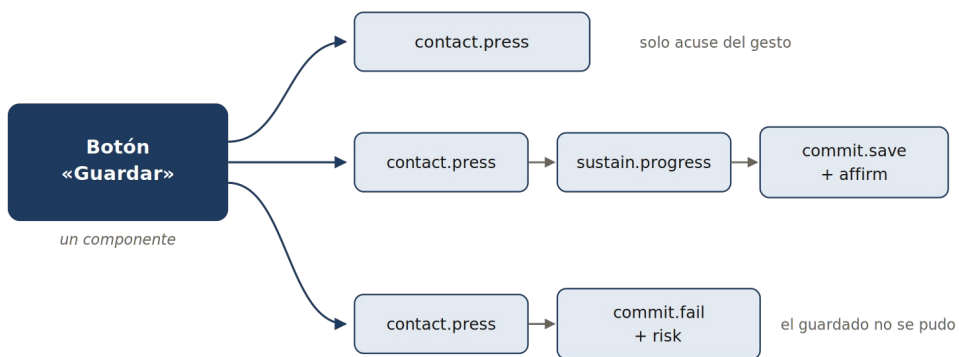
§ 9

Un botón no es un evento

Tomemos un botón simple: «Guardar». Desde el punto de vista del componente tiene un label, un tamaño, una variante, estados hover, active, disabled. Pero desde el punto de vista del evento, el mismo botón puede participar en secuencias distintas.

FIGURA 1.4

Un componente, varios eventos



El mismo botón participa en eventos distintos; una gramática de eventos vuelve explícita esa diferencia.

En todos los casos el componente puede ser «un botón de guardar». Pero los eventos no son los mismos. Un enfoque centrado solo en componentes

tiende a ocultar esa diferencia. Una gramática centrada en eventos la vuelve explícita.

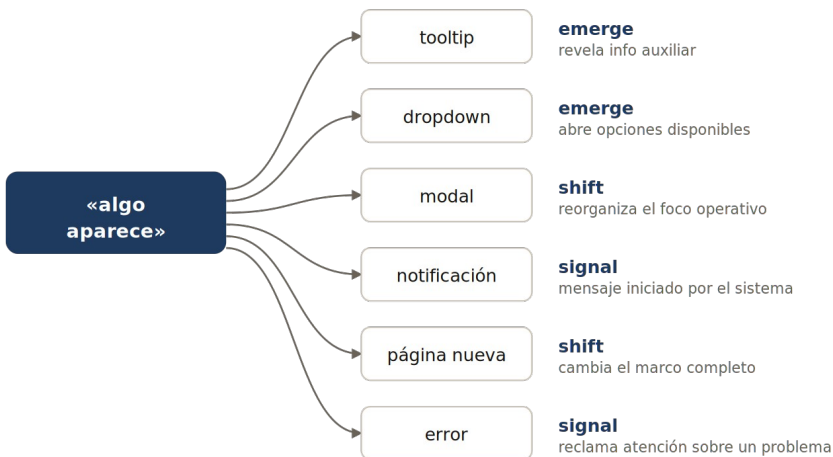
§ 10

Abrir no siempre significa lo mismo

Un tooltip aparece. Un dropdown aparece. Un modal aparece. Una página nueva aparece. Una notificación aparece. Un error aparece. Superficialmente podríamos decir: todos son eventos de aparición. Pero esa descripción es demasiado gruesa.

FIGURA 1.5

No toda aparición es igual



«Algo aparece» es demasiado grueso: cada aparición es un evento distinto según el contexto operativo.

Un tooltip revela información auxiliar sin cambiar el marco principal. Un modal reorganiza el foco operativo y subordina lo demás. Una notificación introduce un mensaje iniciado por el sistema. «Algo aparece» no basta. La pregunta relevante es: ¿qué tipo de aparición es esta dentro del contexto operativo?

§ 11

Por qué esta redefinición importa al programar

Un click no es una semántica. Un change no es una semántica. Un submit no es una semántica. Un drag no es una semántica. Son disparadores técnicos. Pueden dar lugar a eventos muy distintos.

Un click puede ser contacto, consolidación, navegación, apertura, amenaza, pérdida o confirmación. Un submit puede ser inicio de proceso, culminación, fallo, riesgo o resolución. Un drag puede ser manipulación, desplazamiento, reordenación o descarte.

Cuando no tenemos vocabulario para estas diferencias, las resolvemos de forma local. Una gramática semántica permite que antes de discutir «qué transición ponemos aquí», podamos preguntar: ¿qué evento estamos haciendo perceptible? Y antes de discutir «qué color usamos»: ¿este evento es evaluable o solo estructura el campo?

§ 12

Resumen operativo

No estamos diciendo que la interfaz clásica esté equivocada. No estamos diciendo que todo deba animarse, sonar o vibrar. No estamos diciendo que los componentes dejen de importar. No estamos diciendo que todo cambio sea semántico. No estamos diciendo que la ciencia pueda dictar cada duración o cada frecuencia.

Lo que sí estamos diciendo es más sencillo:

La interfaz no se comprende del todo si solo se analiza como superficie de componentes. También debe analizarse desde sus eventos.

§ 13

Síntesis

Esta idea se desarrollará con más precisión en el capítulo de fundamentos psicoperceptivos. Allí veremos que la interfaz no solo se organiza visualmente; también se apoya en percepción de campo, affordances, atención, evaluación, movimiento, sonido, multisensorialidad y experiencia corporal.

Por ahora basta con fijar una tesis:

DISTINCIÓN

La interfaz no es visual más algunos extras. Es perceptiva.

Síntesis

La interfaz es la dimensión perceptible de un contexto operativo.

TABLA 1.2 El desplazamiento conceptual que propone este libro

ANTES	DESPUÉS
Interfaz como superficie	Interfaz como dimensión perceptible
Componente como centro	Evento como unidad de significado
Estado final	Transformación significativa
Efecto local	Gramática del cambio
Implementación fragmentada	Experiencia unificada

El usuario no necesita percibir solo que algo existe, sino qué tipo de cambio ha ocurrido y qué implica. Desde aquí, el siguiente paso será desarrollar la noción de evento con más detalle.

Porque si la interfaz es la dimensión perceptible de un contexto operativo, entonces su unidad básica de cambio no puede ser únicamente el elemento visible.

Tiene que ser el cambio significativo.

Tiene que ser el evento.

La redefinición de interfaz como dimensión perceptible de un contexto operativo prepara el desplazamiento central del libro: del componente al

evento. El componente sigue siendo necesario, pero ya no basta como unidad principal de significado. Cuando la interfaz cambia, responde, espera, interrumpe, confirma o pierde algo, el usuario no interpreta solo objetos: interpreta eventos.

02

Por qué preguntamos ahora

Sedimentación, dependencia del camino y la carga paradigmática del diseño de interfaces

PREGUNTA DEL CAPÍTULO

¿Por qué seguimos diseñando lo que ocurre con convenciones que nadie decidió y que ya no sabemos justificar?

Al lector de este libro le asaltará pronto una pregunta incómoda: si algunas de las convenciones que utilizamos todos los días para diseñar interfaces están tan débilmente fundamentadas como este texto sostendrá, ¿cómo es posible que hayan persistido durante décadas sin que nadie las cuestionase?

¿Por qué, si el intent de cuatro categorías hereda su estructura de los semáforos ferroviarios del siglo XIX, la industria no ha vuelto sobre esa herencia para preguntarse si es correcta? ¿Por qué el sonido en la web se sigue tratando como accesorio cuando la neurociencia afectiva lleva treinta años documentando que el procesamiento auditivo es el más

rápido y el más eficaz para comunicar valencia emocional? ¿Por qué la dimensión temporal del evento —el cómo, no solo el qué— sigue siendo tratada como decoración añadida en lugar de como sustancia del significado?

La respuesta natural, la que primero viene a la mente, es que hay una especie de deuda técnica histórica: decisiones que se tomaron hace tiempo, que funcionaron entonces, y que siguen vigentes por inercia. Este capítulo sostendrá que esa respuesta, aunque contiene una intuición correcta, se queda corta. Lo que ocurre en el diseño de interfaces no es exactamente deuda técnica. Es algo más profundo y más difícil de erradicar: una combinación de *sedimentación cultural sin examen*, *dependencia del camino* y lo que la filosofía de la ciencia llama *carga paradigmática*. Entender estos tres mecanismos es la condición para poder preguntar, por primera vez, qué parte del edificio del diseño de interfaces se sostiene y qué parte simplemente persiste.

§ 1

Por qué «deuda técnica» no es la categoría correcta

La expresión *deuda técnica* fue acuñada por Ward Cunningham a finales de los años ochenta para describir un fenómeno concreto del desarrollo de software: las decisiones de implementación que se toman a sabiendas de que son subóptimas, asumiendo el coste futuro de corregirlas a cambio de avanzar más rápido en el presente. La deuda técnica tiene tres características constitutivas. Hay una **decisión consciente**: alguien sabe que la solución no es la buena. Hay un **trade-off temporal**: se acepta el coste a cambio de un beneficio inmediato. Y hay una **voluntad latente de pagarla**: la deuda se contrae con la intención —aunque sea diferida— de saldarla algún día.

Lo que el diseño de interfaces arrastra no cumple ninguna de estas tres condiciones. Cuando Bootstrap definió en 2012 sus cuatro clases contextuales —*success*, *danger*, *warning*, *info*— no fue una decisión consciente de posponer un análisis más profundo de los estados afectivos que una interfaz necesita comunicar. No hubo ningún debate interno en el equipo de Twitter sobre si esas cuatro categorías agotaban el espacio afectivo relevante. No se registró en ningún commit un comentario del tipo «provisional, revisar cuando tengamos tiempo». Simplemente se adoptó la estructura más obvia —la que todo el mundo ya entendía del semáforo—, se implementaron cuatro clases CSS, y se siguió adelante.

Lo mismo ocurre con prácticamente todas las convenciones históricas que este libro examinará. La idea de que el sonido es un accesorio de la interfaz, no un canal constitutivo, no procede de un análisis perceptivo riguroso que concluyera «el sonido es secundario». Procede de limitaciones tecnológicas iniciales (la web no podía reproducir sonido de forma fiable), políticas de navegador (autoplay restringido por razones de abuso publicitario) y convenciones culturales (oficinas silenciosas, uso en público). Ninguna de estas razones es falsa, pero ninguna es el resultado de preguntarse qué papel ocupa el sonido en la percepción humana de los eventos.

Esta distinción importa porque las dos situaciones se corrigen de forma muy diferente. La deuda técnica se paga reescribiendo código: hay un problema identificado, una solución conocida, y solo falta el tiempo para implementarla. Lo que el diseño de interfaces arrastra no se paga así, porque el problema ni siquiera está identificado. La mayoría de los desarrolladores y diseñadores que hoy escriben `class="alert alert-danger"` no creen estar contrayendo ninguna deuda. Creen estar usando un estándar correctamente. No hay conciencia de insuficiencia, y sin conciencia de insuficiencia no hay deuda que saldar —hay un supuesto que nunca fue examinado.

DISTINCIÓN

Una deuda se contrae sabiendo que se contrae. Un supuesto no examinado es invisible para quien lo sostiene. Lo primero se corrige con tiempo y código. Lo segundo requiere un tipo de trabajo cualitativamente distinto: el examen de las premisas sobre las que uno mismo está operando.

§ 2

Sedimentación cultural sin examen

Basta abrir cualquier aplicación y guardar un archivo. Lo más probable es pulsar el icono de un disquete: un cuadrado de plástico de tres pulgadas y media que casi nadie por debajo de los treinta ha sostenido jamás. Guardamos pulsando la imagen de un objeto muerto. Para «llamar» tocamos la silueta de un auricular que hace tiempo que no tiene esa forma. Adjuntamos un «clip». Mandamos copia con las siglas CC —carbon copy—, que nombran un papel carbón desaparecido con las máquinas de escribir. Borramos arrastrando a una «papelera».

Ninguna de esas imágenes describe ya lo que hace. Son fósiles: capas de una tecnología anterior que quedaron incrustadas en la siguiente y siguieron sirviendo como signo mucho después de morir como objeto. No están ahí porque sean las mejores metáforas posibles, sino porque, una vez que algo se entiende, cambiarlo cuesta más que dejarlo.

Lo que en realidad ocurre en el diseño de interfaces es un fenómeno que se entiende mejor con una metáfora geológica que con una metáfora financiera. Las convenciones no se han contraído como deudas: se han sedimentado como capas. Cada generación de herramientas, frameworks y prácticas ha depositado su sedimento sobre el anterior, y lo que tenemos ahora es el resultado acumulativo de esa sedimentación.

La capa más profunda la constituyen las convenciones heredadas de campos previos. El rojo como señal de peligro no nació en las interfaces

digitales: nació en la señalización ferroviaria del siglo XIX, que a su vez heredó asociaciones aún más antiguas del mundo industrial y del lenguaje natural. La jerarquía tipográfica procede de la imprenta y se formaliza con Gutenberg. La organización en columnas y cajas procede del diseño editorial y del cartelismo del siglo XX. El concepto mismo de «página» como unidad de interfaz web procede directamente del papel. Cada una de estas convenciones fue apropiada porque resolvía un problema inmediato en el nuevo medio digital, y porque el nuevo medio aún no tenía vocabulario propio para plantearlo de otra manera.

Sobre esa capa se depositan las convenciones de la primera generación de interfaces digitales: el escritorio metáfora (Xerox PARC, Apple Lisa, Macintosh), los menús desplegables, las ventanas apilables, los botones tridimensionales con sombras. Esta capa introdujo innovaciones reales, pero también cristalizó soluciones que respondían a limitaciones tecnológicas de la época —pantallas de baja resolución, paletas de color limitadas, potencia de cálculo escasa— y que sobrevivieron mucho después de que esas limitaciones desaparecieran.

Sobre esa capa llegaron la web (HTML, CSS, el modelo de caja), los frameworks de componentes (Bootstrap, Material, y toda su descendencia), los sistemas de diseño corporativos (Fluent, Carbon, Human Interface Guidelines), y las librerías headless (Radix, Bits UI, Ariakit). Cada una de estas capas tomó como punto de partida el sedimento acumulado por las anteriores, resolvió los problemas que tenía delante, y añadió su propio sedimento.

FIGURA 2.1

La sedimentación de las convenciones



El resultado es un edificio de convenciones cuyos cimientos están tan cubiertos por capas posteriores que ya no son visibles. Cuando un diseñador en 2026 elige una paleta de colores para estados del sistema, no está tomando una decisión desde cero: está tomando una decisión condicionada por Material de Google, que a su vez estuvo condicionado por Bootstrap, que a su vez estuvo condicionado por las convenciones web anteriores, que a su vez estuvieron condicionadas por las convenciones de los sistemas operativos de escritorio, que a su vez estuvieron condicionadas por las convenciones industriales, que a su vez heredaron las convenciones ferroviarias. Cada uno de estos eslabones tuvo una razón local para hacer lo que hizo. Pero la cadena completa no fue nunca sometida a examen perceptivo.

Esta sedimentación no es mala en sí misma. Es, de hecho, una forma eficiente de acumular conocimiento: no hace falta reinventar la jerarquía tipográfica cada vez que se diseña una interfaz, ni redescubrir que los botones deben parecer presionables. Lo problemático no es que haya

convenciones heredadas. Lo problemático es que algunas de esas convenciones operan en zonas donde el medio digital ha desbordado completamente los contextos originales que las justificaban, y donde la herencia deja de ser una base sólida para convertirse en una limitación invisible.

§ 3

Dependencia del camino

Una vez que una convención se ha sedimentado y se ha difundido, entra en juego un segundo mecanismo que explica por qué persiste aunque existan alternativas mejores. La filosofía de la ciencia y la economía histórica lo llaman *dependencia del camino* (*path dependency*), un concepto desarrollado por el economista Paul David en los años ochenta y refinado por W. Brian Arthur en su análisis de los rendimientos crecientes en procesos históricos.

La dependencia del camino describe una situación en la que una elección inicial, tomada en condiciones concretas y a menudo por razones contingentes, determina las elecciones posteriores incluso cuando las condiciones originales han desaparecido. Cada decisión refuerza el camino elegido, y el coste de abandonarlo crece con el tiempo hasta que el cambio se vuelve prácticamente imposible: no porque sea técnicamente inviable, sino porque el sistema entero —prácticas, herramientas, conocimientos, expectativas— se ha organizado alrededor del camino inicial.

El ejemplo clásico: QWERTY

El caso más conocido es la distribución QWERTY del teclado, y no por casualidad: fue el ejemplo con el que David puso nombre a la idea. En junio de 1868, Christopher Latham Sholes patentó la máquina que acabaría imponiendo la forma en que el mundo escribe. Su problema no

era la velocidad del mecanógrafo, sino la del metal: las barras de tipos, al golpear dos teclas vecinas casi a la vez, se enganchaban. Sholes fue reordenando las letras para que los pares más frecuentes del inglés quedaran lejos en el teclado y sus barras subieran desde lados opuestos, sin chocar. La distribución que hoy tocan miles de millones de dedos nació de la mecánica de un martillo que ya no existe. Sholes hasta dejó un guiño comercial: colocó las letras de *typewriter* en una sola fila, para que un vendedor pudiera escribir la palabra de un golpe y deslumbrar al cliente.

Esa restricción desapareció hace más de un siglo. No hay barras, no hay enganche, no hay metal. Y sin embargo la mano sigue buscando las letras donde las puso un impresor de Wisconsin para frenar un mecanismo del XIX.

En 1936, August Dvorak patentó una distribución pensada para el mecanógrafo y no para la máquina: las letras más frecuentes bajo los dedos más fuertes, menos recorrido, menos fatiga. Sobre el papel era mejor; casi nadie lo usa. Que fuera realmente superior está discutido: los economistas Liebowitz y Margolis encontraron endeble las pruebas, producidas en buena parte por el propio Dvorak, que era juez y parte. Pero la disputa no toca lo esencial: nadie niega que la razón mecánica que dio forma a QWERTY desapareció, ni que el teclado siga ahí. Ciento cincuenta años después del atasco de palancas, seguimos escribiendo en QWERTY.

Los QWERTYs del diseño de interfaces

El diseño de interfaces tiene varios QWERTYs, y reconocerlos es el primer paso para entender por qué persisten.

El intent de cuatro categorías es un QWERTY. Nació como convención cromática en Bootstrap en 2012, respondió al problema local de necesitar clases CSS que fueran reconocibles culturalmente, y se difundió por compatibilidad con el ecosistema. Hoy es el estándar de facto en prácticamente todos los frameworks. Cambiarlo —proponer, por ejemplo,

las seis regiones afectivas que este libro derivará del espacio circunplejo de Russell— tiene un coste: hay que reescribir documentación, reentrenar a diseñadores, reconstruir sistemas de diseño corporativos. El coste crece cada año que pasa, y mientras tanto la convención original sigue vigente no porque sea la mejor, sino porque es la que está.

La ausencia de gramática temporal es otro QWERTY. La interfaz se concibe históricamente como una superficie estática sobre la que ocurren cosas, no como un medio temporal en el que los eventos son la unidad de significado. Esta concepción viene del papel, del cartel, del escritorio metáfora. Cuando la animación empezó a ser técnicamente posible, se añadió como capa decorativa sobre la interfaz estática en lugar de reformular la concepción misma de la interfaz. Hoy tenemos frameworks headless que resuelven perfectamente la estructura estática del componente pero delegan el motion al desarrollador individual, porque la concepción dominante sigue siendo que el motion es accesorio, no constitutivo.

El tratamiento del sonido como accesorio es otro QWERTY, y quizás el más costoso de los tres. La web, como medio, nació sin sonido. Las primeras implementaciones que añadieron sonido lo hicieron mal —música de fondo automática, efectos publicitarios molestos—, los navegadores reaccionaron con políticas restrictivas de autoplay, y la cultura web desarrolló una aversión generalizada al audio. Hoy, cuando la neurociencia afectiva muestra que el procesamiento auditivo es el más rápido y el más eficaz para comunicar valencia, el sonido en la web sigue siendo tratado como problema, no como canal. El camino que nos llevó hasta aquí fue local y contingente. Pero el camino ya está hecho, y abandonarlo requiere algo más que una buena razón técnica.

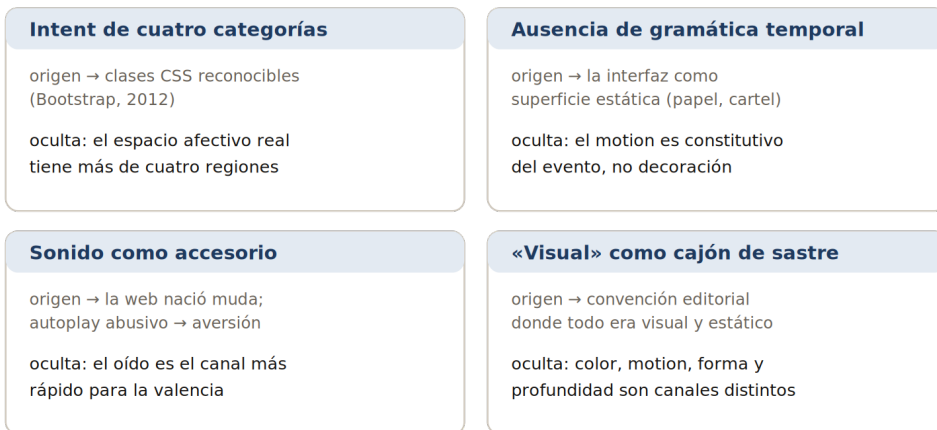
La confusión de canales perceptivos bajo la etiqueta «visual» es un cuarto QWERTY. El diseño de interfaces contemporáneo agrupa bajo un mismo término —«diseño visual»— dimensiones que el cerebro procesa

con circuitos completamente diferentes: color, motion, profundidad, forma, presencia. Esta agrupación proviene de la convención editorial y cartelista, donde efectivamente todo era visual y estático. En el medio digital multicanal, la agrupación ya no corresponde a una distinción perceptiva real, pero se mantiene porque las herramientas, las disciplinas profesionales y las convenciones formativas se organizaron alrededor de ella.

FIGURA 2.2

Los cuatro QWERTYs del diseño de interfaces

convenciones nacidas de un problema local que persisten mucho después de que su razón desapareciera



El coste de abandonarlos crece cada año; siguen vigentes no por ser los mejores, sino por ser los que están.

DISTINCIÓN

La dependencia del camino no dice que las convenciones actuales sean las mejores. Dice que lo son lo bastante como para que nadie pague el coste de cambiarlas — incluso cuando las alternativas son demostrablemente superiores.

§ 4

Carga paradigmática

Hay un tercer mecanismo, todavía más profundo, que explica por qué las convenciones del diseño de interfaces no son solo difíciles de cambiar sino incluso difíciles de ver. Lo describe con precisión Thomas Kuhn en *La estructura de las revoluciones científicas* (1962) bajo el nombre de *paradigma*, y lo que le interesa a este libro es una propiedad específica del paradigma: su carga.

Un paradigma, en la formulación de Kuhn, no es solo un conjunto de teorías o prácticas dominantes. Es un marco completo que define qué preguntas son legítimas, qué respuestas son aceptables, qué objetos son observables y qué problemas merecen ser investigados. Los practicantes de un paradigma operan dentro de él sin cuestionarlo, no porque estén prohibidos de cuestionarlo, sino porque el paradigma mismo no genera las preguntas que lo pondrían en cuestión. Las preguntas radicales son, literalmente, impensables desde dentro.

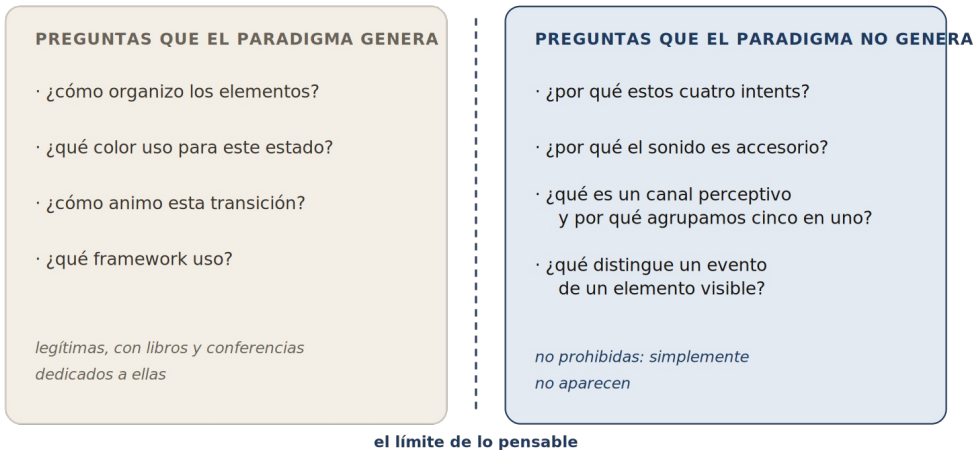
Kuhn estudiaba la ciencia, pero el concepto se aplica con notable precisión al diseño de interfaces. El paradigma actual genera preguntas como: ¿cómo organizo mejor los elementos en la pantalla? ¿Qué color uso para este estado? ¿Cómo animo esta transición? ¿Qué framework uso? Estas son preguntas legítimas: se publican libros enteros sobre ellas, existen conferencias y disciplinas profesionales organizadas alrededor de ellas.

Pero hay otro tipo de preguntas que el paradigma no genera: ¿por qué estos cuatro intents y no otros? ¿Por qué el sonido se trata como accesorio? ¿Qué es exactamente un canal perceptivo y por qué agrupamos cinco en uno? ¿Qué diferencia a un evento interactivo de un elemento visible? Estas preguntas no están prohibidas. Pero tampoco aparecen. Cuando uno examina la literatura establecida del diseño de interfaces —manuales, artículos académicos, documentación de frameworks—, estas preguntas sencillamente no están formuladas. No porque nadie se haya negado a formularlas, sino porque el paradigma no

las genera. El paradigma ha definido el campo de lo pensable, y esas preguntas caen fuera.

FIGURA 2.3

Lo que el paradigma hace pensable (y lo que no)



Un paradigma define qué preguntas son legítimas. Las preguntas radicales caen fuera del campo de lo pensable.

Los supuestos invisibles

La carga paradigmática consiste en los supuestos que el paradigma transporta sin nombrarlos. Son supuestos porque se dan por sentados. Son invisibles porque no son objeto de discusión. Y son carga porque condicionan todo lo demás sin que nadie note que lo condicionan.

El paradigma dominante del diseño de interfaces transporta, entre otros, los siguientes supuestos invisibles. Que la interfaz es fundamentalmente **visual**: el sonido y la háptica son accesorios; el tiempo es una dimensión secundaria añadida mediante animación; lo primario es la superficie visible organizada. Que el componente es la **unidad de diseño**: los eventos son cosas que les pasan a los componentes, no unidades de significado por derecho propio; una librería de componentes cubre el diseño de una interfaz, y una gramática de eventos no es necesaria.

Que las convenciones heredadas son **datos**, no hipótesis: danger es rojo, success es verde, los tooltips aparecen al hover, los modales oscurecen el fondo; estas asociaciones no se examinan, se usan. Que la accesibilidad es una **capa añadida**, no una condición constitutiva: primero se diseña la interfaz y después se añade el cumplimiento de WCAG, sin plantear la posibilidad de que la interfaz esté mal concebida desde el principio y la accesibilidad revele esa mala concepción. Y que la psicología cognitiva y la neurociencia son **material de apoyo**, no fundamento: se citan cuando refuerzan una decisión de diseño, pero el diseño no se deriva de ellas sistemáticamente.

Cada uno de estos supuestos, cuando se formula explícitamente, revela su carácter contingente. Ninguno es necesariamente verdadero. Ninguno está demostrado. Son, simplemente, los supuestos del paradigma. Pero dentro del paradigma no aparecen como supuestos: aparecen como la forma natural de pensar el diseño de interfaces.

La pregunta que el paradigma no genera

Kuhn observaba que los paradigmas no cambian por acumulación de evidencia contraria dentro del propio paradigma. Cambian cuando alguien, desde fuera o desde los márgenes, formula una pregunta que el paradigma no generaba, y esa pregunta abre un campo de investigación que el paradigma anterior no podía cubrir.

La pregunta «¿por qué *danger*, *warning*, *info*, *success* y no otro conjunto?» no es una pregunta especialmente complicada. No requiere equipamiento sofisticado ni datos inéditos. Cualquier diseñador con diez minutos de tiempo podría formulársela. Pero durante catorce años, desde la publicación de Bootstrap en 2012, prácticamente nadie la ha formulado de forma pública y sistemática.

La razón es exactamente la que Kuhn describe: el paradigma no la genera. Dentro del paradigma dominante, esos cuatro intents son *los intents*, del

mismo modo que dentro del paradigma aristotélico los cuerpos pesados caían *hacia su lugar natural*. La pregunta por la adecuación de la categoría no emerge hasta que alguien adopta un marco distinto —en este caso, la psicología afectiva del espacio circumplejo— desde el cual la pregunta se vuelve inevitable.

Esto es lo que este libro intenta hacer. No refutar el paradigma dominante en sus propios términos —eso sería inútil, porque los propios términos son el problema—, sino ofrecer un marco alternativo desde el cual las preguntas que el paradigma actual no genera se vuelvan las preguntas obvias, y las respuestas que el paradigma actual da por sentadas se vuelvan hipótesis que requieren examen.

§ 5

Por qué ahora

Si el paradigma actual del diseño de interfaces está cargado de supuestos no examinados, y si la dependencia del camino hace que estos supuestos persistan incluso cuando sus contextos originales han desaparecido, ¿por qué tiene sentido formular ahora las preguntas que este libro formula? ¿Qué ha cambiado que haga este momento propicio para un examen que hace diez años habría sido prematuro y dentro de diez podría ser tardío? Han cambiado al menos tres cosas.

La primera es que la neurociencia afectiva y la psicología perceptiva han alcanzado un nivel de maduración que hace viable fundamentar el diseño de interfaces en ellas. Las correspondencias crossmodales de Spence, el procesamiento rápido de la amígdala de LeDoux, la teoría del appraisal de Scherer, el modelo circumplejo del afecto de Russell, los esquemas sensoriomotores de Johnson: todas estas teorías tienen hoy la solidez empírica suficiente para servir de fundamento a un sistema de diseño. Hace veinte años muchas de ellas eran propuestas teóricas en discusión. Hoy son consenso razonable.

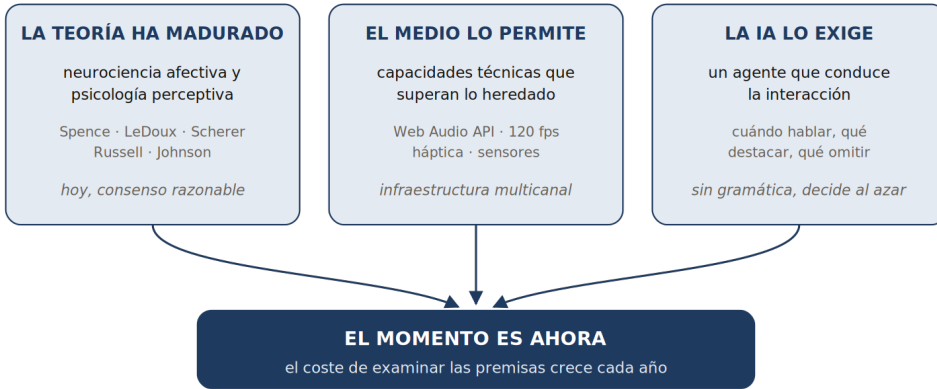
La segunda es que las capacidades técnicas del medio digital han superado con creces las limitaciones que justificaban muchas de las convenciones heredadas. La Web Audio API permite síntesis paramétrica en tiempo real. Los navegadores modernos mantienen 120 fps sin dificultad. Los dispositivos tienen háptica, giroscopios, micrófonos, sensores de luminosidad. La infraestructura existe para implementar una gramática perceptiva multicanal. Lo que falta es la teoría que diría qué implementar.

La tercera, y quizás la más urgente, es la llegada de la inteligencia artificial como agente activo en la interfaz. Cuando una IA comienza a dirigir la interacción —cuándo hablar, cómo interrumpir, qué destacar, qué omitir— necesita una gramática sobre la que operar. Sin gramática, la IA decide arbitrariamente. Con gramática, la IA opera sobre el mismo sistema perceptivo que el usuario ya ha aprendido implícitamente. La ausencia de gramática no era un problema crítico cuando las interfaces eran reactivas. Se convierte en un problema crítico cuando las interfaces son conducidas.

Estas tres circunstancias no van a esperar. La infraestructura técnica seguirá avanzando, la IA seguirá integrándose en los productos, y la sedimentación seguirá añadiendo capas. El coste de examinar las premisas crece con cada año que pasa. La ventana para formular estas preguntas sin que resulten nostálgicas o irrelevantes es ahora.

FIGURA 2.4

Por qué ahora: tres condiciones convergen



Tres circunstancias que hace diez años no coincidían y que no van a esperar.

Lo que este libro no propone

Conviene cerrar este capítulo aclarando qué no propone este libro, para que el lector sepa desde el principio lo que puede esperar.

No propone descartar las convenciones actuales. Muchas de ellas capturan intuiciones perceptivas correctas y deben conservarse, aunque con fundamentación que antes no tenían. El rojo para danger funciona — pero funciona porque activa una región del espacio afectivo de alta valencia negativa y alto arousal, no porque el semáforo lo dijera.

No propone un nuevo framework que reemplace a los existentes. Los frameworks son implementaciones; este libro es una teoría sobre la que pueden construirse múltiples implementaciones. El objetivo es el marco conceptual, no el producto técnico.

No propone que el diseño de interfaces deba volverse una disciplina académica con métodos experimentales obligatorios. Propone que las convenciones que se usan a diario sean examinables: que exista un

vocabulario compartido para preguntar, cuando alguien toma una decisión de diseño, por qué la toma.

Lo que sí propone es someter a examen las premisas sobre las que opera la disciplina. Preguntar por qué damos por sentado lo que damos por sentado. Y, cuando la respuesta sea «porque siempre se ha hecho así», preguntar si las razones originales todavía se sostienen, o si estamos, como los mecanógrafos de 1950, escribiendo en QWERTY sobre palancas que ya no se atascan.

PRINCIPIO

La pregunta que el paradigma no genera es, precisamente, la pregunta que hay que formular. No porque todas las respuestas estén equivocadas, sino porque las respuestas correctas merecen saberse correctas, y las incorrectas merecen dejar de serlo.

03

Fundamentos psicoperceptivos de la interfaz

De qué está hecha la percepción interactiva

PREGUNTA DEL CAPÍTULO

¿Qué sabe la percepción humana que el diseño de interfaces todavía no ha incorporado a su lenguaje?

Mira un botón y, sin proponértelo, ya sabes que se puede pulsar. Aparece un aviso en rojo y tu atención salta hacia él antes de leerlo. Algo se mueve en el borde de la pantalla y tu mirada lo persigue. Nada de eso es una decisión: es percepción, ocurriendo por debajo del pensamiento. Antes de diseñar lo que una interfaz dice, conviene entender cómo el usuario ve, atiende y evalúa sin darse cuenta.

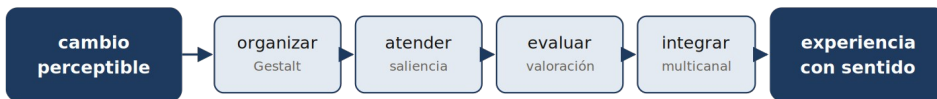
Si este libro sostiene que la interfaz debe entenderse desde sus eventos, no puede apoyarse solo en vocabulario de diseño, convenciones de framework o intuiciones de implementación. Necesita preguntarse cómo

perciben las personas. Cómo organizan un campo visual, cómo detectan cambios, cómo orientan la atención, cómo atribuyen causalidad. Cómo evalúan una situación, cómo integran señales de varios canales y cómo convierten una acción en experiencia significativa.

Una interfaz no se comprende solo como imagen, ni solo como estructura, ni solo como código. Se comprende como un campo perceptivo en el que un usuario atiende, actúa, evalúa, predice, recuerda e integra señales distribuidas entre varios canales.

FIGURA 3.1

El campo perceptivo: de un cambio a una experiencia con sentido



La interfaz no es solo imagen o código: es un campo donde el usuario atiende, evalúa e integra señales.

§ 1

Por qué necesitamos fundamentos perceptivos

El diseño de interfaces ha construido una tradición práctica muy sólida. Sabemos organizar jerarquías visuales, usar contraste, agrupar elementos, diseñar componentes y sistemas. Pero si la interfaz hace perceptibles eventos, necesitamos explicar cómo un usuario percibe que algo ha ocurrido, que su acción fue registrada, que un estado quedó fijado, que

una operación sigue viva, que una alerta reclama atención o que una pérdida ya se produjo.

PRINCIPIO

Si la interfaz es perceptiva, su gramática no puede depender solo de convenciones visuales. Debe apoyarse en cómo las personas perciben, atienden, actúan y evalúan cambios.

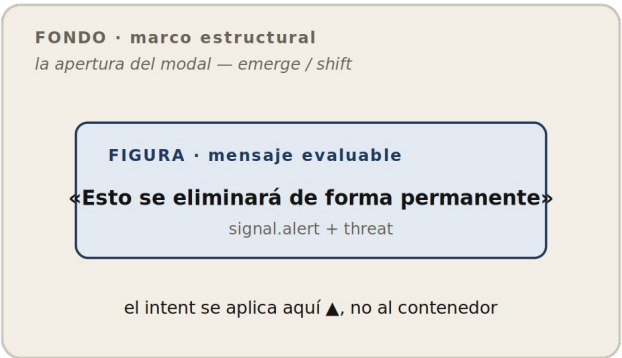
§ 2

Organización perceptiva: Gestalt, agrupación y figura/fondo

Un usuario no percibe una interfaz como suma neutra de elementos. La percibe como un campo organizado: unas cosas pertenecen juntas, otras quedan separadas; unas actúan como figura, otras como fondo. La revisión de Wagemans y colaboradores sobre un siglo de psicología Gestalt resume fenómenos como agrupación, organización figura/fondo, continuidad y cierre como principios centrales de la percepción visual.

Esto prepara una distinción esencial: hay eventos que son figura y eventos que son fondo. Un error dentro de un modal puede ser figura evaluable. La apertura del modal puede ser fondo estructural. Si no distinguimos ambas cosas, terminamos coloreando el contenedor en lugar del mensaje.

FIGURA 3.2
Figura y fondo: el intent vive en el mensaje, no en el marco



Si no separamos figura de fondo, terminamos coloreando el contenedor en lugar del mensaje.

§ 3

Affordances y acción situada

Una interfaz no es solo algo que el usuario observa. Es un campo de posibilidades. Gibson formuló la noción de affordance para nombrar lo que el entorno ofrece al organismo en términos de acción posible. Un botón ofrece ser pulsado. Un slider ofrece ser arrastrado. Un modal comunica que ciertas acciones quedan suspendidas y otras adquieren prioridad.

TABLA 3.1 Affordances y eventos

AFFORDANCE	QUÉ OFRECE	EVENTO	LECTURA
Presionar	Actuar sobre algo	contact	El sistema ha sentido mi acción
Elegir	Fijar una opción	commit.select	Esto queda elegido
Manipular	Mover directamente	handle	Estoy controlando este objeto
Atender	Mirar o responder	signal	Esto requiere mi atención
Entrar/salir	Cambiar	emerge / shift	Algo apareció /

	disponibilidad		cambió el marco
Esperar	Sostener proceso	sustain	Esto sigue ocurriendo

§ 4

Atención, saliencia y carga cognitiva

Una interfaz siempre compite por recursos limitados. Cada señal, cada movimiento, cada sonido y cada color saturado tiene un coste. Posner estudió la orientación de la atención como un proceso que asigna recursos de procesamiento a una parte del campo perceptivo, incluso antes de que haya movimiento ocular explícito.

No todo evento debe reclamar atención. Un `signal.alert` sí puede justificar interrupción. Un `emerge.reveal` quizá solo debe hacer aparecer contenido. Un `sustain.progress` debe mantener conciencia de proceso sin competir con la tarea principal.

FIGURA 3.3

Saliencia: cuánta atención debe reclamar cada evento



No todo evento debe reclamar atención: la saliencia ha de ser proporcional a lo que está en juego.

§ 5

Valoración del evento: cómo el usuario evalúa lo que ocurre

Algunos cambios no solo se perciben. Se evalúan: ¿esto es bueno o malo para mí?, ¿favorece mi meta?, ¿bloquea mi acción?, ¿puedo corregirlo?, ¿requiere respuesta inmediata? Scherer describe el proceso emocional como una secuencia de evaluaciones en términos de relevancia, implicaciones, capacidad de afrontamiento y significado normativo.

La valoración del evento permite formular una distinción que después será central: no todos los eventos son evaluables. Algunos son mensaje (fallo, advertencia, confirmación, pérdida). Otros son marco (apertura, expansión, navegación, sostenimiento). El intent tiene sentido cuando hay algo que evaluar.

PRINCIPIO

El intent no nace del color. Nace de la evaluación de un evento respecto a metas, riesgo, control y consecuencia.

§ 6

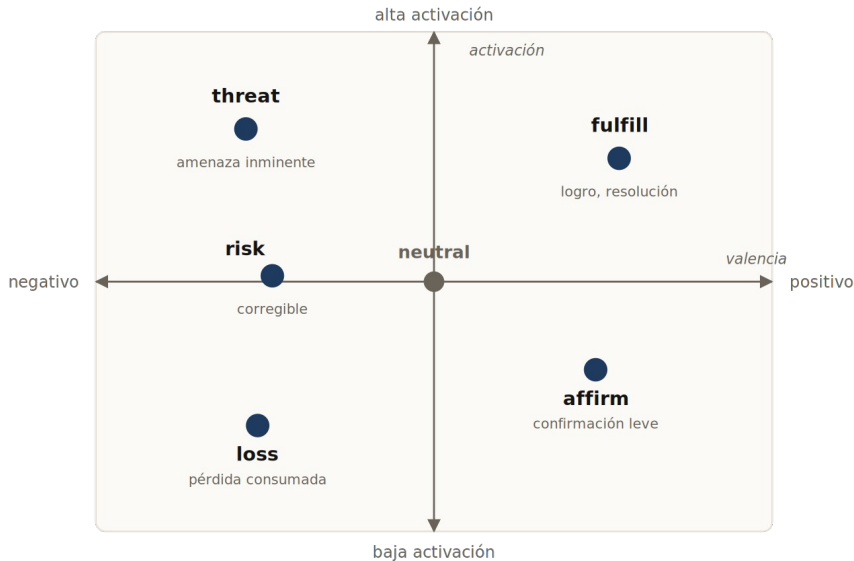
Valencia y activación: el espacio afectivo

Si un evento es evaluable, necesitamos describir de qué tipo de evaluación se trata. El modelo circumplejo de Russell propone dos ejes: valencia (positivo/negativo) y activación (baja/alta activación). Esto permite distinguir regiones que el semáforo heredado (success/warning/danger/info) mezcla.

No es lo mismo threat (negativo, alta activación) que loss (negativo, baja activación). No es lo mismo affirm (positivo leve) que fulfill (positivo, activación media-alta). Esta distinción permite formular los intents no como colores, sino como regiones evaluativas del evento.

FIGURA 3.4

El espacio afectivo: los seis intents como regiones evaluativas



Distingue regiones que el semáforo mezcla: threat (negativo, alta activación) ≠ loss (negativo, baja).

§ 7

Movimiento, causalidad y continuidad

En diseño de interfaces solemos hablar de «animación» como recurso visual añadido. Pero para esta gramática conviene distinguir animación de motion. La animación es la implementación. Motion es la percepción de que algo se mueve, se comprime, aparece, se desplaza o cambia de régimen en el tiempo.

Michotte mostró que ciertos patrones simples de movimiento producen una impresión directa de causación: si una respuesta ocurre cerca de la acción del usuario y con forma temporal adecuada, puede percibirse como consecuencia de esa acción. Si llega tarde o con forma incongruente, se percibe como evento separado.

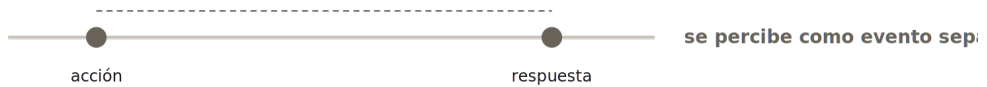
FIGURA 3.5

Causalidad percibida: el tiempo decide si la respuesta es consecuencia

RESPUESTA INMEDIATA



RESPUESTA TARDÍA



Si la respuesta llega cerca de la acción y con forma temporal adecuada, el usuario la lee como su consecuencia.

§ 8

Sonido, multisensorialidad e integración de canales

El sonido tiene propiedades que otros canales no tienen: puede percibirse sin mirar la pantalla, puede comunicar urgencia rápidamente, puede marcar cierre, confirmación o pérdida, y puede funcionar como canal de respaldo cuando motion o visión están reducidos. Spence revisa cómo ciertas correspondencias crossmodales — pitch con tamaño, brillo con peso, movimiento ascendente con resolución — aparecen de forma consistente.

PRINCIPIO

El problema histórico no ha sido el sonido en la interfaz. Ha sido el sonido sin gramática: arbitrario, desproporcionado o desacoplado del evento.

El usuario no percibe motion, sonido, color y háptica como departamentos aislados. Wallace y colaboradores muestran que la capacidad del cerebro para ligar estímulos de distintas modalidades depende de su estructura

temporal. Si el sonido llega demasiado tarde respecto al movimiento, el usuario percibe dos eventos, no uno. Una interacción semántica multicanal necesita congruencia temporal, de intensidad, de dirección y afectiva.

FIGURA 3.6

Integración multicanal: la congruencia temporal une los canales

CANALES ALINEADOS



CANALES DESFASADOS



Si el sonido llega demasiado tarde respecto al movimiento, el cerebro percibe dos sucesos en lugar de uno.

§ 9

Esquemas sensoriomotores y cognición corporeizada

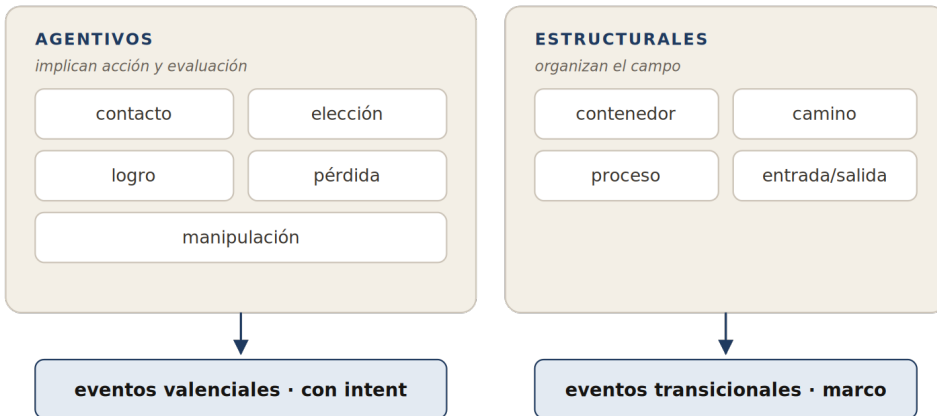
Muchas categorías que usamos para pensar interfaces proceden de patrones corporales básicos: contacto, presión, contenedor, camino, entrada, salida, peso, resistencia, pérdida, agarre, encaje, proceso. No las usamos solo como metáforas decorativas. Funcionan porque la interfaz digital se apoya en esquemas que el usuario ya conoce por su experiencia corporal.

Algunos esquemas son agentivos e implican acción y evaluación: contacto, elección, logro, pérdida, manipulación. Otros son estructurales y organizan

el campo: contenedor, camino, proceso, entrada/salida. Esta diferencia prepara la distinción entre eventos valenciales y transicionales.

FIGURA 3.7

Esquemas corporales: dos familias de origen



Los esquemas agentivos preparan los eventos evaluables; los estructurales, los que organizan el campo.

§ 10

Accesibilidad como prueba de arquitectura perceptiva

Si una interfaz distribuye significado entre varios canales, debe preguntarse qué ocurre cuando uno se reduce. Si el color no basta, debe haber forma, texto o posición. Si se reduce motion, la causalidad debe apoyarse en otros recursos. Si el sonido está desactivado, una alerta debe seguir siendo perceptible.

PRINCIPIO

La accesibilidad no solo adapta la interfaz. Revela si el significado estaba bien distribuido entre canales.

§ 11

Cómo usará este libro estos fundamentos

TABLA 3.2 Mapa de fundamentos y función en el libro

FUNDAMENTO	QUÉ EXPLICA	QUÉ APORTA A LA TEORÍA
Gestalt	Agrupación, figura/fondo	Diferencia entre marco y mensaje
Affordances	Percepción de acción posible	Contexto operativo, contacto, manipulación
Atención	Saliencia, orientación, carga	Signal, proporción, interrupción
Valoración del evento	Evaluación respecto a metas	Semánticas valenciales/transicionales
Valencia/activación	Estructura afectiva básica	Sistema de intents
Motion	Causalidad, continuidad, energía	Temporalidad y física del evento
Sonido	Urgencia, valencia, cierre	Microeventos auditivos
Multisensorialidad	Integración temporal de señales	Evento multicanal unificado
Esquemas corporales	Contacto, camino, contenedor, pérdida	Familias semánticas
Accesibilidad	Reducción o sustitución de canales	Migración semántica

§ 12

El espacio evaluativo: valencia y activación

Hasta ahora hemos descrito cómo el usuario percibe, atiende y actúa. Falta una dimensión esencial: cómo evalúa lo que ocurre.

Cuando un evento sucede, el usuario no solo lo reconoce. También lo interpreta: ¿esto es bueno o malo para mí? ¿requiere acción inmediata? ¿puedo ignorarlo? ¿ya ocurrió o aún puedo evitarlo? ¿se resolvió algo que estaba pendiente?

Una forma útil de organizar estas evaluaciones es el espacio propuesto por James A. Russell, que describe los estados afectivos mediante dos dimensiones: valencia (positivo / negativo) y activación (baja / alta).

Este modelo no pretende capturar toda la complejidad emocional. Pero resulta suficiente para distinguir diferencias que una interfaz necesita comunicar de forma consistente.

Por ejemplo: no es lo mismo algo negativo urgente que algo negativo ya ocurrido; no es lo mismo una confirmación leve que una culminación; no es lo mismo informar que advertir; no es lo mismo pedir atención que registrar una pérdida.

Estas diferencias no son estilísticas. Son operativas.

Este libro utilizará ese espacio como base para definir los intents, no como una teoría psicológica completa, sino como una estructura mínima para organizar la evaluación del evento.

Síntesis

La interfaz se percibe como campo organizado: por eso necesitamos Gestalt y figura/fondo. Se usa como campo de acción: por eso necesitamos affordances. Compite por atención limitada: por eso necesitamos saliencia y carga cognitiva. Comunica resultados, riesgos y pérdidas: por eso necesitamos valoración del evento. Modula tono afectivo: por eso necesitamos valencia y activación. Cambia en el tiempo: por eso necesitamos motion. Puede sonar, vibrar, moverse y cambiar de presencia a la vez: por eso necesitamos multisensorialidad. Se apoya en experiencia corporal: por eso necesitamos esquemas. Y debe seguir siendo legible cuando un canal falta: por eso la accesibilidad es constitutiva, no añadida.

Estos fundamentos no pretenden probar una taxonomía definitiva ni dictar cada valor concreto de duración, frecuencia o intensidad. La ciencia no sustituye al diseño. Lo disciplina. Le impide confundir color con intent,

motion con decoración, sonido con accesorio y evento técnico con significado experiencial.

Una gramática semántica de la interfaz debe apoyarse en cómo los seres humanos perciben cambios, actúan sobre posibilidades, atienden señales, evalúan consecuencias e integran canales.

El siguiente capítulo podrá volver ahora al concepto central del libro: el evento como unidad básica de cambio. Ya no partiremos solo de una intuición práctica. Tendremos un mapa de fundamentos que explica por qué el evento, y no solo el componente, puede convertirse en la unidad central de una gramática perceptiva de la interfaz.

Estos fundamentos no dictan una taxonomía cerrada. No prueban que solo existan ocho familias ni seis intents. Lo que hacen es otra cosa: ofrecen criterios para que las categorías del libro no parezcan arbitrarias.

A partir de aquí, cuando hablemos de evento, gramática, interacción semántica, familias, intent y canales, no estaremos hablando de nombres inventados desde el diseño. Estaremos nombrando formas recurrentes de percibir, actuar, atender, evaluar e integrar cambios.

DISTINCIÓN

Este libro no inventa categorías perceptivas. Les da nombre dentro de la interfaz.

04

El evento como unidad básica de cambio

Qué cuenta como cambio significativo en una interfaz

PREGUNTA DEL CAPÍTULO

¿Qué cuenta como evento en una interfaz, y qué lo distingue de un simple cambio de estado?

Pulsas una tecla y aparece una letra. ¿Eso es un evento? El cursor parpadea cada segundo: ¿y eso? El reloj de la esquina avanza un minuto, una notificación se desliza, un botón cambia de tono al pasar el ratón. Algunas de esas cosas importan a lo que el usuario está haciendo; otras son ruido de fondo. Distinguir cuáles cuentan como cambio es el problema de este capítulo.

La definición de evento se apoya en los fundamentos anteriores. Desde la organización perceptiva, un evento modifica el campo: algo aparece, cambia de figura, se subordina. Desde las affordances, un evento modifica lo que el usuario puede hacer. Desde la atención, un evento puede orientar, reclamar o liberar foco. Desde la valoración del evento, un

evento puede volverse evaluable: favorable, riesgoso, urgente, perdido o resuelto. Desde la multisensorialidad, un evento puede distribuirse entre varios canales y aun así percibirse como una sola unidad.

Por eso un evento no equivale a un click, a un callback ni a un cambio interno de estado. Un evento existe cuando una modificación perceptible altera la relación operativa entre usuario y sistema.

El capítulo anterior propuso una redefinición inicial: la interfaz no es solo una superficie de componentes, sino la dimensión perceptible de un contexto operativo. Desde ahí aparecía una consecuencia inevitable: si la interfaz no solo muestra cosas, sino que también hace ocurrir transformaciones, entonces la unidad básica de cambio no puede buscarse únicamente en el componente. Tiene que buscarse en el cambio. Tiene que buscarse en el evento.

Su objetivo no es todavía cerrar la taxonomía de familias semánticas. Tampoco entrar en canales de expresión, intents o valores concretos. Aquí necesitamos algo más básico: entender qué es un evento, qué lo diferencia de una acción técnica, de un estado, de una variación visual, de un efecto y de una consecuencia.

¿Cuándo algo que cambia en una interfaz merece ser tratado como evento significativo?

§ 1

El cambio no basta

En una interfaz pasan muchas cosas. Un botón cambia de color al pasar el cursor. Un icono modifica su opacidad. Una lista se reordena. Un menú aparece. Un campo muestra un error. Un spinner empieza a girar. Un toast entra desde la esquina. Una tarjeta desaparece. Un sonido breve acompaña una confirmación. Una vibración responde al contacto. Una vista sustituye a otra.

Podríamos llamar «evento» a todo eso. Pero esa definición sería demasiado amplia. No todo cambio perceptible es un evento en sentido fuerte. Algunos cambios son ornamentales. Otros son puramente mecánicos. Otros son variaciones locales sin relevancia para la acción del usuario.

La pregunta no es simplemente: ¿algo ha cambiado? La pregunta es: ¿ese cambio modifica lo que el usuario entiende que está ocurriendo?

PRINCIPIO

Un evento no es cualquier cambio perceptible. Es un cambio perceptible con relevancia operativa.

§ 2

Variación, efecto y evento

Una variación es un cambio perceptible que todavía no compromete significado. Un botón que cambia su sombra al hover puede indicar interactividad, pero si no altera el estado ni comunica una acción registrada, no es todavía un evento pleno.

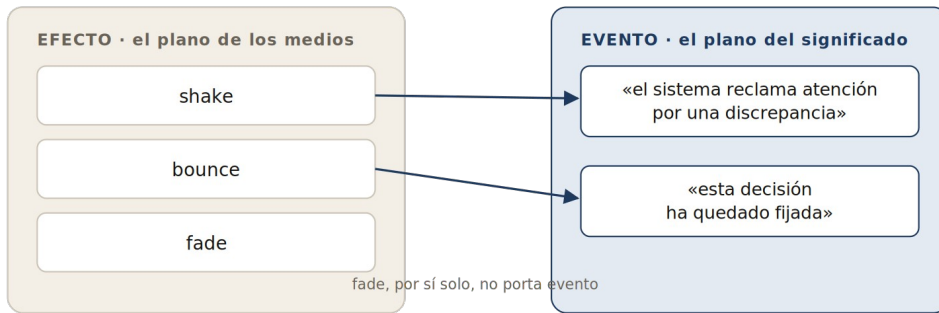
Un efecto es una realización perceptiva concreta: una animación, un sonido, una vibración, un cambio cromático, un fade, un bounce, un shake. El efecto pertenece al plano de los medios.

Un evento pertenece al plano del significado operativo. No se define por la forma que toma, sino por la transformación que vuelve legible.

Un shake puede ser un efecto. Pero «el sistema reclama atención por una discrepancia» es un evento. Un bounce puede ser un efecto. Pero «esta decisión ha quedado fijada» es un evento.

FIGURA 4.1

Variación, efecto y evento



El efecto es forma; el evento es significado. El mismo efecto puede portar un evento o no portarlo.

§ 3

El evento no está en el código

Un click solo dice que ocurrió una activación de entrada. No dice qué significa esa activación para el usuario. El mismo click puede abrir un menú, confirmar una eliminación, seleccionar una opción, iniciar una carga, enviar un formulario, cerrar un diálogo o navegar a otro destino. Desde el código, todos empiezan igual. Desde la experiencia, no tienen nada que ver.

TABLA 4.1 Tres planos del evento

PLANO	QUÉ REGISTRA	EJEMPLO
Evento técnico	Una ocurrencia del sistema o dispositivo	click, submit, change, dragstart
Evento	Una transformación significativa para el usuario	Contacto, consolidación, señal, aparición
Interacción semántica	Una forma recurrente de hacer legible esa transformación	contact, commit, signal, handle, emerge

DISTINCIÓN

Un evento técnico ocurre en el sistema. Un evento ocurre para el usuario.

§ 4

El evento como transformación del contexto operativo

Un evento es una modificación perceptible del contexto operativo que tiene significado para la acción, la atención o la interpretación del estado.

Esta definición contiene tres condiciones. La primera: una modificación perceptible. Algo debe cambiar en el campo perceptivo. La segunda: relevancia operativa. El cambio debe afectar a lo que el usuario puede hacer, debe atender o necesita interpretar. La tercera: debe pertenecer a una clase reconocible — contacto, consolidación, señal, manipulación, aparición, cambio de contexto, sostenimiento.

FIGURA 4.2

Las tres condiciones que convierten un cambio en evento



CLASES RECONOCIBLES



No basta con que algo cambie: debe ser relevante para la acción y pertenecer a una clase reconocible.

Un botón que se comprime al ser pulsado comunica contacto. Un switch que cambia de estado comunica consolidación. Un modal que se abre

comunica cambio de contexto. Un spinner que aparece comunica sostenimiento. Un elemento que desaparece comunica pérdida o retirada. En todos estos casos, lo importante no es que algo cambie en pantalla. Lo importante es que el cambio reorganiza la relación entre usuario y sistema.

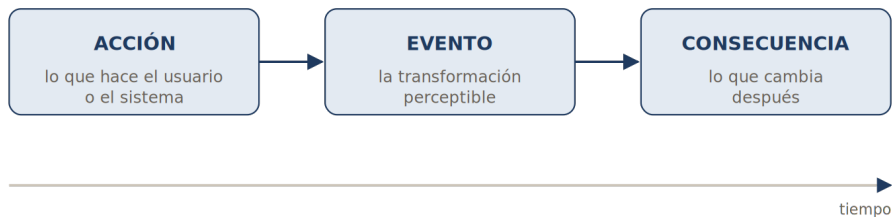
§ 5

Acción, evento y consecuencia

La acción es lo que hace el usuario o el sistema. El evento es la transformación perceptible que esa acción produce. La consecuencia es lo que cambia después en el contexto operativo.

FIGURA 4.3

Acción, evento y consecuencia



No siempre coinciden en el tiempo:

- a veces la acción inicia un proceso y la consecuencia llega después;
- a veces el sistema produce un evento sin acción del usuario (una alerta, una sesión expirada).

La acción es la causa; el evento, su huella perceptible; la consecuencia, el nuevo estado del contexto operativo.

Acción, evento y consecuencia no siempre coinciden en el tiempo. A veces la acción y el evento están muy próximos. A veces la acción inicia un proceso. A veces la consecuencia llega después. A veces el sistema produce un evento sin acción inmediata del usuario: una notificación, una alerta, una sesión expirada.

§ 6

Estado y evento: dos formas de sentido

Un estado describe una condición relativamente estable: seleccionado, abierto, cargando, inválido. Un evento describe una transformación: seleccionar, abrir, empezar a cargar, fallar. Los estados orientan. Los eventos explican cómo se llegó a esa orientación. El usuario necesita ambos.

TABLA 4.2 Estado, evento y consecuencia

PLANO	PREGUNTA	EJEMPLO	RIESGO SI FALTA
Estado	¿En qué condición está?	El switch está encendido	No sabe cómo actuar ahora
Evento	¿Qué cambio ocurrió?	El switch acaba de activarse	No sabe si su acción tuvo efecto
Consecuencia	¿Qué implica?	La preferencia ya se aplicó	No entiende el alcance del cambio

Una interfaz centrada solo en estados puede mostrar el resultado final, pero dejar pobremente articulado el cambio que llevó hasta él. Una interfaz centrada solo en eventos puede volverse efímera. La gramática necesita ambos planos.

§ 7

El evento tiene forma temporal

Un evento no es solo una etiqueta. Tiene forma en el tiempo. Algunos son casi puntuales. Otros tienen fase de entrada y salida. Otros son sostenidos. Otros tienen estructura interna compleja.

FIGURA 4.4

La forma temporal del evento

cada familia tiene un perfil en el tiempo; la temporalidad es parte de la semántica, no decoración



Un contacto se siente como ataque; una señal, como irrupción; un sostenimiento, como continuidad.

Un contacto se siente como ataque. Una consolidación se siente como asentamiento. Una señal se siente como irrupción. Una aparición se siente como entrada de presencia. Un cambio de contexto se siente como cruce de umbral. Un sostenimiento se siente como continuidad. Una manipulación se siente como trayectoria.

La temporalidad no es decoración. Es parte de la semántica.

§ 8

El evento tiene granularidad

Los eventos pueden observarse a distintas escalas. A escala pequeña, un botón responde al contacto. A escala media, una operación se inicia. A escala de flujo, un usuario completa un proceso. La gramática necesita permitir todos los niveles, pero sin mezclarlos.

Conviene distinguir entre microeventos (cambios inmediatos), eventos compuestos (varias fases), secuencias semánticas (encadenamientos de

familias distintas) y resultados de la acción (estabilización del nuevo contexto).

PRINCIPIO

No todos los eventos tienen el mismo tamaño. Algunos son microeventos; otros son secuencias completas de transformación.

§ 9

El evento tiene origen

Un evento puede estar iniciado por el usuario o por el sistema. Cuando el usuario inicia, la interfaz debe preservar la relación causal entre acción y respuesta. Cuando el sistema inicia, la cuestión principal es atención: el usuario necesita saber por qué algo aparece o por qué se interrumpe el flujo.

TABLA 4.3 Origen del evento

ORIGEN	PREGUNTA DEL USUARIO	EJEMPLO	RIESGO
Usuario	¿Mi acción tuvo efecto?	Pulsar, arrastrar, seleccionar	Pérdida de agencia
Sistema	¿Por qué ocurre esto ahora?	Notificación, alerta, sesión expirada	Interrupción o sorpresa
Mixto	¿Qué parte depende de mí?	Guardar con validación, subir archivo	Ambigüedad causal

§ 10

El evento tiene dirección

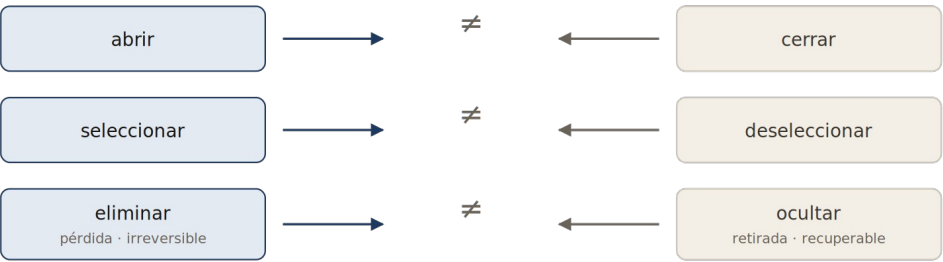
Algunos eventos son reversibles. Otros no. Algunos abren posibilidades. Otros las cierran. Abrir no significa lo mismo que cerrar. Seleccionar no significa lo mismo que deseleccionar. Eliminar no significa lo mismo que ocultar.

Muchas implementaciones tratan estas parejas como simples reversos animados. Pero perceptivamente eso es insuficiente. Entrar y salir no son la misma narración en sentido contrario.

FIGURA 4.5

El evento tiene dirección

entrar y salir no son la misma narración en sentido contrario



Tratar estas parejas como simples reversos animados pierde una diferencia semántica real.

§ 11

El evento tiene valencia o estructura

Algunos eventos son evaluables. Comunican algo que puede leerse como bueno, malo, urgente, grave, reparable o perdido. Una confirmación puede ser positiva. Una alerta puede ser amenazante. Un fallo puede exigir corrección.

Otros eventos son estructurales. No comunican por sí mismos una evaluación, sino una reorganización del campo perceptivo: algo aparece, se expande, cambia de plano, desplaza el foco, sostiene una operación.

Hay eventos que son mensaje. Hay eventos que son marco.

TABLA 4.4 Eventos evaluables y eventos estructurales

TIPO	QUÉ HACE	EJEMPLO	¿ACEPTA INTENT?
------	----------	---------	-----------------

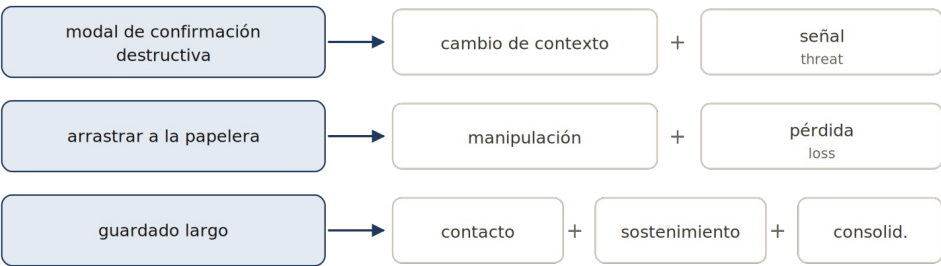
Evaluable	Comunica resultado, riesgo, pérdida o logro	Confirmación, alerta, error, eliminación	Sí
Estructural	Organiza presencia, foco, marco o continuidad	Abrir, expandir, navegar, sostener	Normalmente no
Compuesto	Combina marco y mensaje	Modal de error, wizard con validación	En el mensaje, no en el marco

§ 12

El evento puede ser simple o compuesto

Muchos eventos reales no pertenecen a una sola familia de forma pura. Son composiciones. Un modal de confirmación destructiva combina cambio de contexto y señal de amenaza. Un drag hacia una papelera combina manipulación y pérdida. Un guardado largo combina contacto, sostenimiento y consolidación.

FIGURA 4.6
El evento puede ser simple o compuesto



Un evento compuesto debe poder descomponerse en eventos simples sin perder la unidad de experiencia.

Un evento compuesto debe poder descomponerse en eventos simples sin perder la unidad de experiencia.

§ 13

El evento tiene huella

Un evento no termina siempre cuando acaba su animación. A veces deja un estado. A veces deja una marca visual. A veces deja una obligación. A veces deja una pérdida. A veces deja una posibilidad nueva.

Conviene distinguir entre duración expresiva (el tiempo durante el cual el evento se vuelve perceptible), estado resultante (la condición que queda después), resultado de la acción (lo que cambia en el campo de acción) y huella semántica (el modo en que el evento sigue importando después de ocurrir).

PRINCIPIO

La duración de un evento no equivale a su importancia. Lo decisivo es la consecuencia que deja en el contexto operativo.

§ 14

El evento puede fallar

Un evento falla cuando la transformación que intenta comunicar no llega al usuario con la claridad adecuada. Puede fallar por latencia, por indiferenciación, por exceso, por ambigüedad, por canal único, por incongruencia multicanal o por falsa valencia.

Estos fallos muestran que la semántica no es un lujo. Es una forma de evitar ambigüedad, exceso y pérdida de continuidad. Una interfaz puede funcionar técnicamente y fallar semánticamente: ejecutar la acción correcta y aun así dejar al usuario con la pregunta «¿qué acaba de pasar exactamente?»

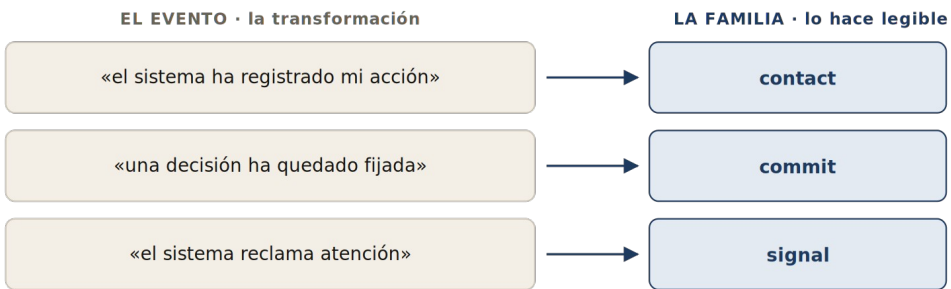
§ 15

Del evento a la interacción semántica

El evento es la transformación significativa. La interacción semántica es la forma recurrente de hacerla legible. El evento de que el sistema ha registrado una acción se hace legible mediante la familia de contacto. El evento de que una decisión ha quedado fijada se hace legible mediante la familia de consolidación. El evento de que el sistema reclama atención se hace legible mediante la familia de señal.

FIGURA 4.7

Del evento a la interacción semántica



El evento es la transformación significativa; la familia es la forma recurrente de hacerla legible.

§ 16

Resumen operativo

No estamos diciendo que cada microcambio deba convertirse en categoría. No estamos diciendo que toda animación sea un evento. No estamos diciendo que la interfaz deba explicar explícitamente cada transformación. No estamos diciendo que el usuario deba aprender una taxonomía consciente.

Cuando una transformación afecta a la acción, la atención o la interpretación del estado, la interfaz debería tener una forma consistente de hacerla perceptible. Eso es todo.

Una gramática de eventos no obliga a expresarlo todo con más fuerza. Al contrario, ayuda a decidir qué merece expresión y qué debe permanecer silencioso.

§ 17

Síntesis

El evento es la unidad básica de cambio significativo en una interfaz. No coincide con el componente, ni con el estado, ni con el evento técnico, ni con el efecto.

Tabla de síntesis del capítulo

El capítulo anterior redefinió la interfaz. Este capítulo ha definido el evento. El siguiente paso será explicar por qué esos eventos necesitan una gramática: un sistema estable de diferencias que permita reconocerlos, componerlos y expresarlos sin depender de decisiones locales aisladas.

La pregunta ya no será solo: ¿qué evento ha ocurrido? Sino: ¿cómo puede una interfaz hacer que esa clase de evento sea reconocible cada vez que ocurre?

TABLA 4.5 Síntesis del capítulo: qué convierte un cambio en evento

IDEA	SÍNTESIS
El cambio no basta	Solo hay evento cuando el cambio importa operativamente
El efecto no basta	Una animación o sonido no son semánticos por sí mismos

El código no basta	Un evento técnico no define el significado experiencial
El estado no basta	El usuario necesita comprender la transformación
La duración no basta	La importancia depende de la consecuencia
La semántica organiza	Una interacción semántica hace legible una clase de evento

05

Por qué una gramática de eventos

De la intuición a la estructura

PREGUNTA DEL CAPÍTULO

¿Por qué los eventos de una interfaz necesitan una gramática y no solo una colección de patrones?

En una misma aplicación, guardar parpadea en verde en una pantalla, muestra un texto discreto en otra y no dice nada en una tercera. Cada solución funciona por separado; juntas, obligan al usuario a reaprender en cada rincón. Lo que falta no es más diseño: es un acuerdo sobre qué significa cada cosa.

El capítulo anterior nos dejó en un punto delicado. Ya no estamos hablando solo de componentes. Tampoco solo de estados. Hemos empezado a llamar evento a esa clase de cambio que el usuario percibe y que importa para lo que hace, para lo que atiende o para cómo interpreta el estado.

Esa definición cambia el problema. Si la interfaz no solo muestra cosas, sino que hace perceptibles transformaciones, entonces no basta con preguntarse qué elementos hay en pantalla. Hay que preguntarse qué tipos de cambio ocurren, cómo se distinguen, cómo se repiten y cómo el usuario aprende a reconocerlos.

Si cada una de esas transformaciones se resuelve de forma local, la interfaz puede seguir funcionando, pero su lenguaje queda fragmentado. Cada componente decide cómo responder. Cada pantalla decide cómo animar. Cada equipo decide cuándo usar color, sonido, vibración o silencio. El usuario, entonces, no aprende una estructura común: interpreta caso por caso.

¿Puede una interfaz tener una gramática de sus eventos?

§ 1

Por qué la palabra gramática

La palabra «gramática» no se usa aquí por adorno. Tampoco pretende trasladar de forma literal la gramática de una lengua natural al diseño de interfaces. Se usa porque necesitamos nombrar algo que las palabras «estilo», «patrón», «efecto», «animación» o «taxonomía» no terminan de cubrir.

Una taxonomía clasifica. Una guía de estilo normaliza la forma. Un patrón resuelve un problema recurrente. Un efecto materializa una respuesta perceptiva concreta. Una gramática, en cambio, define un sistema de diferencias y relaciones que permite producir significado reconocible. No se limita a decir qué unidades existen. También define cómo se distinguen, cómo se combinan, cuándo una forma cambia de sentido, qué variaciones son aceptables y qué combinaciones producen ruido o ambigüedad.

TABLA 5.1 Por qué gramática y no otro concepto

CONCEPTO	QUÉ HACE	POR QUÉ NO BASTA
Efecto	Materializa una respuesta perceptiva	Puede existir sin significado estable
Patrón	Resuelve un problema recurrente de interacción	Suele estar ligado a casos concretos
Guía de estilo	Normaliza la forma visual	No siempre explica el significado del cambio
Taxonomía	Clasifica tipos de evento	No define composición, modulación ni canal
Gramática	Define diferencias, relaciones y reglas de combinación	Es el nivel necesario para articular eventos como lenguaje

§ 2

Qué define una gramática de eventos

La interfaz ya tiene gramáticas parciales. Tiene una gramática visual: jerarquía, contraste, alineación, proximidad. Tiene una gramática de componentes: botones, campos, menús. Tiene una gramática de estados: hover, focus, active, disabled. Pero todavía no suele tener una gramática explícita del evento: una forma sistemática de distinguir entre tocar, fijar, advertir, emerger, cambiar de marco, manipular, sostener, completar o perder.

Definición

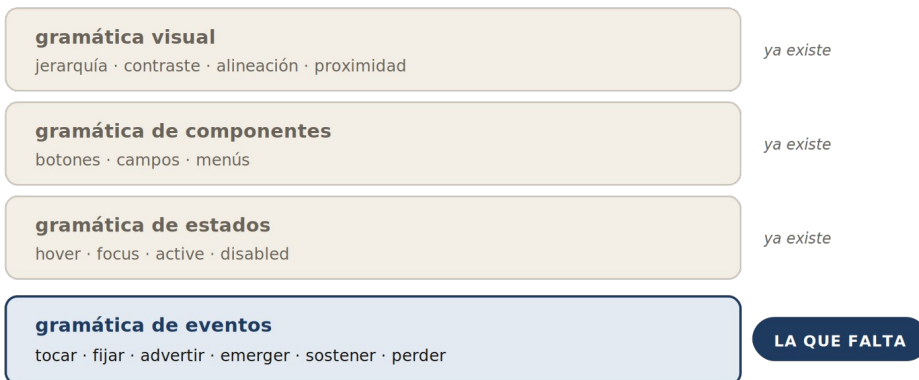
Una gramática de eventos es el sistema de diferencias, relaciones y reglas de composición mediante el cual una interfaz hace reconocibles las clases significativas de cambio que ocurren dentro de un contexto operativo.

Esta definición contiene piezas: diferencias (no todo cambio significa lo mismo), relaciones (los eventos se encadenan y componen), reglas de composición (contacto → sostenimiento → consolidación es una secuencia, no una acumulación de efectos), y reconocimiento (la gramática existe para el usuario, no para el sistema).

No llamamos gramática a esta capa porque queramos hacer una metáfora elegante. La llamamos gramática porque necesitamos algo más que una lista de eventos: necesitamos un sistema que defina diferencias, relaciones, combinaciones, modulaciones y límites de expresión. La palabra gramática aparece, entonces, porque el problema ya no es solo identificar eventos, sino articularlos. Una interfaz necesita distinguirlos, repetirlos con estabilidad, combinarlos en secuencias, modularlos cuando son evaluables, distribuirlos entre canales de expresión y reducirlos cuando la accesibilidad o el contexto lo exigen. Esa red de distinciones y reglas es lo que este libro llama gramática de eventos.

FIGURA 5.1

Las gramáticas de la interfaz: las que ya tiene y la que falta



La interfaz ya tiene gramáticas parciales; lo que aún no suele tener es una gramática explícita del evento.

§ 3

Responder no es suficiente

El usuario pulsa: algo cambia. Envía: aparece un spinner. Falla: el campo se marca en rojo. Desde el punto de vista técnico, hay respuesta. Pero desde el punto de vista perceptivo: ¿qué clase de respuesta es esa?

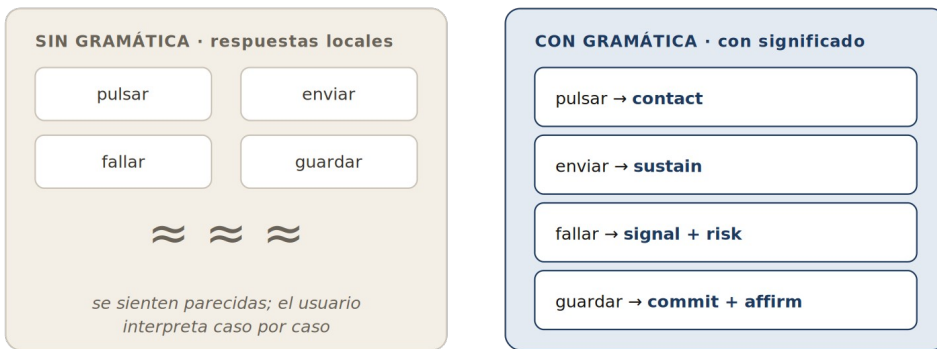
Si todo se resuelve con respuestas locales, la interfaz puede funcionar pero no construir lenguaje. El usuario aprende por contexto, pero ese aprendizaje procede de su esfuerzo por reconstruir lo que el sistema no diferencia.

PRINCIPIO

Una interfaz no necesita responder más. Necesita responder con diferencias que tengan significado.

FIGURA 5.2

Responder no es suficiente



No se trata de responder más, sino de responder con diferencias que tengan significado.

§ 4

El problema de los efectos aislados

Un efecto aislado responde a una pregunta local: ¿cómo hacemos que esto se note? Una gramática responde a otra: ¿qué clase de transformación necesita hacerse perceptible aquí? Si empezamos por el efecto, la interfaz se llena de soluciones agradables pero no coherentes. Si empezamos por el evento, el efecto se convierte en realización de una diferencia previa.

Distinción clave

Un efecto pregunta: ¿cómo se ve o se siente esto? Una gramática pregunta: ¿qué diferencia debe reconocer el usuario?

§ 5

Tres fallos de una interfaz sin gramática

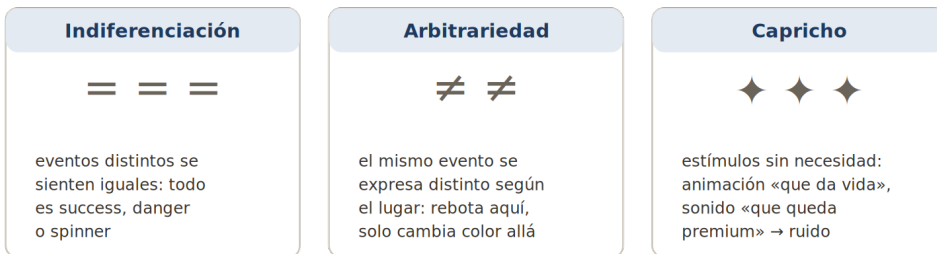
Indiferenciación: eventos distintos se sienten iguales. Todo resultado positivo es success, todo problema es danger, toda espera es spinner. La interfaz tiene vocabulario, pero es demasiado pequeño para las diferencias que necesita expresar.

Arbitrariedad: el mismo tipo de evento se expresa de formas distintas según el lugar. Una selección rebota en una pantalla, cambia solo de color en otra. La gramática permite variación sin perder identidad.

Capricho: estímulos sin necesidad semántica. Una animación porque «da vida», un sonido porque «queda premium». Cada estímulo que no aporta significado añade ruido. Una gramática no solo dice cómo expresar; también ayuda a decidir cuándo callar.

FIGURA 5.3

Tres fallos de una interfaz sin gramática



Una gramática combate los tres: diferencia, mantiene identidad y ayuda a decidir cuándo callar.

§ 6

Gramática y economía cognitiva

Cada vez que la interfaz cambia, el usuario responde a preguntas: ¿qué cambió? ¿lo causé yo? ¿sigue ocurriendo? ¿terminó? ¿es bueno o malo? ¿debo actuar? La gramática no elimina trabajo cognitivo. Lo distribuye mejor.

TABLA 5.2 Qué reduce una gramática de eventos

SIN GRAMÁTICA	CON GRAMÁTICA
El usuario interpreta cada cambio por contexto	El usuario reconoce familias de cambio
Los efectos son soluciones locales	Los efectos expresan eventos identificables
El color carga con demasiada semántica	La semántica se distribuye entre canales
La accesibilidad elimina señales sin reemplazo	La carga semántica puede migrar entre canales

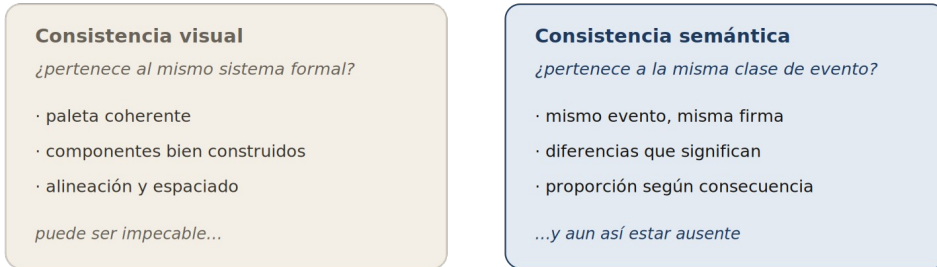
§ 7

De la consistencia visual a la consistencia semántica

La consistencia visual responde a: ¿esto pertenece al mismo sistema formal? La consistencia semántica responde a otra: ¿esto pertenece a la misma clase de evento? Un sistema puede ser visualmente consistente y semánticamente pobre: paleta impecable, componentes bien contruidos, pero eventos indiferenciados.

FIGURA 5.4

Consistencia visual frente a consistencia semántica



Un sistema puede ser visualmente impecable y semánticamente pobre: eventos indiferenciados bajo una forma pulida.

§ 8

Los niveles de la gramática

Cinco niveles que no deben mezclarse: la familia (qué clase de evento), el verbo (qué ocurre concretamente), la fase (en qué momento temporal), el intent (cómo se evalúa si es evaluable) y la realización (cómo se expresa por canales concretos).

Si mezclamos familia y efecto, acabamos diciendo «este evento es un bounce». Si mezclamos intent y color, «esto es rojo». Si mezclamos evento técnico y semántica, «esto es un click». Separar niveles permite pensar mejor.

FIGURA 5.5

Los cinco niveles de la gramática

NIVEL	PREGUNTA	EJEMPLO
familia	¿qué clase de evento?	commit
verbo	¿qué ocurre concretamente?	save
fase	¿en qué momento temporal?	asentar
intent	¿cómo se evalúa, si es evaluable?	affirm
realización	¿cómo se expresa por canales?	color · motion · sonido

Separar los niveles evita confundir el evento con su forma: ni «esto es un bounce» ni «esto es rojo».

§ 9

Composición y proporción

Los eventos reales rara vez aparecen aislados. Un guardado largo contiene contacto, sostenimiento y consolidación. Una eliminación contiene contacto, señal de amenaza, cambio de contexto y pérdida. Sin gramática, son acumulación de efectos. Con gramática, cada parte tiene función.

Además, no todas las diferencias merecen la misma intensidad. Si cada guardado automático se celebra como culminación, la recompensa se desgasta. Si cada validación fallida se trata como amenaza, la interfaz fatiga.

PRINCIPIO

La intensidad de una señal debe ser proporcional a la consecuencia perceptiva del evento.

TABLA 5.3 Proporción semántica

EVENTO	CONSECUENCIA	INTENSIDAD RECOMENDABLE
Contacto común	Baja, inmediata	Mínima y breve
Autoguardado	Positiva leve	Confirmación suave
Operación larga completada	Positiva con tensión previa	Resolución más expresiva
Eliminación irreversible	Alta consecuencia	Señal grave, no necesariamente ruidosa
Amenaza activa	Urgencia alta	Máxima claridad

§ 10

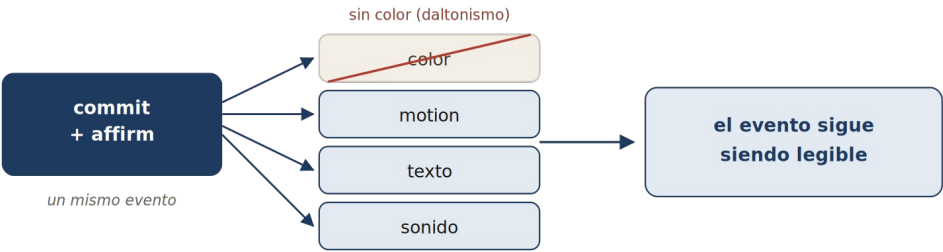
La gramática debe ser multicanal

Si todo depende del color, falla ante daltonismo. Si todo depende del motion, falla ante reducción de movimiento. La gramática debe distribuir significado de forma que el evento tenga identidad que sobreviva a la reducción de un canal.

La accesibilidad no solo elimina canales; obliga a redistribuir significado.

FIGURA 5.6

La gramática es multicanal



El significado se distribuye entre canales para que la identidad del evento sobreviva a la pérdida de uno.

§ 11

La gramática debe admitir silencio

No todo contacto necesita sonido. No toda selección necesita celebración. No toda espera necesita animación constante. Un efecto agradable una vez puede volverse insoportable al repetirse cien veces al día.

PRINCIPIO

Una gramática perceptiva no consiste en expresar siempre más, sino en saber cuándo una señal debe permanecer mínima o incluso silenciosa.

§ 12

Gramática no significa rigidez

Una gramática no elimina estilo. Lo hace más responsable. Un sistema financiero, una herramienta creativa y un editor de código no deberían sonar igual. Pero si pertenecen a la misma familia de evento, deberían conservar relación reconocible con esa familia. La gramática no fija una única forma. Fija una relación entre significado y expresión.

§ 13

Gramática y aprendizaje implícito

El usuario no debe aprender la gramática explícitamente. Funciona por familiaridad, por recurrencia, por diferencia sensible. El aprendizaje implícito requiere suficiente regularidad para reconocer y suficiente elasticidad para adaptarse.

TABLA 5.4 Condiciones del aprendizaje implícito

CONDICIÓN	QUÉ APORTA
Recurrencia	El patrón aparece suficientes veces
Diferenciación	No se confunde con otros eventos
Estabilidad	Conserva núcleo semántico
Proporción	La intensidad coincide con la importancia

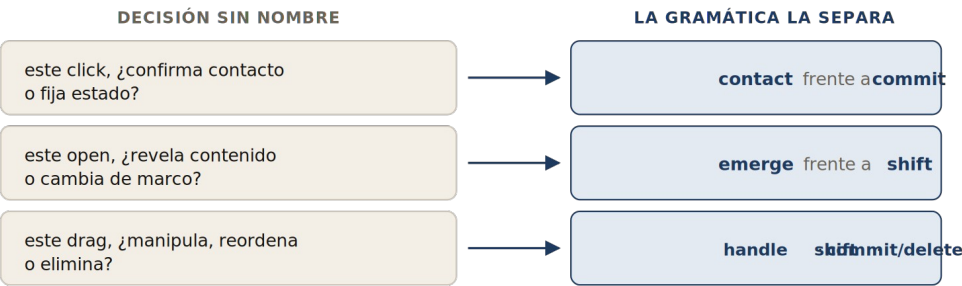
Multicanalidad	Puede sobrevivir a cambios de canal
Sobriedad	No fatiga por repetición

§ 14

La gramática como herramienta para programadores

Programar interfaces implica tomar decisiones sin nombre claro. ¿Este click confirma contacto o fija estado? ¿Este open revela contenido o cambia de marco? ¿Este drag manipula, reordena o elimina? Sin gramática, se resuelve con lógica local. Con gramática, se separan los niveles.

FIGURA 5.7
La gramática como herramienta para programar



Sin gramática se resuelve con lógica local; con gramática, cada decisión se separa en niveles con nombre.

§ 15

Lo que una gramática debe responder

TABLA 5.5 Preguntas mínimas de una gramática de eventos

PREGUNTA	RESPUESTA QUE BUSCA
----------	---------------------

¿Qué clase de evento es?	Familia semántica
¿Qué ocurrió concretamente?	Verbo o variante
¿En qué momento está?	Fase temporal
¿Es evaluable?	Posibilidad de intent
¿Qué canales participan?	Realización multicanal
¿Qué intensidad merece?	Proporción
¿Qué pasa si un canal falta?	Migración semántica
¿Con qué otros eventos se compone?	Secuencia o composición

§ 16

Resumen operativo

No estamos diciendo que todas las interfaces necesiten una gramática explícita con nombres técnicos. No estamos diciendo que cada evento deba disparar motion, sonido o háptica. No estamos diciendo que la gramática deba ser idéntica en todos los productos.

Si una interfaz produce transformaciones significativas de forma recurrente, necesita una manera consistente de hacerlas perceptibles.

§ 17

Síntesis

Una interfaz madura no solo necesita componentes consistentes. Necesita eventos consistentes.

Tabla de síntesis del capítulo

El siguiente paso será definir con más precisión la unidad que articula esa consistencia: la interacción semántica como configuración perceptiva recurrente que hace legible una clase de evento dentro de un contexto operativo.

TABLA 5.6 Distinciones que fija este capítulo

DISTINCIÓN	POR QUÉ IMPORTA
Efecto y evento	El efecto es medio; el evento es significado
Estado y transformación	El usuario necesita saber qué cambió
Marco y mensaje	No todo evento acepta intent
Canal y significado	Ningún canal agota la semántica
Intensidad y consecuencia	No todo merece la misma energía
Silencio y ausencia	No expresar también puede ser decisión semántica

06

Qué es una interacción semántica

La forma perceptiva recurrente de un evento

PREGUNTA DEL CAPÍTULO

¿Qué convierte una respuesta concreta de la interfaz en una unidad de lenguaje reutilizable?

Los capítulos anteriores han preparado el terreno. Primero redefinimos la interfaz como dimensión perceptible de un contexto operativo. Después introducimos el evento como unidad básica de cambio significativo. Explicamos por qué las convenciones heredadas no han formulado antes estas preguntas. Y desarrollamos por qué esos eventos necesitan una gramática.

Ahora aparece la pregunta central de este capítulo:

Una interacción semántica es una configuración perceptiva recurrente mediante la cual una interfaz hace legible una clase de evento dentro de un contexto operativo.

§ 1

Por qué necesitamos otro término

Un componente nombra una unidad estructural. Un estado nombra una condición estable. Un evento técnico nombra una ocurrencia del sistema. Un efecto nombra una realización perceptiva. Un patrón nombra una solución recurrente. Un intent nombra el tono evaluativo. Pero entre todo eso falta una capa: la unidad que puede decir esto es una clase de evento que la interfaz necesita hacer reconocible.

La interacción semántica nombra la capa donde componentes, estados, efectos y colores se organizan alrededor de un significado operativo.

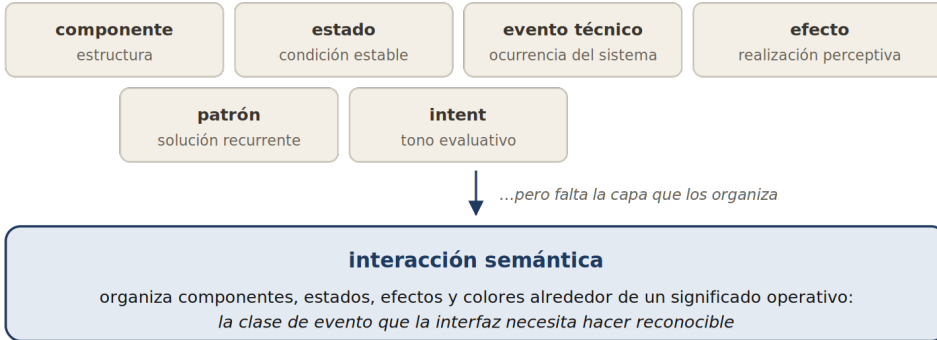
TABLA 6.1 Por qué no basta con los términos existentes

TÉRMINO	QUÉ NOMBRA	POR QUÉ NO BASTA
Componente	Una unidad estructural	No dice qué evento ocurre
Estado	Una condición estable	No explica la transformación
Evento técnico	Una ocurrencia del sistema	No contiene significado experiencial
Efecto	Una realización perceptiva	Puede existir sin semántica
Patrón	Una solución recurrente	Puede contener varias semánticas
Intent	Tono evaluativo	No define la clase de evento
Interacción semántica	Forma recurrente de hacer legible un evento	Articula significado y percepción

FIGURA 6.1

La capa que falta

CADA TÉRMINO NOMBRA ALGO...



Entre componente, estado, efecto, patrón e intent faltaba un nombre para la clase de evento que se hace legible.

§ 2

Definición de interacción semántica

La definición contiene cinco partes: configuración perceptiva (puede incluir movimiento, color, sonido, presencia, háptica), recurrente (debe poder reaparecer en distintos componentes y situaciones), hace legible (su función es reducir ambigüedad), una clase de evento (contacto, consolidación, señal, manipulación, aparición, cambio de contexto, sostenimiento) y dentro de un contexto operativo (el significado no existe en abstracto).

Definición

Interacción semántica: configuración perceptiva recurrente que hace reconocible una clase de evento dentro de un contexto operativo.

§ 3

No es una interacción concreta

El mismo gesto puede pertenecer a semánticas distintas. Un click puede ser contacto, confirmación, navegación, advertencia, pérdida, apertura, cierre o selección. La interacción concreta es la superficie gestual. La interacción semántica es la lectura significativa de lo que ocurre.

TABLA 6.2 Un mismo gesto, múltiples semánticas posibles

INTERACCIÓN CONCRETA	PUEDEN SIGNIFICAR	SEMÁNTICA POSIBLE
Pulsar botón	El sistema registró mi gesto	Contacto
Pulsar botón	La acción quedó confirmada	Consolidación
Pulsar botón	Se abre una superficie	Aparición
Pulsar botón	Entro en otro flujo	Cambio de contexto
Pulsar botón	Pierdo un recurso	Pérdida / commit
Arrastrar elemento	Lo estoy moviendo	Manipulación

§ 4

No es una animación

Un bounce puede expresar selección, culminación, encaje o celebración. Un fade puede expresar aparición, retirada, pérdida o simple suavizado. La animación es ambigua hasta que se integra en una gramática. El orden correcto no es: tenemos un bounce → ¿qué significa? Sino: necesitamos expresar consolidación → ¿qué configuración perceptiva la hace legible?

Distinción clave

La animación pertenece al plano de la realización. La interacción semántica pertenece al plano del significado operativo.

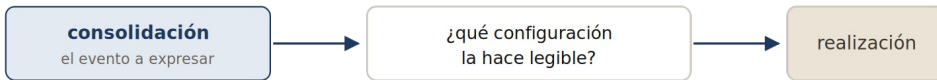
FIGURA 6.2

No es una animación

EL ORDEN EQUIVOCADO · del efecto al significado



EL ORDEN CORRECTO · del significado al efecto



La animación es ambigua hasta integrarse en una gramática: el significado precede a la forma.

§ 5

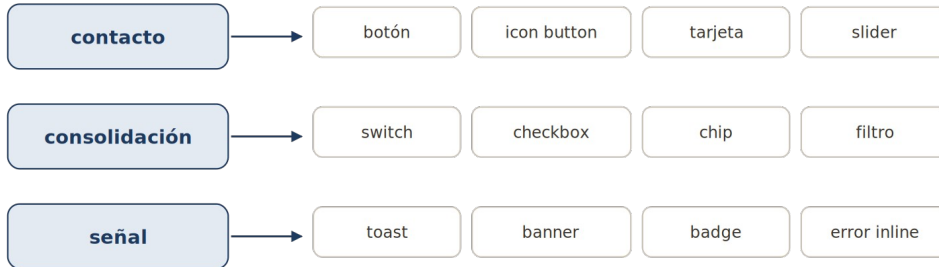
No es un componente

La semántica de contacto puede aparecer en un botón, un icon button, una tarjeta clicable, un slider thumb o un control de reproducción. La consolidación puede aparecer en un switch, un checkbox, un chip, un filtro. La señal puede aparecer en un toast, un banner, un badge, un error inline. Si una categoría depende de un componente concreto, todavía no es una interacción semántica en sentido fuerte.

FIGURA 6.3

No es un componente

una misma semántica reaparece en componentes muy distintos



Si una categoría depende de un componente concreto, todavía no es una interacción semántica en sentido fuerte.

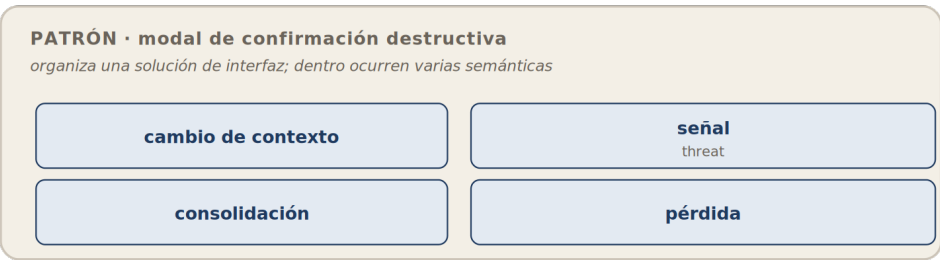
§ 6

No es un patrón de diseño

Un modal de confirmación destructiva puede contener cambio de contexto, señal de amenaza, consolidación y pérdida. Un toast puede contener señal con intent de riesgo, confirmación o pérdida. El patrón organiza una solución de interfaz. La semántica organiza el significado perceptivo de los eventos que ocurren dentro.

FIGURA 6.4

No es un patrón de diseño



El patrón organiza la solución; la semántica organiza el significado de los eventos que ocurren dentro.

§ 7

No es un intent

La semántica responde a ¿qué clase de evento ocurre? El intent responde a ¿cómo debe evaluarse afectivamente? No es lo mismo commit + fulfill, signal + fulfill y contact + affirm. En todos hay valencia positiva, pero no el mismo evento. El intent colorea o modula la semántica. No la sustituye.

TABLA 6.3 Semántica e intent no son lo mismo

PREGUNTA	PERTENECE A	EJEMPLO
¿Qué clase de evento ocurre?	Semántica	Contacto, señal, consolidación, aparición
¿Qué ocurrió concretamente?	Verbo	Guardar, eliminar, abrir, seleccionar
¿En qué fase está?	Fase	Enter, exit, start, progress, drop
¿Cómo se evalúa?	Intent	Neutral, affirm, fulfill, risk, threat, loss
¿Cómo se expresa?	Canales	Motion, sonido, color, presencia, háptica

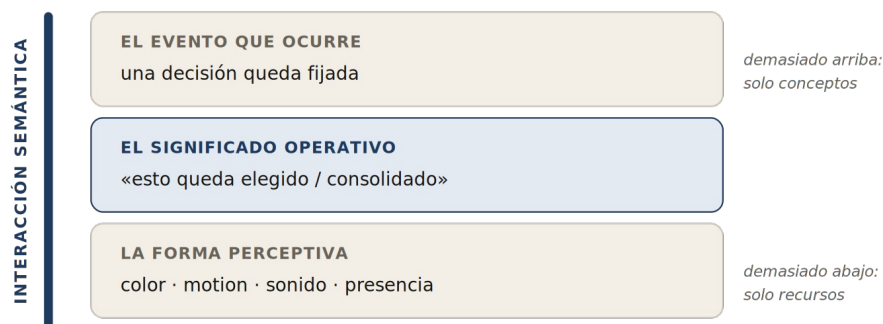
§ 8

La interacción semántica como unidad de articulación

Articula tres planos: el evento que ocurre, el significado operativo que ese evento tiene, y la forma perceptiva que permite reconocerlo. Si nos quedamos demasiado arriba, solo tenemos conceptos abstractos. Si bajamos demasiado pronto, solo tenemos recursos. La interacción semántica conecta ambas capas.

FIGURA 6.5

La interacción semántica articula tres planos



Conecta el evento con su significado y con la forma que lo hace reconocible, sin quedarse en ningún extremo.

§ 9

Toda interacción semántica responde a una pregunta perceptiva

Una semántica merece existir cuando responde a una pregunta que aparece una y otra vez en la interacción. No porque sea bonita ni porque encaje en un framework, sino porque el usuario necesita resolver esa diferencia para actuar con claridad.

TABLA 6.4 Preguntas perceptivas y familias semánticas

PREGUNTA PERCEPTIVA	FAMILIA SEMÁNTICA
¿El sistema ha sentido mi acción?	Contacto / contact
¿Esto ha quedado fijado?	Consolidación / commit
¿Algo reclama mi atención?	Señal / signal
¿Estoy controlando este objeto?	Manipulación / handle
¿Algo ha aparecido o desaparecido?	Aparición / emerge
¿Ha cambiado el contexto?	Cambio de contexto / shift
¿Esto sigue ocurriendo?	Sostenimiento / sustain

§ 10

Tiene una firma, no una receta

Una receta fija valores: duración 180ms, scale 0.97, frecuencia 2600 Hz. Una firma semántica dice algo más abstracto: breve, inmediata, proporcional, causalmente ligada a la acción, con baja persistencia. La receta puede cambiar. La firma debe permanecer.

Distinción clave

Una receta fija valores. Una firma conserva identidad bajo variación.

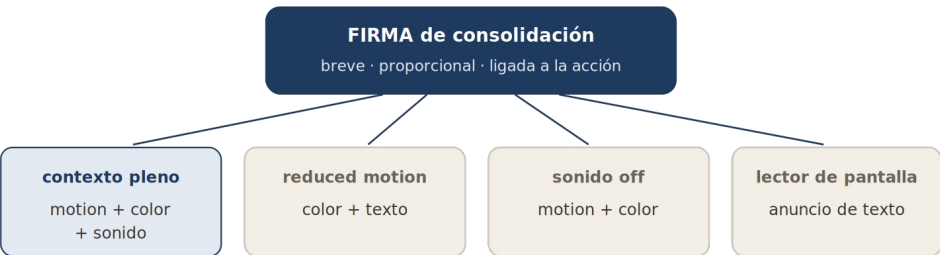
§ 11

Puede realizarse por varios canales

Si el usuario reduce motion, la semántica no debería desaparecer. Si el sonido está desactivado, la información no debería perderse. La interacción semántica es lo que permite esa migración: sabemos qué significado debía transportarse, así que podemos redistribuirlo.

FIGURA 6.6

Una firma, no una receta



La firma se conserva bajo variación; por eso el mismo significado puede migrar entre canales.

§ 12

Tiene estructura temporal

TABLA 6.5 Formas temporales de las semánticas

SEMÁNTICA	FORMA TEMPORAL	LECTURA
Contacto	Puntual	«me ha sentido»
Consolidación	Breve + resolutiva	«esto quedó fijado»
Señal	Entrada + posible persistencia	«atiende esto»
Aparición	Entrada / salida	«algo entra o sale del campo»
Cambio de contexto	Cruce de umbral	«ahora estás en otro régimen»
Manipulación	Pick / carry / drop	«lo agarro, lo muevo, lo coloco»
Sostenimiento	Inicio / continuidad / fin	«sigue ocurriendo»

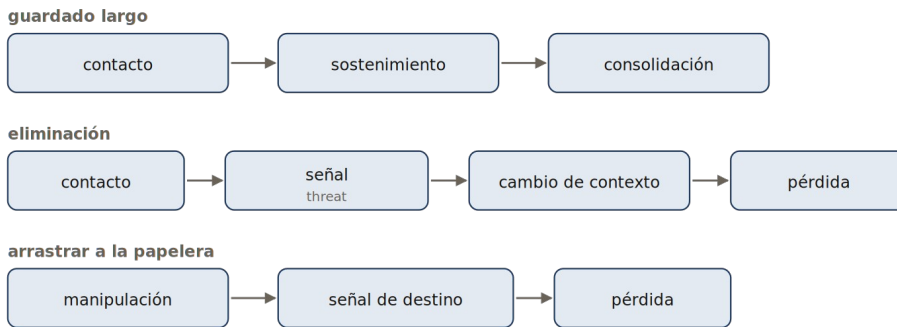
§ 13

Puede componerse con otras semánticas

Un guardado largo: contacto → sostenimiento → consolidación. Una eliminación: contacto → señal de amenaza → cambio de contexto → pérdida. Un drag a papelera: manipulación → señal de destino → pérdida. La gramática permite asignar cada función a su lugar.

FIGURA 6.7

Composición en secuencia



Las semánticas se encadenan en secuencias; la gramática asigna a cada parte su función.

§ 14

Tiene fronteras

TABLA 6.6 Fronteras semánticas importantes

NO CONFUNDIR	DIFERENCIA
Contacto / consolidación	Registrar gesto no es fijar estado
Aparición / señal	Aparecer no es reclamar atención
Aparición / cambio de contexto	Entrar en campo no es cambiar régimen
Sostenimiento / culminación	Seguir ocurriendo no es haber terminado
Ocultación / pérdida	Retirar de la vista no es destruir
Intent / semántica	El tono evaluativo no define el tipo de evento

§ 15

Requiere estabilidad y elasticidad

La estabilidad está en el núcleo: qué pregunta responde, qué tipo de evento hace legible, qué forma temporal tiene. La elasticidad está en la realización: duración, intensidad, estilo, tema, plataforma, accesibilidad.

PRINCIPIO

La identidad de una interacción semántica no está en repetir siempre la misma forma, sino en conservar la misma función perceptiva bajo variaciones controladas.

§ 16

Puede ser mínima

A veces será casi invisible. A veces una latencia ajustada. A veces una microcompresión. A veces un silencio deliberado. Cuanto más frecuente es un evento, más sobria debe ser su expresión.

PRINCIPIO

Semántico no significa intenso. Significa proporcional, reconocible y necesario.

§ 17

Cómo saber si algo merece ser interacción semántica

TABLA 6.7 Condiciones mínimas de una interacción semántica

CONDICIÓN	PREGUNTA
Recurrencia	¿Aparece en muchos contextos?
Relevancia operativa	¿Afecta a acción, atención o estado?
Diferenciabilidad	¿Se distingue de otras semánticas?
Estabilidad	¿Puede aprenderse?
Multicanalidad	¿Puede expresarse por más de un canal?

Economía cognitiva	¿Ayuda a comprender mejor?
Proporción	¿Su intensidad corresponde a su consecuencia?

Una interacción semántica no se declara; se justifica.

§ 18

Ejemplo: selección

El usuario marca una opción. Desde la implementación: `click`, `state.selected = true`, `class="selected"`. Desde la experiencia: una decisión del usuario queda fijada. La selección no es mero contacto. Comunica «esto queda elegido». Si usamos la misma respuesta que para un botón cualquiera, el usuario siente que tocó pero no necesariamente que eligió.

§ 19

Ejemplo: notificación

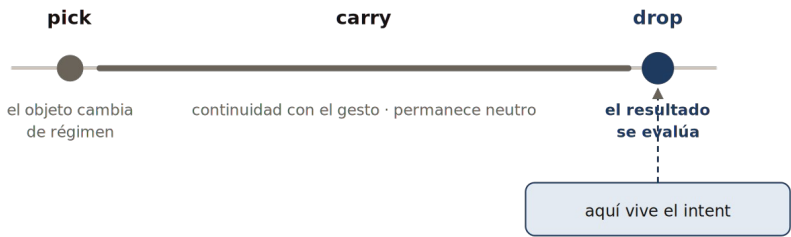
Un toast aparece, un badge cambia, un banner se muestra. La semántica no es «toast» — toast es un patrón. La semántica es señal: el sistema inicia comunicación y reclama cierto grado de atención. El intent modula: neutra si informa, riesgo si advierte, amenaza si exige reacción, pérdida si registra algo consumado.

§ 20

Ejemplo: manipulación

Arrastrar un elemento no es solo pulsar y mover. Es una experiencia continua con fases: `pick`, `carry`, `drop`. En el `pick`, el objeto cambia de régimen. En el `carry`, conserva continuidad con el gesto. En el `drop`, el resultado se evalúa. La evaluación se concentra en el `drop`. El `carry` debe permanecer neutro.

FIGURA 6.8
Las fases de la manipulación



La evaluación se concentra en el drop; el carry debe permanecer neutro para no anticipar el resultado.

§ 21

Lo que una interacción semántica debe especificar

TABLA 6.8 Ficha mínima de una interacción semántica

CAMPO	PREGUNTA
Qué comunica	¿Qué tipo de evento vuelve legible?
Pregunta perceptiva	¿Qué necesita resolver el usuario?
Fronteras	¿Con qué no debe confundirse?
Temporalidad	¿Cómo ocupa el tiempo?
Origen	¿Responde al usuario o parte del sistema?
Intent	¿Es evaluable o estructural?
Canales	¿Por dónde se expresa mejor?
Accesibilidad	¿Cómo migra si un canal falta?
Proporción	¿Cuánta energía merece?
Errores	¿Cómo suele fallar?

§ 22

Resumen operativo

Cuando una clase de evento reaparece una y otra vez, y su presencia cambia lo que el usuario puede hacer, atender o entender, la interfaz necesita una forma consistente de hacerla reconocible. A esa forma consistente la llamamos interacción semántica.

§ 23

Síntesis

TABLA 6.9 Qué responde cada capa

CONCEPTO	QUÉ RESPONDE
Componente	¿Qué elemento participa?
Evento técnico	¿Qué ocurrió en el sistema?
Evento	¿Qué transformación percibe el usuario?
Interacción semántica	¿Cómo se hace reconocible esa clase de transformación?
Intent	¿Cómo se evalúa afectivamente, si procede?
Realización	¿Por qué canales se expresa?

Una interfaz no solo necesita componentes reutilizables. Necesita formas recurrentes de hacer legibles sus eventos.

El siguiente capítulo desarrollará los criterios para decidir cuándo una diferencia merece convertirse en interacción semántica. Porque no todo cambio necesita nombre propio. Pero los cambios que organizan la experiencia sí.

07

Criterios para identificar una interacción semántica

Cuándo una diferencia merece convertirse en lenguaje

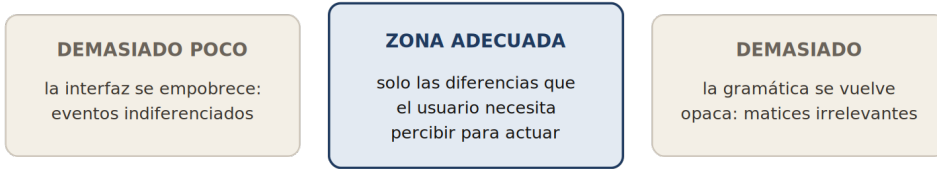
PREGUNTA DEL CAPÍTULO

¿Cómo se decide si una diferencia perceptiva merece convertirse en lenguaje del sistema?

Si una interacción semántica es una forma recurrente de hacer legible una clase de evento, necesitamos decidir qué diferencias merecen convertirse en semánticas y cuáles no. Si distinguimos demasiado poco, la interfaz se empobrece. Si distinguimos demasiado, la gramática se vuelve opaca.

FIGURA 7.1

Ni demasiado poco, ni demasiado



Una semántica debe declararse solo cuando una diferencia merece de verdad convertirse en lenguaje.

§ 1

El riesgo de declarar semánticas por intuición

Una semántica no se justifica porque podamos nombrarla. Se justifica porque nombra una diferencia que el usuario necesita reconocer de forma recurrente para actuar con claridad.

PRINCIPIO

Una interacción semántica no existe porque tenga nombre. Existe cuando nombra una diferencia que el usuario necesita percibir.

§ 2

Criterio 1: relevancia operativa

Una diferencia merece convertirse en semántica si afecta a la acción, a la atención o a la interpretación del estado. Una semántica no debería nacer de una variación estética, sino de una diferencia operativa.

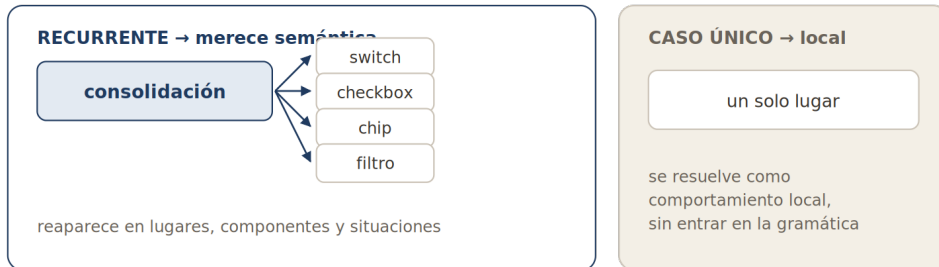
§ 3

Criterio 2: recurrencia estructural

Una interacción semántica debe corresponder a una clase de evento que reaparece en distintos lugares, componentes o situaciones. Si solo ocurre en un caso específico, puede resolverse como comportamiento local.

FIGURA 7.2

Recurrencia estructural



Si una clase de evento solo aparece en un caso, no necesita convertirse en lenguaje del sistema.

§ 4

Criterio 3: diferenciabilidad perceptiva

No basta con que internamente sepamos que dos eventos son distintos. El usuario debe poder percibir esa diferencia. Contacto no debería sentirse igual que consolidación. Aparición no debería sentirse igual que señal. Cambio de contexto no debería sentirse igual que expansión.

PRINCIPIO

Una diferencia que no puede percibirse todavía no funciona como diferencia semántica.

FIGURA 7.3

Diferenciabilidad perceptiva

no basta con que sean distintos por dentro: el usuario debe percibir la diferencia



Una diferencia que no puede percibirse todavía no funciona como diferencia semántica.

§ 5

Criterio 4: estabilidad de significado

Una semántica debe conservar su significado a través de distintas realizaciones. Si solo existe cuando usamos una animación concreta, no es todavía una semántica fuerte. Es un efecto. La semántica de contacto puede expresarse con microcompresión visual en web, pulso háptico en móvil, sonido breve sin motion o cambio de foco visible. La realización cambia; el significado permanece.

§ 6

Criterio 5: elasticidad de realización

Una interfaz bancaria, una aplicación médica, una herramienta creativa y un editor de código no deberían expresar igual sus eventos. Pero cuando dos productos expresan la misma familia, ambos deberían mantener un aire reconocible con ella: la identidad puede variar el acento, no el idioma.

FIGURA 7.4

Elasticidad de realización



Distintos productos expresan el evento de forma distinta, pero conservan relación reconocible con su familia.

§ 7

Criterio 6: economía cognitiva

Una interacción semántica debe reducir carga interpretativa. Si una categoría nueva obliga a pensar más, recordar más o diferenciar matices irrelevantes, probablemente no merece existir. La pregunta: ¿esta semántica ayuda al usuario a interpretar mejor el evento o solo satisface nuestra necesidad interna de clasificar?

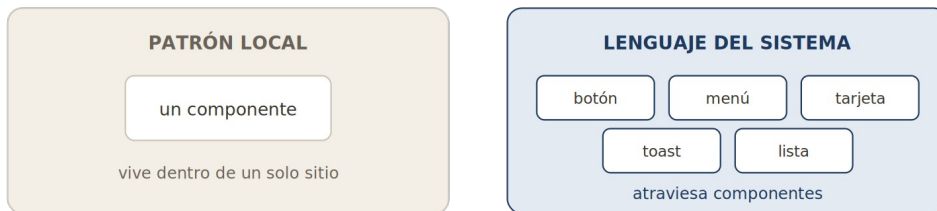
§ 8

Criterio 7: transversalidad

Si una supuesta semántica solo existe dentro de un componente concreto, quizá no estamos ante una semántica sino ante un patrón local. La transversalidad es lo que convierte una respuesta en lenguaje del sistema.

FIGURA 7.5

Transversalidad



La transversalidad es lo que convierte una respuesta en lenguaje del sistema, no en truco de un componente.

§ 9

Criterio 8: compatibilidad multicanal

Una semántica no debería depender exclusivamente de un único canal. Si se reduce motion, parte del significado puede migrar a color, forma, texto o sonido. Si el color no es fiable, la forma, el texto, la posición o el icono deben sostener la diferencia.

PRINCIPIO

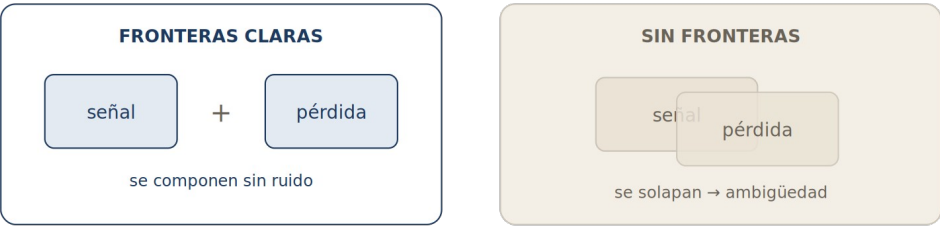
Una semántica no debería vivir en un canal. Debería poder viajar entre canales sin perder su función.

§ 10

Criterio 9: capacidad de composición

Las interacciones semánticas no aparecen siempre aisladas. Una categoría merece entrar en la gramática si puede componerse con otras sin generar ambigüedad. Esto exige fronteras claras.

FIGURA 7.6
Capacidad de composición



Una categoría entra en la gramática si puede componerse con otras sin generar ambigüedad.

§ 11

Criterio 10: proporción expresiva

Si todo suena, nada importa. Si todo vibra, la háptica se vuelve ruido. Si todo rebota, la confirmación pierde valor. La proporción expresiva pregunta: ¿cuánta intensidad merece este evento?

TABLA 7.1 Proporción expresiva

EVENTO	CONSECUENCIA	FRECUENCIA	INTENSIDAD
Contacto cotidiano	Baja	Alta	Mínima
Autoguardado	Baja positiva	Alta	Confirmación suave
Operación completada	Media-alta	Baja	Resolución perceptible
Amenaza activa	Alta	Baja	Interrupción justificada
Pérdida consumada	Alta, ya ocurrida	Baja	Gravedad sin alarma

§ 12

Criterio 11: asimetría significativa

La direccionalidad importa: abrir y cerrar no son el mismo evento con el signo cambiado, ni seleccionar equivale a deshacer una selección, ni eliminar es el espejo de restaurar. La asimetría es especialmente clara en semánticas como handle, donde pick, carry y drop no son fases equivalentes.

PRINCIPIO

Una salida no es siempre una entrada invertida. Puede tener otro significado, otra energía y otra consecuencia.

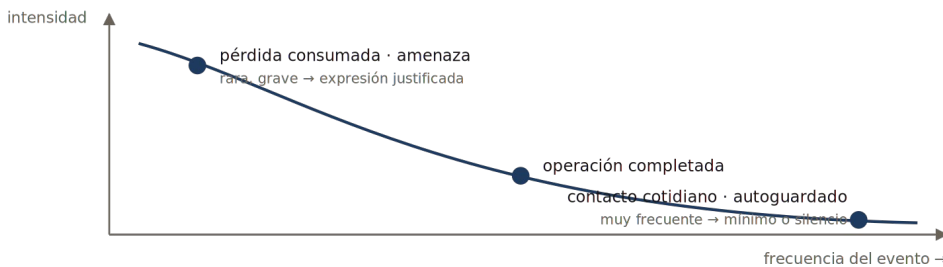
§ 13

Criterio 12: sobriedad y silencio

No todo contacto necesita sonido. No toda confirmación necesita celebración. No toda espera necesita animación constante. Un sonido agradable una vez puede volverse insoportable cincuenta veces al día.

FIGURA 7.7

Sobriedad: a más frecuencia, menos intensidad



Un sonido agradable una vez se vuelve insoportable cincuenta veces al día.

§ 14

Criterio 13: posibilidad de validación

Si una semántica está bien definida, deberíamos poder comprobar si ayuda. Si retrasamos el contacto, debería disminuir la sensación de

agencia. Si usamos la misma firma para contacto y consolidación, deberían aumentar las dudas sobre si el estado quedó fijado. La validación convierte la semántica en hipótesis de diseño, no en opinión estética.

§ 15

Criterio 14: compatibilidad con accesibilidad

Si la semántica desaparece al quitar un canal, era demasiado frágil. Un error no puede depender solo del rojo. Una alerta no puede depender solo del sonido. Una manipulación no puede depender solo de sombra si el contraste es pobre.

PRINCIPIO

La accesibilidad no solo adapta la interfaz. Revela si la semántica estaba bien distribuida.

§ 16

Matriz de decisión

TABLA 7.2 Matriz para aceptar una interacción semántica

CRITERIO	PREGUNTA	SI LA RESPUESTA ES NO...
Relevancia	¿Afecta a acción, atención o estado?	Es variación local
Reurrencia	¿Aparece en distintos contextos?	Es caso específico
Diferenciabilidad	¿Se distingue perceptivamente?	Se fusiona con otra
Estabilidad	¿Mantiene significado bajo variación?	Es receta o efecto
Elasticidad	¿Puede adaptarse a plataformas?	Es demasiado rígida
Economía	¿Reduce ambigüedad?	Añade ruido taxonómico
Transversalidad	¿Atraviesa componentes?	Es patrón local

Multicanalidad	¿Puede migrar entre canales?	Es frágil
Composición	¿Se combina sin confusión?	Necesita mejores fronteras
Proporción	¿Intensidad corresponde al evento?	Puede fatigar o trivializar
Sobriedad	¿Sabe cuándo callar?	Puede sobreestimar
Validación	¿Produce predicciones observables?	Es opinión difícil de evaluar
Accesibilidad	¿Sobrevive a reducción de canal?	No está completa

§ 17

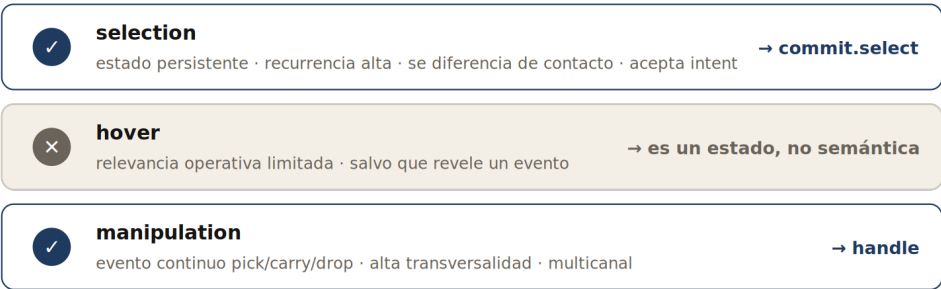
Tres ejemplos

Selection: sí merece semántica, como `commit.select`. Fija estado persistente, tiene recurrencia alta, se diferencia de contacto, acepta intent.

Hover: no como semántica propia. Estado de disponibilidad con relevancia operativa limitada, salvo que revele evento.

Manipulation: sí, como `handle`. Evento continuo irreducible con estructura pick/carry/drop, alta transversalidad y multicanalidad.

FIGURA 7.8
Tres ejemplos a través de los criterios



Una interacción semántica no se acepta porque sea posible, sino porque es necesaria.

§ 18

Resumen operativo

Una diferencia debe convertirse en interacción semántica solo cuando ayuda de forma recurrente, perceptible y proporcional a comprender un evento operativo.

§ 19

Síntesis

TABLA 7.3 Síntesis de criterios

CRITERIO	FUNCIÓN
Relevancia operativa	Asegura que la diferencia importa
Reurrencia	Evita categorías demasiado locales
Diferenciabilidad	Garantiza que pueda percibirse
Estabilidad	Permite aprendizaje
Elasticidad	Permite adaptación
Economía cognitiva	Evita ruido taxonómico
Transversalidad	Separa semántica de componente
Multicanalidad	Evita fragilidad de canal único
Composición	Permite secuencias complejas
Proporción	Calibra intensidad
Sobriedad	Permite callar
Validación	Convierte la semántica en hipótesis comprobable
Accesibilidad	Asegura distribución robusta del significado

Una interacción semántica no se acepta porque sea posible. Se acepta porque es necesaria.

El siguiente capítulo dará el paso que estos criterios preparan: ¿qué familias de evento sobreviven a estos criterios y merecen formar parte de la gramática fundamental de la interfaz?

08

De las clases de evento a las familias semánticas

Cómo reducir sin empobrecer

PREGUNTA DEL CAPÍTULO

¿Cómo se reduce un campo abierto de eventos a unas pocas familias sin perder las diferencias que importan?

Los capítulos anteriores han construido ya las piezas necesarias. Este capítulo aplica los criterios del capítulo anterior. La pregunta ya no es qué es una interacción semántica, sino: ¿qué familias de evento sobreviven cuando aplicamos esos criterios?

La respuesta que propone este libro es una reducción desde un campo exploratorio amplio hacia ocho familias fundamentales: contact, commit, signal, handle, emerge, shift, sustain, delegate. Esta reducción no busca empobrecer el sistema. Busca conservar las diferencias esenciales sin convertir cada matiz en categoría raíz.

Estas familias no son categorías de diseño. Son categorías perceptivas. Cada una se apoya en una forma distinta de organizar la experiencia:

contact se apoya en agencia y causalidad inmediata; commit, en cierre cognitivo y memoria de estado; signal, en atención y saliencia; handle, en acción corporal y manipulación; emerge, en figura/fondo y presencia; shift, en cambio de contexto de trabajo; sustain, en continuidad temporal y reducción de incertidumbre; delegate, en atribución de agencia y autoría compartida.

PRINCIPIO

Las familias no salen de componentes. Salen de preguntas perceptivas recurrentes.

FIGURA 8.1

Las ocho familias y su base perceptiva



Las familias no salen de componentes, sino de preguntas perceptivas recurrentes.

§ 1

De una lista exploratoria a una gramática manejable

En los primeros borradores aparecía una lista amplia de trece semánticas. Esa lista cumplió una función necesaria: abrir el campo y mostrar que la interfaz contiene tipos de cambio muy distintos. Pero una lista exploratoria no es todavía una gramática madura. La reducción debe hacerse

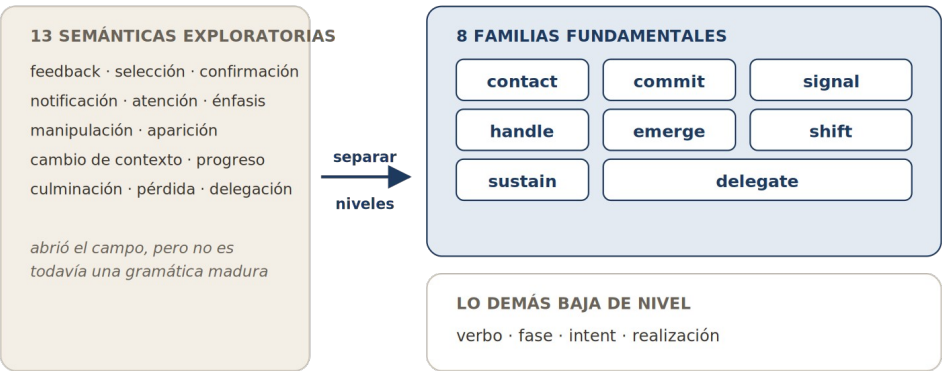
separando niveles: no todo debe ser familia. Algunas diferencias pertenecen al verbo, a la fase, al intent o a la realización.

TABLA 8.1 Niveles de la gramática

NIVEL	PREGUNTA	EJEMPLO
Familia	¿Qué clase perceptiva de evento es?	contact, commit, signal
Verbo	¿Qué ocurrió concretamente?	select, delete, notify, expand
Fase	¿En qué momento temporal está?	enter, exit, pick, carry, drop
Intent	¿Cómo se evalúa, si procede?	neutral, affirm, fulfill, risk, threat, loss
Realización	¿Por qué canales se expresa?	motion, sonido, color, presencia, háptica

FIGURA 8.2

De una lista exploratoria a una gramática manejable



Reducir no es empobrecer: es separar lo que es familia de lo que es verbo, fase, intent o realización.

§ 2

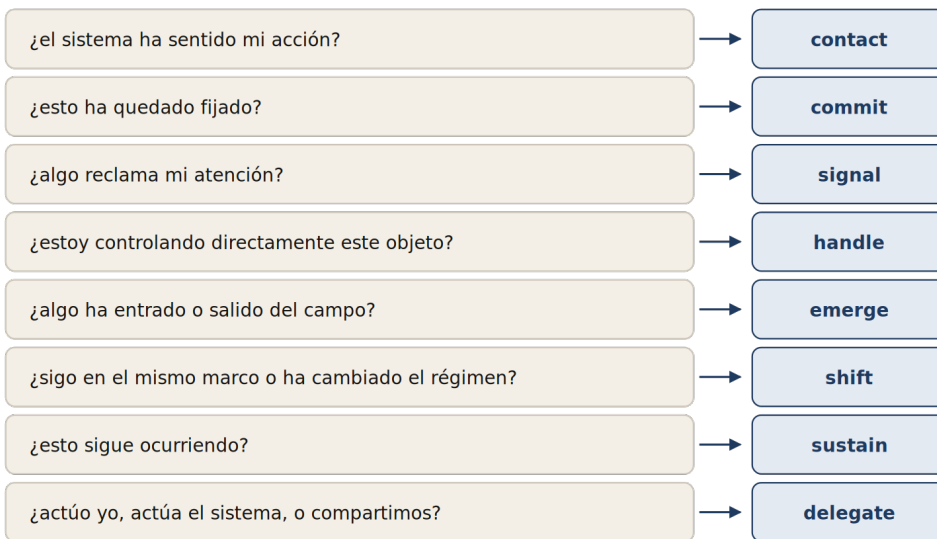
Las ocho preguntas fundamentales

Aplicando los criterios aparece una estructura compacta de ocho preguntas perceptivas. No son preguntas que el usuario formule verbalmente. Son preguntas operativas que la interfaz debe responder.

Las familias semánticas no forman una jerarquía moral. Distinguen funciones perceptivas.

FIGURA 8.3

Las ocho preguntas perceptivas



§ 3

Las ocho familias fundamentales

contact

Comunica que una acción directa del usuario ha sido registrada. ¿El sistema ha sentido mi acción? No comunica todavía resultado ni estado persistente: comunica que la acción y su respuesta forman un solo gesto. Aquí se reubica la antigua semántica feedback. Ejemplos: `contact.press`, `contact.release`, `contact.activate`.

commit

Comunica que algo ha quedado fijado, resuelto o establecido. ¿Esto ha quedado fijado? Aquí entran selección, confirmación, guardado, cancelación, fallo, eliminación, restauración, culminación o pérdida consumada. Es una de las familias más claramente evaluables.

signal

Comunica que el sistema inicia una señal para orientar la atención. ¿Algo reclama mi atención? No todas las señales son alertas: también hay anuncios, recordatorios, hints, badges y énfasis. Acepta intent plenamente.

handle

Comunica manipulación directa de un objeto. ¿Estoy controlando directamente este objeto? Tiene estructura temporal propia: pick → carry → drop. El carry es estructural y neutro; la evaluación aparece sobre todo en el drop.

emerge

Comunica aparición, retirada o disponibilidad en el campo perceptivo. ¿Algo ha entrado o salido del campo? Normalmente no acepta intent: una aparición no es buena o mala por aparecer. Lo evaluable puede estar dentro de lo que aparece.

shift

Comunica cambio de contexto. ¿Sigo en el mismo marco o ha cambiado el régimen? Cambiar de vista, de modo, avanzar en un wizard o entrar en edición reorganiza el marco desde el que se interpreta la interfaz. Normalmente no acepta intent.

sustain

Comunica continuidad operativa. ¿Esto sigue ocurriendo? No comunica resultado: comunica que una operación sigue viva. Si un proceso se vuelve problemático, lo más limpio es componer `sustain.progress` con `signal.warn + risk`, en lugar de convertir todo el proceso en evento evaluativo.

delegate

Comunica que cambia quién lleva la iniciativa o quién ejecuta la acción. ¿Actúo yo, actúa el sistema, o compartimos la acción? No comunica resultado ni urgencia por sí misma: el intent llega cuando la acción delegada produce una consecuencia. Aquí entran sugerir, planificar, autorizar, actuar por el usuario, esperar revisión, escalar y devolver el control. Su problema propio es la autoría legible: que se perciba quién acaba de actuar.

TABLA 8.2 Las ocho familias fundamentales

FAMILIA	PREGUNTA	QUÉ COMUNICA	TEMPORALIDAD	INTENT
contact	¿me ha sentido?	Registro de acción	Puntual	Leve
commit	¿quedó fijado?	Estado o consecuencia	Resolutiva	Sí
signal	¿debo atender?	Mensaje del sistema	Entrada + persist.	Sí
handle	¿lo controlo?	Manipulación directa	pick/carry/drop	En drop
emerge	¿apareció?	Presencia/ disponibilidad	enter/exit	Normalmente no
shift	¿cambió el marco?	Cambio de régimen	leave/arrive	Normalmente no
sustain	¿sigue?	Continuidad de proceso	start/progress/ end	Normalmente no

Lo que se pierde y lo que se conserva

La reducción no elimina las semánticas exploratorias. Las reubica.

TABLA 8.3 Reubicación de las semánticas exploratorias

SEMÁNTICA EXPLORATORIA	REUBICACIÓN EN FAMILIAS
feedback	contact
selection	commit.select
completion	commit.complete + fulfill
destruction	commit.delete + loss
notification	signal.notify
attention	signal.alert / signal.warn
emphasis	signal.emphasize
revelation	emerge.reveal
expansion	emerge.expand
context	shift si cambia marco; emerge si solo aparece superficie
navigation	shift.navigate
persistence	sustain
manipulation	handle.pick / carry / drop

§ 5

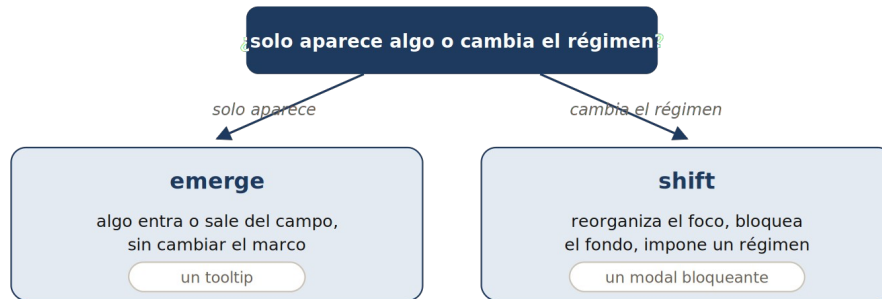
Tres casos difíciles

El caso context

Cuando algo aparece sin cambiar profundamente el régimen operativo, estamos cerca de emerge. Cuando reorganiza el foco, bloquea el fondo o impone un nuevo régimen, estamos en shift. Un tooltip es emerge; un modal bloqueante es shift. La pregunta: ¿solo aparece algo o cambia el régimen de operación?

FIGURA 8.4

emerge frente a shift



La frontera no está en si algo aparece, sino en si cambia el régimen desde el que se opera.

El caso signal

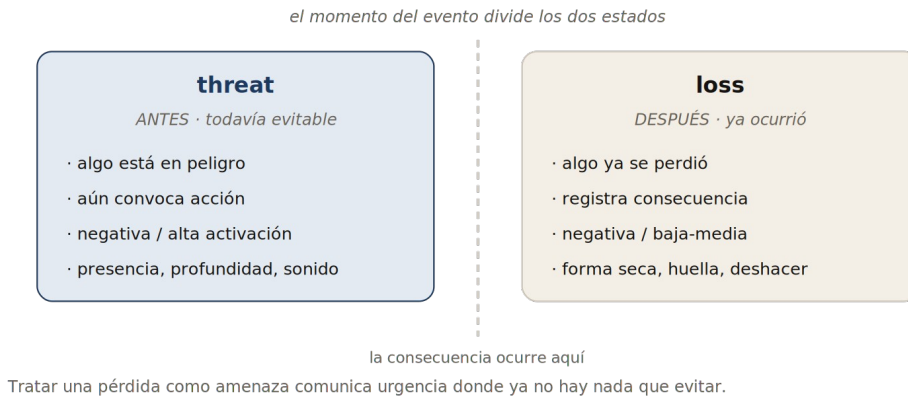
Las tres semánticas previas (notification, attention, emphasis) comparten una función profunda: el sistema intenta modificar la atención del usuario. Lo que cambia es la intensidad, el tono, la persistencia y el intent. La familia es la misma: signal. La variación ocurre por verbo, prioridad e intent.

El caso commit

commit no significa «guardar». Significa que algo queda fijado en el contexto operativo. Aquí conviene anticipar una distinción importante: threat comunica que algo está en peligro o exige reacción inmediata; loss comunica que algo ya se perdió. No son el mismo estado afectivo. Una amenaza todavía convoca acción; una pérdida consumada registra consecuencia. Tratar una pérdida como amenaza comunica urgencia donde ya no hay nada que evitar.

FIGURA 8.5

threat frente a loss



§ 6

Familias e intents

Las familias dicen qué tipo de evento ocurre. Los intents dicen cómo se evalúa afectivamente, cuando el evento es evaluable. Son seis: neutral, affirm, fulfill, risk, threat y loss. El Capítulo 10 los desarrolla uno a uno; su tabla canónica —qué declara cada uno, su valencia y sus riesgos de mal uso— está en el Apéndice B.

No todas las familias aceptan intent. emerge, shift y sustain normalmente no; contact puede aceptar modulación leve; commit y signal lo aceptan plenamente; handle lo acepta sobre todo en drop. El intent modula eventos evaluables, no eventos cuya función es estructurar el campo perceptivo.

§ 7

Ejemplo narrativo: eliminar una tarea

Imaginemos una aplicación de gestión de proyectos. El usuario ve una tarea con archivos adjuntos y comentarios. Abre el menú de acciones y pulsa «Eliminar».

- **1. contact.press** — el sistema registra el gesto. Todavía no se ha eliminado nada.
- **2. signal.warn + threat** — el sistema advierte. El usuario aún puede evitar la consecuencia: corresponde threat, no loss.
- **3. shift.enter-mode** — aparece un modal de confirmación. El modal reorganiza el contexto; el intent no pertenece al modal como contenedor, sino al mensaje evaluable dentro de él.
- **4. commit.delete + loss** — el usuario confirma y la tarea deja de existir. Aquí entra loss: no debe sonar ni moverse como threat. Necesita registrar consecuencia, no preparar reacción.
- **5. sustain.progress** — si hay sincronización remota. El sostenimiento no es negativo por sí mismo.
- **6. emerge.present** — aparece una barra de undo. La aparición no es la pérdida: la pérdida ya ocurrió en el paso 4.
- **7. signal.announce + neutral** — el sistema informa del resultado y ofrece deshacer.

FIGURA 8.6

Eliminar una tarea: la secuencia completa



Un solo gesto despliega siete eventos: el threat del paso 2 (aún evitable) no debe sonar como el loss del paso 4.

§ 8

Por qué no seis

Podría parecer tentador agrupar shift dentro de emerge. Pero cambiar de contexto no es lo mismo que hacer aparecer algo. Un tooltip aparece pero no cambia el marco; un cambio de vista puede no «aparecer» como capa, pero desplaza al usuario a otro régimen. Si eliminamos shift, la gramática pierde capacidad para nombrar una de las experiencias más frecuentes en interfaces complejas: cruzar un umbral operativo.

§ 9

Por qué no doce

Algunas de las semánticas iniciales se entienden mejor como verbos, variantes, fases o intents. Como muestra la tabla de reubicación, cada semántica exploratoria encuentra sitio dentro de las ocho familias. Si

conservamos todas como raíz, la gramática se vuelve más pesada y menos aprendible. La gramática no pierde precisión al reducirlas; gana organización.

§ 10

Síntesis

La taxonomía no pretende agotar todas las interfaces posibles ni fijar una verdad natural inmutable. Es una hipótesis teórica compacta: suficientemente pequeña para aprenderse, suficientemente rica para distinguir los cambios que una interfaz necesita hacer legibles.

La síntesis completa —pregunta perceptiva, relación con el intent, fase típica, canales dominantes y antipatrón capital de cada familia— queda recogida como tabla canónica en el Apéndice B.

Y una honestidad final: este capítulo ha resuelto casos frontera de la interfaz que ya sabemos mirar. Los que el presente pone delante —el texto que fluye de un modelo generativo, la interfaz optimista, la colaboración en vivo— se examinan en el Apéndice E. Allí se dice con claridad lo que la gramática aún no nombra.

Una familia semántica no es una etiqueta para un efecto. Es una respuesta estable a una pregunta perceptiva recurrente.

El siguiente capítulo desarrollará la distinción que atraviesa todo el sistema: por qué algunas familias y eventos aceptan intent y otras no. Porque no todos los cambios son mensaje. Algunos son marco.

09

Semánticas valenciales y transicionales

Qué eventos son mensaje y qué eventos son marco

PREGUNTA DEL CAPÍTULO

¿Por qué algunos eventos aceptan evaluación y otros solo cambian el marco donde se trabaja?

§ 1

La distinción central

El Capítulo 3 presentó varios fundamentos que ahora convergen en una distinción central.

Desde la valoración del evento, algunos eventos se evalúan respecto a metas, control, urgencia y consecuencia. Desde la Gestalt, algunos eventos funcionan como figura y otros como fondo. Desde las affordances, los eventos operativos pueden portar valencia, mientras los estructurales organizan posibilidades. Desde la atención, una señal puede reclamar procesamiento, pero una transición puede solo orientar.

Todas esas distinciones apuntan a una misma idea:

PRINCIPIO

Hay eventos que son mensaje. Hay eventos que son marco.

Los primeros pueden portar intent. Los segundos normalmente no.

El error no es usar intent. El error es aplicarlo al evento equivocado. Un modal puede contener una amenaza, pero el modal no es la amenaza. Un spinner puede preceder a un fallo, pero el spinner no es el fallo.

PRINCIPIO

El intent modula la figura evaluable, no el fondo estructural.

§ 2

Qué significa que un evento sea evaluable

Un evento es evaluable cuando puede juzgarse en relación con las metas, expectativas o riesgos del usuario. Puede ser bueno o malo, leve o grave, puede exigir acción inmediata o permitir revisión, puede resolver una tensión o abrir un problema.

Un evento transicional o estructural no porta, por sí mismo, un juicio afectivo. Su función es reorganizar el campo perceptivo, revelar disponibilidad, mantener continuidad o cambiar el contexto. Dice «algo apareció», «el marco ha cambiado» o «el proceso sigue vivo» — pero no dice todavía «bueno», «malo», «grave», «logrado» o «perdido».

TABLA 9.1 Evento evaluable y evento estructural

TIPO	QUÉ HACE	EJEMPLOS	INTENT
Evaluable	Porta juicio sobre el evento	Confirmar, fallar, advertir, perder	Sí
Estructural	Organiza el campo perceptivo	Abrir, expandir, navegar, sostener	Normal. no
Compuesto	Contiene marco y mensaje	Modal de error, wizard con validación	En el mensaje

§ 3

Neutralidad no es ausencia de intent

neutral no debería convertirse en una manera de decir «no hemos pensado si este evento acepta intent». Un `signal.announce` + neutral sí tiene sentido: el sistema comunica algo sin valencia fuerte. Pero un `emerge.open` no necesita intent neutro para funcionar. Su función no es evaluar, sino organizar presencia.

DISTINCIÓN

Un intent neutral es evaluación de baja valencia; la ausencia de intent indica que el evento no pertenece al plano evaluativo.

§ 4

Recordatorio: los seis intents

El Capítulo 10 desarrolla cada intent en detalle; la tabla canónica de los seis está en el Apéndice B. Como recordatorio, el mapa mínimo:

FIGURA 9.2

Recordatorio: los seis intents



Las familias dicen qué evento ocurre; los intents, cómo se evalúa cuando el evento es evaluable.

§ 5

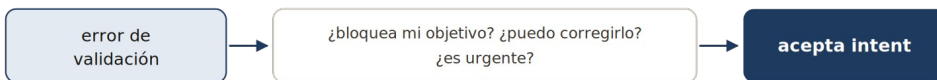
Fundamento: valoración del evento

La teoría de la valoración del evento explica por qué algunos eventos aceptan intent y otros no. La valoración del evento es el proceso por el cual un estímulo se evalúa en relación con metas, control, consecuencias y urgencia. Un error de validación dispara preguntas de valoración del evento: ¿bloquea mi objetivo? ¿puedo corregirlo? ¿es urgente? Pero la apertura de un panel o el cambio de vista no disparan el mismo tipo de evaluación.

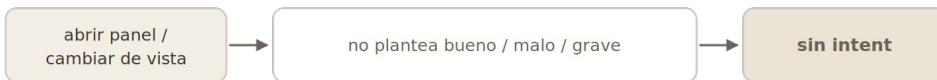
FIGURA 9.3

Por qué unos eventos aceptan intent y otros no

DISPARA VALORACIÓN → evaluable



NO DISPARA VALORACIÓN → estructural



El intent tiene sentido cuando el evento dispara una valoración; si no la dispara, no hay nada que evaluar.

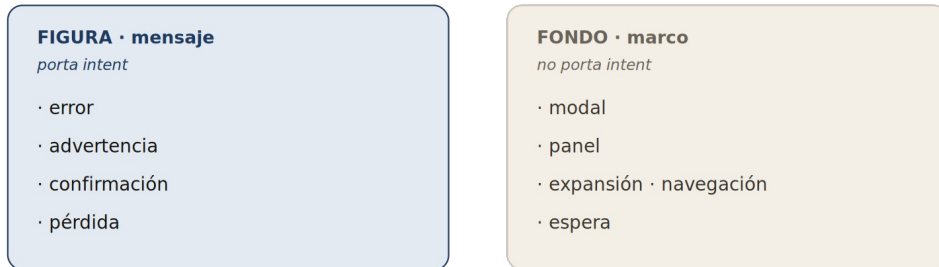
§ 6

Fundamento: figura y fondo

La percepción organiza el campo en figura y fondo. Algunos eventos son figura: el error, la advertencia, la confirmación, la pérdida. Otros son fondo estructural: el modal, el panel, la expansión, la navegación, la espera. La valencia no está en el modal como estructura. Está en el mensaje que el modal contiene.

FIGURA 9.4

Figura y fondo: qué evento porta el juicio



La valencia no está en el modal como estructura, sino en el mensaje que el modal contiene.

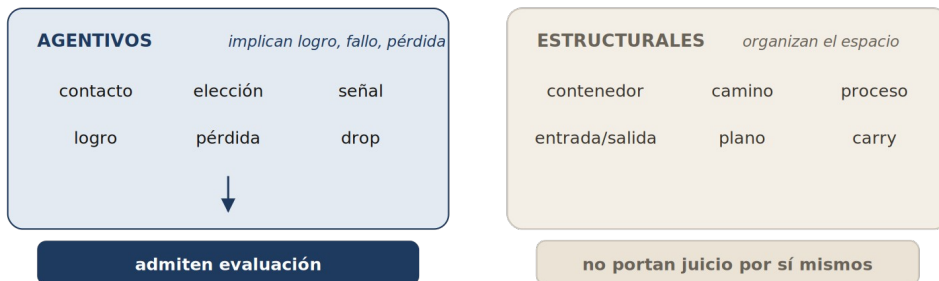
§ 7

Fundamento: esquemas sensoriomotores

Algunos esquemas son agentivos: contacto, elección, señal, logro, pérdida, manipulación/drop. Otros son estructurales: contenedor, camino, proceso, entrada/salida, plano, carry. Los esquemas agentivos tienden a admitir evaluación porque implican logro, fallo, contacto, elección, pérdida o control. Los estructurales organizan el campo perceptivo.

FIGURA 9.5

Esquemas sensoriomotores: agentivos y estructurales



Los esquemas agentivos implican logro, fallo o pérdida; los estructurales organizan dónde ocurren.

§ 8

Confirmaciones desde affordances, atención y afecto

Desde las affordances, los eventos operativos e informativos tienden a ser evaluables porque afectan a lo que el usuario puede hacer o debe interpretar. Las affordances estructurales organizan el campo sin portar juicio por sí mismas.

Desde la atención, `signal.alert` puede reclamar evaluación porque introduce una figura que el usuario debe atender. `emerge`, `shift` y `sustain` muchas veces redirigen o acompañan la atención sin convertir ese cambio en mensaje afectivo.

Desde el modelo valencia-activación, los intents tienen sentido cuando un evento puede ubicarse en una región evaluativa. Una expansión no es positiva por expandirse. Una navegación no es exitosa por cambiar de vista. Una espera no es amenaza por durar.

La convergencia es clara: el intent pertenece al evento que porta juicio, no al evento que organiza el espacio donde ese juicio aparece.

§ 9

Reglas de composición

Regla 1 — El intent va en el evento evaluable. Incorrecto: `emerge.open + risk`. Correcto: `emerge.open → signal.warn + risk`.

Regla 2 — El marco puede contener mensaje, pero no absorberlo. Incorrecto: `shift.enter-mode + threat`. Correcto: `shift.enter-mode → signal.alert + threat`.

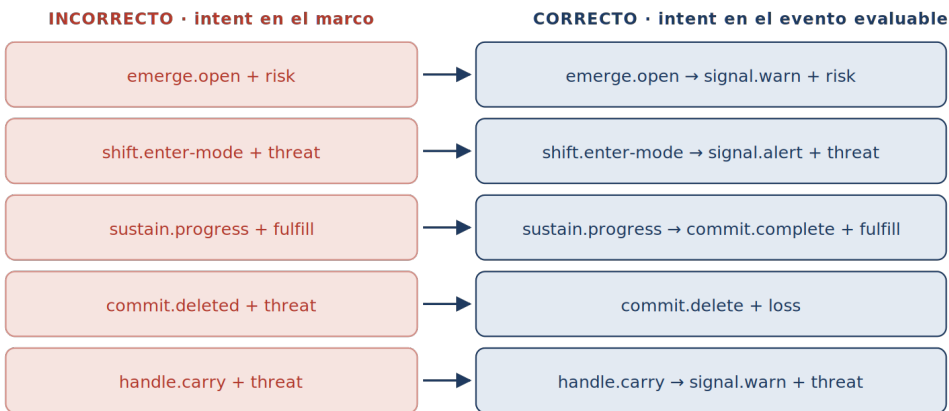
Regla 3 — El proceso no es el resultado. Incorrecto: `sustain.progress + fulfill`. Correcto: `sustain.progress → commit.complete + fulfill`.

Regla 4 — La amenaza no es la pérdida. Incorrecto: `commit.delete + threat`. Correcto: `commit.delete + loss`. Y si antes hay advertencia: `signal.warn + threat` → `commit.delete + loss`.

Regla 5 — En manipulación, el `carry` no debe cargar el juicio. Incorrecto: `handle.carry + threat`. Correcto: `handle.carry` → `signal.warn + threat` (si zona destructiva) → `commit.delete + loss` (si se suelta ahí).

FIGURA 9.6

Las cinco reglas de composición del intent



El patrón es siempre el mismo:

el marco (`emerge`, `shift`, `sustain`, `handle.carry`) encadena hacia el evento evaluable (`signal`, `commit`), y el `intent` vive allí, no en el contenedor.

Las reglas 1 y 2 admiten una excepción, y conviene declararla con la misma precisión con que se declaró la regla. Existe un caso en que el evento estructural puede portar un tono no neutral: cuando su única razón de ocurrir es hacer perceptible una situación evaluable. Una superficie que se abre solo porque hay una acción destructiva pendiente no es un contenedor que trae una advertencia: su aparición es ya la advertencia. Diferir todo el tono al mensaje interior deja mudo el primer momento perceptivo —la interrupción misma—, y esa mudez se lee como infra-reacción ante un contexto grave.

La prueba es sustractiva: si se quita el evento estructural, ¿la advertencia sobrevive en otro sitio? Si sobrevive —una señal previa, un mensaje interior, un estado que ya la porta—, el evento es marco y queda neutral. Si la aparición es la única advertencia, puede portar el tono. Y la excepción es cara: debe declararse como tal, con su justificación, nunca asumirse. Una superficie que aparece por cualquier otra razón —mostrar opciones, cambiar de contexto, contener— sigue siendo marco. El alcance de la excepción es estrecho: aparición (emerge) y cruce de marco (shift); el contacto queda fuera.

LO QUE LA PRÁCTICA CORRIGIÓ

Esta excepción no nació en la teoría: la impuso la práctica. Una implementación de referencia de esta gramática, al construir un diálogo de confirmación destructiva, encontró que mantener neutral la aparición dejaba el primer momento perceptivo sin evaluación — el tono llegaba tarde, con el mensaje interior. La regla general resistió; el caso frontera exigió nombre, prueba y coste. Así evoluciona una gramática: por defecto firme, excepción declarada.

§ 10

Ejemplo: archivo no guardado al cerrar

El usuario edita un documento y pulsa cerrar. La interfaz detecta cambios sin guardar.

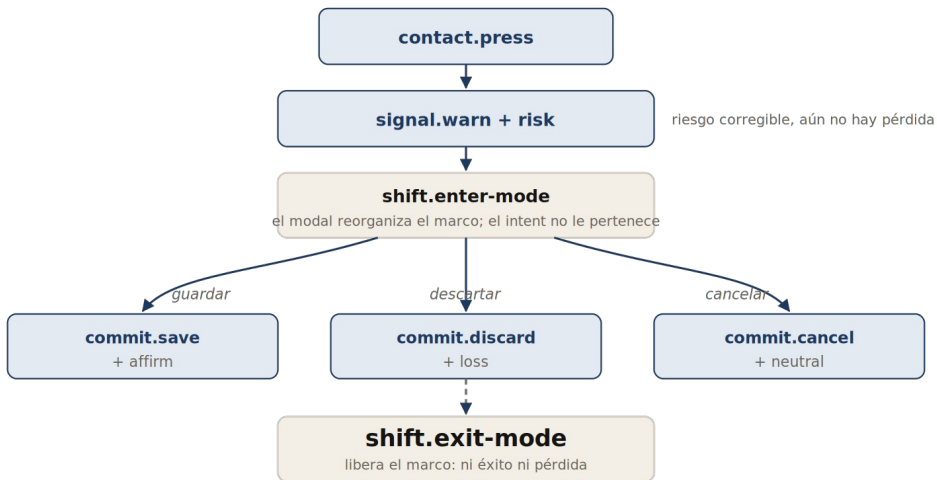
1. `contact.press` — el sistema confirma que recibió el gesto.
2. `signal.warn + risk` — el sistema detecta cambios sin guardar. Todavía no hay pérdida; hay riesgo corregible.
3. `shift.enter-mode` — aparece un diálogo que bloquea la salida. El modal reorganiza el marco. El intent no pertenece al modal. Nótese que este diálogo *no* es la aparición-que-advierte: la advertencia ya vive en el paso 2 (`signal.warn + risk`) y sobrevive sin el modal. El test sustractivo lo resuelve: es marco, y queda neutral.

Si el usuario elige guardar: `commit.save + affirm`. Si descarta: `commit.discard + loss`. Si cancela: `commit.cancel + neutral`.

4. `shift.exit-mode` — el modal se cierra. La salida del modal no es éxito ni pérdida. Libera el marco.

FIGURA 9.7

Archivo no guardado al cerrar



El mismo modal conduce a affirm, loss o neutral según la decisión; el marco no carga ninguno de los tres.

§ 11

Consecuencias para diseño e implementación

Reduce sobreactuación: si solo los eventos evaluables reciben intent, la interfaz reserva la expresión afectiva para donde realmente la necesita.

Mejora la composición: cada parte de una operación compuesta tiene su función asignada.

Mejora la accesibilidad: si sabemos qué evento porta el significado evaluable, podemos redistribuirlo mejor cuando un canal se reduce.

Evita falsas emociones: una interfaz no debería comunicar amenaza cuando hay pérdida, ni celebración cuando solo hay confirmación suave.

§ 12

Síntesis

Este capítulo ha introducido una distinción central: no todos los eventos aceptan intent porque no todos los eventos son evaluables. Esta distinción no convierte los eventos estructurales en secundarios ni prohíbe que una transición tenga atmósfera. Solo establece que el intent debe vivir donde hay evaluación operativa, no donde solo hay organización del campo.

TABLA 9.2 Síntesis: evaluabilidad por familia

FAMILIA	EVALUABILIDAD	REGLA PRÁCTICA
contact	Leve o anticipatoria	Puede anticipar peso pero no porta resultado
commit	Alta	Es el lugar natural del intent
signal	Alta	Comunica el tono del mensaje
handle	Mixta; sobre todo en drop	El carry no carga juicio; el drop sí
emerge	Estructural	El intent va en lo que emerge, no en la aparición
shift	Estructural	El intent va en el evento dentro del nuevo marco
sustain	Estructural	Si falla, componer con signal o commit

El intent no es una propiedad del componente, ni del contenedor, ni de la transición. Es una modulación evaluativa del evento que porta juicio.

El siguiente capítulo desarrollará en detalle el sistema de intents: por qué success / warning / danger / info no basta, por qué loss debe separarse de

threat, por qué affirm no es lo mismo que fulfill, y por qué info quizá no es un intent sino una función de señal.

TABLA 9.3 Antipatrones de evaluación: confundir marco y mensaje

ANTIPATRÓN	QUÉ CONFUNDE	CORRECCIÓN
Modal rojo	Marco con mensaje	shift + signal.threat
Spinner de error	Proceso con resultado	sustain + commit.fail
Success de navegación	Cambio de contexto con logro	shift + commit.fulfill si procede
Danger para pérdida	Amenaza con pérdida	commit.delete + loss
Alerta para toda aparición	Aparición con señal	emerge + signal solo si hay mensaje

10

Intent: más allá del semáforo

Por qué success, warning, danger e info no bastan

El capítulo anterior estableció una regla fundamental:

PRINCIPIO

El intent vive en el evento evaluable, no en el contenedor estructural.

Un modal no es peligroso por abrirse. Una carga no es negativa por durar. Una navegación no es exitosa por cambiar de pantalla. El intent debe aplicarse allí donde el evento porta juicio. Ahora aparece otra pregunta: ¿qué intents merece tener una interfaz?

PREGUNTA DEL CAPÍTULO

¿Qué ocurre cuando dejamos de tratar el intent como color y empezamos a tratarlo como evaluación perceptiva del evento?

§ 1

El intent como problema no examinado

Existe en el diseño de interfaces una convención tan extendida que ha dejado de parecer convención. Cuando algo ha ido mal, usamos rojo; cuando algo requiere precaución, amarillo; cuando algo ha ido bien, verde; cuando algo es informativo, azul. A ese conjunto lo llamamos danger, warning, success, info; los nombres cambian de framework en framework, pero la estructura permanece.

Lo notable no es que usemos esa convención. Lo notable es que casi nunca preguntamos de dónde viene, qué cubre, qué omite y qué mezcla. ¿Por qué success debe cubrir tanto una operación importante completada como un autoguardado menor? ¿Por qué danger debe cubrir tanto una amenaza activa como una pérdida ya consumada? ¿Por qué warning mezcla un riesgo corregible y, a veces, una amenaza seria? ¿Por qué info aparece junto a los intents si muchas veces no comunica valencia, sino solo saliencia atencional?

Ese desplazamiento —del color como etiqueta a la evaluación como región— es el que este capítulo se propone realizar.

§ 2

La herencia del semáforo

La taxonomía heredada no nació en las interfaces digitales. Antes de que existieran botones verdes de success o alertas rojas de danger, esos colores ya cargaban con una historia de señalización industrial, ferroviaria y vial.

Esa historia tiene una fecha y una escena. En diciembre de 1868, frente al Parlamento de Londres, se instaló el primer semáforo del mundo: un poste con brazos de señal y dos lámparas de gas, una roja y una verde, que un policía accionaba a mano. No lo había ideado un urbanista, sino un ingeniero de ferrocarriles, John Peake Knight, que se limitó a trasladar a la calle lo que ya funcionaba en las vías. Los colores tampoco eran suyos: el rojo venía de mucho más atrás —de la señalización marítima y de

incendios, donde llevaba siglos significando peligro—, y el ferrocarril lo había adoptado para ordenar «pare». Apenas un mes después, una fuga de gas hizo explotar el aparato e hirió gravemente al policía que lo manejaba, y se retiró. Pero la asociación ya había echado a andar: rojo es pare, verde es siga. Décadas más tarde, la electricidad y el automóvil la convertirían en el semáforo de tres luces que hoy reconoce todo el mundo.

El rojo, el amarillo y el verde pertenecían a entornos donde una señal debía reconocerse rápido, a distancia y con consecuencias físicas claras. Eran instrucciones operativas, y esa estructura era adecuada para señalización porque resolvía un problema urgente: decisiones rápidas con bajo coste interpretativo. Pero no era una teoría afectiva completa. No pretendía distinguir amenaza activa de pérdida consumada, ni confirmación suave de culminación significativa.

Cuando las interfaces gráficas empezaron a necesitar comunicar estados, heredaron esa estructura porque ya era comprensible. El diseño digital no inventó la asociación: la recibió. El problema no fue heredar; el problema fue olvidar que se estaba heredando. Con el tiempo, la asociación cromática dejó de parecer convención histórica y empezó a parecer estructura natural: el rojo pasó a llamarse danger; el verde, success; el amarillo, warning; y el azul terminó ocupando el cajón residual, info. Ahí ocurrió la mutación decisiva:

PRINCIPIO

Una convención cromática empezó a funcionar como si fuera una taxonomía semántica.

§ 3

De la carretera a la pantalla

La web estabilizó esa mutación. Frameworks como Bootstrap ofrecieron clases contextuales simples —success, warning, danger, info—, una decisión razonable y barata cognitivamente. Pero al entrar en CSS, tokens,

componentes y guías, dejó de parecer un atajo: se convirtió en vocabulario. El núcleo siguió siendo el mismo: positivo, precaución, negativo grave, informativo, neutral.

Ese núcleo parecía suficiente mientras el intent solo modificaba color. Pero en una interfaz perceptiva multicanal, el intent puede modular motion, sonido, duración, ritmo, intensidad, persistencia, profundidad, forma, háptica y silencio. En ese momento la convención heredada muestra sus límites: una cosa es elegir un color; otra muy distinta, decidir cómo debe moverse, sonar, durar, aparecer, persistir o sentirse un evento.

§ 4

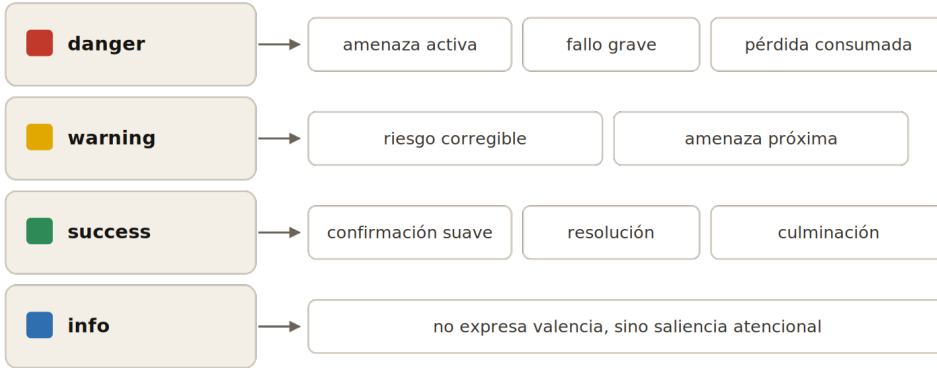
Qué codifica realmente la convención heredada

Si examinamos success, warning, danger e info, vemos que no forman una taxonomía afectiva completa: forman una taxonomía cromática útil. Eso no las vuelve inútiles; las vuelve insuficientes. Danger mezcla amenaza activa, fallo grave y pérdida consumada. Success mezcla confirmación suave, resolución positiva y culminación. Warning no distingue el riesgo corregible de la amenaza próxima. Info muchas veces no expresa evaluación afectiva, sino saliencia atencional.

FIGURA 10.1

Lo que la convención heredada mezcla

cada etiqueta es, en el fondo, un color que reúne eventos afectivamente distintos



Codifica colores útiles, pero insuficientes: mezcla amenaza con pérdida, confirmación con culminación.

PRINCIPIO

La convención heredada codifica colores útiles. La gramática del intent debe codificar regiones evaluativas del evento.

Por eso necesitamos separar: threat $\neq$ loss, affirm $\neq$ fulfill, info $\neq$ intent.

§ 5

Qué debería ser un intent

Un intent es la modulación evaluativa de un evento, y esa definición impone cuatro condiciones. Primero, pertenece al evento, no al componente. Segundo, pertenece al plano evaluativo, no a cualquier cambio. Tercero, debe poder modular varios canales. Cuarto, debe expresar una región afectiva funcional, no una etiqueta genérica.

§ 6

El espacio afectivo

El sistema de intents se apoya en el espacio valencia/activación descrito por James A. Russell —no como teoría completa de la emoción, sino como

base operativa—. La valencia distingue lo positivo de lo negativo; la activación distingue baja y alta movilización atencional.

Con esas dos dimensiones, las diferencias que el semáforo aplasta se vuelven visibles. No es lo mismo threat que loss: ambos son negativos, pero uno tiene alta activación y convoca acción, y el otro registra una consecuencia consumada con activación más baja. Tampoco es lo mismo affirm que fulfill: ambos son positivos, pero uno confirma suavemente y el otro resuelve una tensión.

§ 7

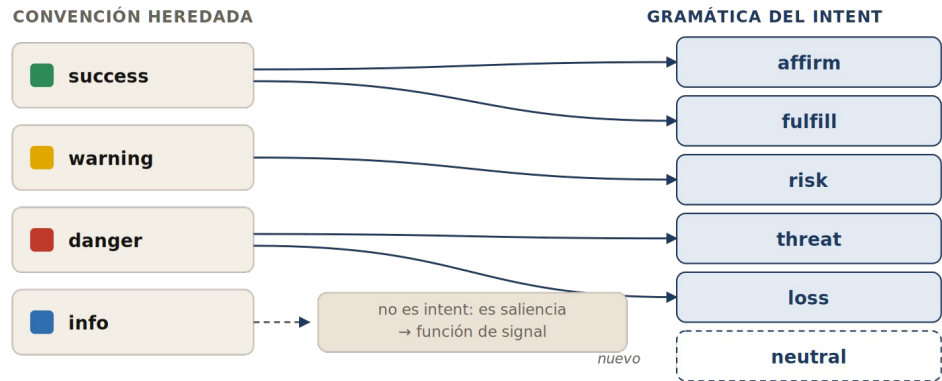
Las seis regiones evaluativas

Sobre ese espacio, el sistema propone seis regiones: neutral, affirm, fulfill, risk, threat y loss. Cada una ocupa una zona distinta de valencia y activación, y cada una tiene una lectura característica.

TABLA 10.1 Los seis intents como regiones del espacio evaluativo

INTENT	REGIÓN EVALUATIVA	LECTURA
neutral	baja activación, sin valencia fuerte	el evento no requiere evaluación afectiva significativa
affirm	valencia positiva, baja activación	confirma sin alterar significativamente el foco
fulfill	valencia positiva, activación media-alta	marca resolución de tensión o culminación
risk	valencia negativa, activación media	problema corregible que requiere atención
threat	valencia negativa, alta activación	exige atención urgente y posible acción inmediata
loss	valencia negativa, menor activación y posterioridad	registra consecuencia ya consumada

FIGURA 10.2
Del semáforo a seis regiones evaluativas



threat ≠ loss, affirm ≠ fulfill, info ≠ intent: cada región recupera una distinción que el color mezclaba.

PRINCIPIO

Los intents no son colores ni estilos. Son regiones evaluativas del evento que pueden expresarse mediante distintos canales de expresión.

§ 8

Lo que la convención heredada pierde

La separación se aprecia mejor en contraste. Threat convoca acción; loss registra consecuencia. Affirm confirma normalidad; fulfill resuelve tensión. Info no responde a cómo se evalúa el evento, sino a si debe ser notado: por eso no es un intent, sino una cuestión de saliencia.

TABLA 10.2 Convenciones del semáforo y separación propuesta

CONVENCIÓN	QUÉ MEZCLA	SEPARACIÓN PROPUESTA
danger / error	amenaza, fallo, pérdida	threat, risk, loss
warning	riesgo corregible, amenaza	risk, a veces threat
success	confirmación, resolución, logro	affirm, fulfill

info	saliencia, mensaje neutro	signal.announce + neutral
------	---------------------------	---------------------------

§ 9

Cómo modula el intent

El intent no se expresa solo en color. Puede hacerse perceptible mediante duración, energía de motion, timbre sonoro, persistencia, profundidad, forma, háptica o silencio. La Parte II recorrerá esos canales uno a uno; lo que importa aquí es que la evaluación del evento tiene muchas más voces que el pigmento.

§ 10

Neutralidad no es ausencia de intent

Neutral ocupa una región de baja activación sin carga evaluativa fuerte. Pero no todo evento sin carga evaluativa necesita recibir neutral: algunos eventos simplemente no pertenecen al plano evaluativo. `emerge.open`, `shift.navigate` o `sustain.progress` pueden no necesitar intent, y ese silencio evaluativo es información, no descuido.

§ 11

Familia e intent

La división de trabajo del sistema queda así: la familia dice qué tipo de evento está ocurriendo; el intent dice en qué región evaluativa cae ese evento; los canales dicen cómo se vuelve perceptible esa evaluación.

§ 12

Antipatrones

El semáforo redivivo. Mapear los seis intents de vuelta a `success`, `warning` y `danger`. Si `affirm` y `fulfill` comparten expresión, y `risk`, `threat` y `loss` comparten alarma, la gramática se ha perdido: vuelve la ambigüedad

que los intents venían a resolver. Cada intent necesita una firma diferenciable, aunque comparta canal.

Celebración prematura. Declarar affirm o fulfill cuando el evento aún no se ha resuelto. Lo que se celebra es la petición, no la consecuencia. El intent positivo llega con el commit, no con el contacto.

Intent decorativo. Asignar intent a eventos que no son evaluables — aperturas, transiciones, esperas— para dar tono. El intent no es tono: es evaluación. Si el evento es marco, no porta intent; la evaluación viaja en la figura.

TABLA 10.3 Antipatrones del intent

ANTIPATRÓN	QUÉ OCULTA
derivar intent de color	oculta el espacio valencia/activación
usar danger para pérdida	oculta la diferencia temporal antes/después
usar success para todo	oculta la diferencia entre baja y mayor activación positiva
usar info como intent	oculta la diferencia entre saliencia y evaluación

§ 13

Ejemplo narrativo

Cuatro momentos de un mismo producto muestran las regiones en acción. Un autoguardado es commit.save + affirm: valencia positiva, baja activación. Un envío completado es commit.submit + fulfill: valencia positiva, activación media-alta. La advertencia previa a una eliminación es signal.warn + threat: valencia negativa, alta activación, antes de la consecuencia. Y la eliminación consumada es commit.delete + loss: valencia negativa, menor activación, después de la consecuencia.

§ 14

Principios

Tres principios condensan el capítulo. El intent pertenece al evento, no al componente. El intent modula el plano evaluable, no el estructural. Y el intent se deriva de la evaluación del evento en el espacio valencia/activación, no del color con que se represente.

§ 15

Síntesis

Con este capítulo, el sistema de intents deja de ser una variación cromática y pasa a funcionar como capa evaluativa del evento. A partir de ahora, color, motion, sonido, forma, presencia, háptica y tiempo no definirán el intent: lo expresarán.

Cierre de la Parte I

Con este capítulo se cierra la Parte I. Empezamos redefiniendo la interfaz como la dimensión perceptible de un contexto operativo, y terminamos con un sistema mínimo de piezas: el evento como unidad de cambio, la gramática como forma de organizarlo, la interacción semántica como su unidad de lenguaje, las ocho familias como sus clases perceptivas y los seis intents como su plano evaluativo.

Nada de eso dice todavía cómo se percibe un evento concreto. Esa es la tarea de la Parte II: recorrer los canales por los que un evento se vuelve sensible —tiempo, motion, presencia, forma, color, sonido, háptica— y aprender a coordinarlos para que digan una sola cosa.



Canales

Channels

Por dónde se vuelve sensible un evento: tiempo, motion, presencia, forma, color, sonido y háptica; su accesibilidad y su coordinación.

11

Los canales de expresión del evento

Por dónde se vuelve sensible un evento

PREGUNTA DEL CAPÍTULO

¿Por dónde llega un evento al usuario, y qué papel juega cada canal en su lectura?

El usuario acaba de guardar un documento. ¿Cómo lo sabe? Quizá apareció un instante un texto —«Guardado»—, o un icono cambió de color, o algo se asentó con un leve rebote, o sonó un clic discreto, o el teléfono devolvió un pulso breve bajo el dedo. El evento era el mismo —algo quedó fijado—, pero podía tomar cuerpo de maneras muy distintas. Ninguna de ellas es el evento: todas son vías por las que el evento se vuelve perceptible.

Hasta aquí hemos hablado de eventos. Hemos definido la interfaz como la dimensión perceptible de un contexto operativo, y hemos explicado que el usuario no percibe solo componentes, sino transformaciones: algo responde, algo se fija, algo aparece, algo reclama atención, algo se pierde,

algo continúa, algo cambia de marco. Hemos llamado evento a cada uno de esos cambios con significado operativo: los que alteran lo que el usuario puede hacer, atender o interpretar.

Pero un evento no se percibe en abstracto. Para que exista en la experiencia del usuario, debe tomar cuerpo en algún canal: debe durar, moverse, aparecer, sonar, cambiar de color, ocupar un plano, modificar una forma, dejar una marca, producir contacto o retirarse. A esa materia sensible la llamaremos canales de expresión de la interfaz.

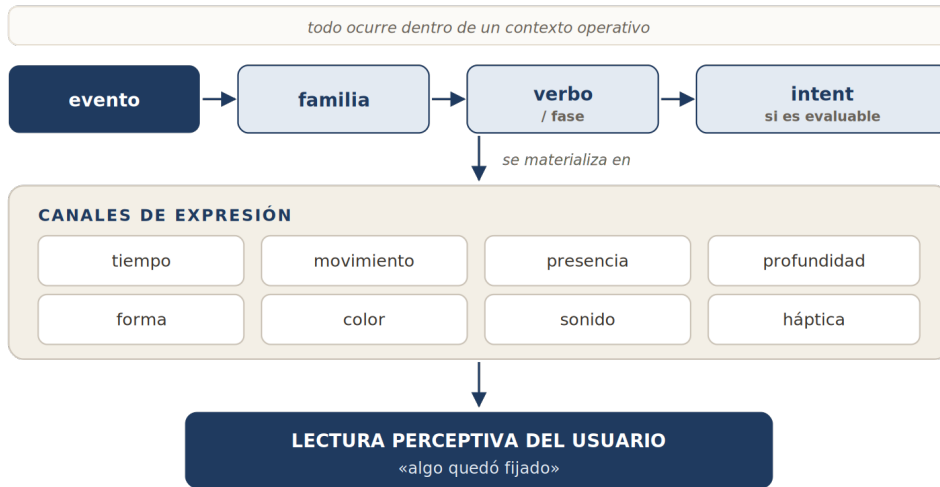
Un canal de expresión no es una propiedad técnica. No es opacity, transform, box-shadow, background-color, AudioContext o vibration API: esas son herramientas. El canal es lo que el usuario percibe a través de ellas. El desarrollador escribe transform: scale(.97), pero el usuario no percibe «un transform»: percibe que el botón cede brevemente a su acción. El desarrollador escribe box-shadow, pero el usuario percibe elevación, plano, separación o prioridad. El desarrollador dispara un sonido breve, pero el usuario percibe confirmación, aviso, cierre, amenaza o recompensa.

El problema de muchas interfaces no es que carezcan de efectos. Es que esos efectos no están organizados como canales de significado.

El mapa completo: del evento a la lectura perceptiva

FIGURA 11.1

El mapa completo: del evento a la lectura perceptiva



La gramática no empieza en el color ni en motion: empieza en el evento y solo al final se materializa en canales.

Antes de entrar canal por canal conviene situar la arquitectura completa del sistema. La gramática que este libro propone no empieza en los canales —no empieza en el color, ni en motion, ni en sonido—: empieza en el evento. Un evento ocurre dentro de un contexto operativo; pertenece a una familia semántica; puede concretarse mediante un verbo o una fase; si es evaluable, puede recibir un intent; y solo entonces se materializa mediante canales de expresión.

La secuencia completa es: evento → familia semántica → verbo / fase → evaluación si procede → intent → canales de expresión → lectura unificada.

Tres ejemplos muestran la cadena entera. Un `commit.complete` + `fulfill` se materializa en motion resolutivo, sonido breve ascendente, color positivo y marca persistente, y se lee «objetivo cumplido». Un `signal.warn` + `risk` toma forma de callout con color de advertencia moderada, icono y texto persistente, y se lee «revisa esto». Un `contact.press` pide baja latencia, estado pressed, micro motion opcional y háptica opcional, y se lee «el sistema ha sentido mi acción».

Esta secuencia evita un error frecuente: empezar por el efecto. Si empezamos por el efecto, preguntamos qué animación queda bien aquí; si empezamos por el canal, qué sonido usamos; si empezamos por el color, si esto va en rojo, verde o amarillo. Pero la pregunta correcta es anterior —¿qué evento necesita percibir el usuario?— y solo después podemos decidir por qué canales conviene expresarlo.

PRINCIPIO

Los canales no se coordinan entre sí en abstracto. Se coordinan alrededor de un evento: primero se identifica el evento, después se decide qué canales lo vuelven perceptibles.

§ 1

Por qué hablar de canales

Durante mucho tiempo hemos tratado la interfaz como si tuviera una división simple —visual, auditivo, háptico— y dentro de «visual» hemos metido casi todo: color, forma, layout, sombra, movimiento, aparición, profundidad, iconografía, estado, foco, contraste. Pero esa agrupación es demasiado gruesa. Un cambio de color no comunica lo mismo que un desplazamiento, ni una sombra lo mismo que un borde, ni una aparición lo mismo que un temblor, ni una transición lenta lo mismo que una respuesta instantánea. Todo puede entrar por la vista, pero no todo pertenece al mismo canal de expresión.

Cada canal, de hecho, tiende a transportar significados propios. El tiempo comunica inmediatez, espera, continuidad, persistencia o resolución. Motion comunica causalidad, trayectoria, energía, masa y cambio. La presencia comunica aparición, retirada, disponibilidad o desaparición, y la profundidad, plano, jerarquía, elevación, subordinación o régimen operativo. La forma comunica límite, affordance, contorno, agrupación, iconografía y estabilidad. El color comunica saliencia, estado, valencia y refuerzo, pero no debería cargar solo con todo el significado. El sonido

comunica urgencia, cierre, confirmación, alarma, pérdida o recompensa con una velocidad que el canal visual no siempre alcanza. Y la háptica comunica contacto, impacto, resistencia, fricción y encaje, aunque su disponibilidad dependa mucho del dispositivo y, en la web, siga siendo limitada.

§ 2

Fundamento psicoperceptivo: por qué no basta con decir «visual»

La separación entre canales no nace de una ocurrencia terminológica. Nace de una observación básica: la percepción humana no procesa todas las señales de una interfaz como si fueran una sola materia uniforme. Incluso dentro de lo que solemos llamar visual hay dimensiones perceptivas distintas: el color no comunica lo mismo que la forma, la forma no comunica lo mismo que la profundidad, la profundidad no comunica lo mismo que el movimiento. Todas pueden llegar por la vista, pero no cumplen la misma función cognitiva.

La psicología Gestalt mostró que la percepción organiza el campo visual según principios como proximidad, semejanza, continuidad, cierre, región común, figura/fondo y destino común. Esto ayuda a explicar por qué una interfaz no se percibe como una lista de elementos, sino como un campo organizado de relaciones.

La teoría de affordances añade otra dimensión: no percibimos solo propiedades visibles, sino posibilidades de acción. Un botón no es solo una forma rectangular; es algo que ofrece ser presionado. Un objeto elevado no es solo un objeto con sombra; puede sugerir manipulación. Un campo con borde activo no es solo un contorno; indica disponibilidad de entrada.

La atención introduce un tercer plano. Algunas señales orientan, otras interrumpen, otras sostienen y otras fatigan. Por eso no basta con decir que algo «se ve»: hay que preguntar qué demanda atencional genera.

La integración multisensorial añade todavía otra capa. Cuando motion, sonido, háptica y color ocurren de manera temporal y semánticamente congruente, el usuario puede percibirlos como un único evento. Cuando se desacoplan, los percibe como señales separadas o incluso contradictorias.

Por eso este libro distingue canales de expresión. No porque cada canal tenga fronteras absolutas, sino porque cada uno tiende a transportar tipos distintos de significado.

§ 3

Propiedad técnica y canal de expresión

Una misma propiedad técnica puede participar en varios canales:

- **opacity puede comunicar aparición, retirada, debilitamiento, indisponibilidad o subordinación del fondo;**
- **scale puede comunicar contacto, profundidad, énfasis, celebración o manipulación;**
- **box-shadow puede comunicar profundidad, elevación, foco, manipulación o cambio de plano;**
- **el color puede comunicar estado, categoría, saliencia, valencia o marca;**
- **el sonido puede comunicar contacto, alerta, cierre, pérdida o logro;**
- **la vibración puede comunicar contacto, impacto, encaje, rechazo o amenaza corporal.**

Por eso no basta con definir tokens técnicos. Hace falta una capa semántica que pregunte: ¿qué dimensión del evento comunica esta propiedad?

§ 4

El evento como composición perceptiva

Una interacción semántica no se expresa necesariamente por un solo canal. Un `contact.press` puede combinar tiempo breve, micro motion, estado visual y quizá háptica. Un `commit.complete + fulfill` puede combinar motion resolutivo, sonido ascendente, color positivo y marca persistente. Un `signal.warn + risk` puede combinar color moderado, forma de callout, icono, texto y persistencia. Un `handle.pick` puede combinar profundidad, sombra, motion de elevación, forma de agarre y háptica opcional. Un `sustain.progress` puede combinar tiempo, presencia persistente, barra, texto de estado y motion suave.

Lo importante es que el usuario no perciba esos canales como piezas separadas, sino como una lectura coherente del evento. La gramática no pregunta primero «¿qué animación ponemos?»; pregunta qué evento debe hacerse perceptible y qué canales lo expresan mejor.

§ 5

La web como campo principal de ejemplos

Este libro toma la web como campo principal de observación. No porque la gramática se limite a la web, sino porque en la web la fragmentación de los canales se vuelve especialmente visible. El desarrollador trabaja con DOM, CSS, JavaScript, estados, clases, media queries, eventos técnicos, tokens, componentes, Web Audio API y accesibilidad; cada capa tiene su propia lógica. Y, sin embargo, el usuario no experimenta esas capas por separado.

La web también muestra muy bien las asimetrías entre canales:

- **color, forma, presencia y motion están muy disponibles;**
- **sonido existe, pero está condicionado por políticas del navegador, permisos, contexto de uso y cultura del silencio;**

- **háptica existe de manera limitada e inconsistente;**
- **profundidad es simulada, pero perceptivamente poderosa;**
- **tiempo y latencia dependen de rendimiento, red, dispositivo y sincronización del render.**

La web será nuestro laboratorio principal. No será el límite de la teoría.

§ 6

Síntesis

Un evento necesita hacerse sensible. Los canales de expresión no son efectos decorativos ni propiedades técnicas aisladas: son vías por las que el usuario percibe que algo responde, aparece, se fija, reclama atención, se pierde, continúa o cambia de marco. La tesis del capítulo puede formularse así:

PRINCIPIO

El evento es la unidad de significado. Los canales son la materia sensible que lo hace perceptible.

12

Tiempo, latencia y sincronía

El canal que atraviesa todos los canales

PREGUNTA DEL CAPÍTULO

¿Qué comunica el tiempo de una interfaz antes de que ningún otro canal diga nada?

Basta pulsar una tecla y mirar la pantalla. Si la letra aparece antes de una décima de segundo, no se percibe espera alguna: surge bajo el dedo, es del que escribe. Si esa demora se estira hasta cerca de un segundo, todavía se sigue el hilo, pero ya se nota que el sistema responde —no que uno actúa—. Y pasada la decena de segundos, la atención se ha marchado a otra parte: la tarea se rompió. Son umbrales viejos y sorprendentemente estables, porque no dependen de la máquina, sino de la percepción.

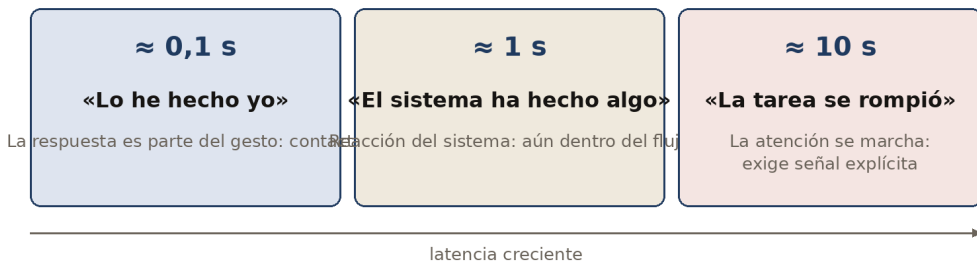
Antes de que algo se mueva, suene, vibre o cambie de color, ocurre en el tiempo. El tiempo no es una propiedad secundaria de la interfaz: es el canal que organiza la relación entre acción, espera, continuidad, consecuencia y memoria. Una interfaz puede sentirse inmediata, fluida,

lenta, rota, viva, bloqueada o concluida según su temporalidad. La misma respuesta, con el mismo color y el mismo componente, cambia de significado si llega demasiado tarde o demasiado pronto.

En esta gramática de eventos, el tiempo no es solo duración de animación. Es una forma de significado.

FIGURA 12.1

Tres ventanas de percepción temporal



Los umbrales no dependen de la máquina, sino de la percepción (Miller, 1968; Nielsen, 1993).

§ 1

La dimensión temporal de la interfaz

Solemos pensar la interfaz como espacio: una pantalla, una disposición, un layout, una jerarquía. Pero la interacción no ocurre solo en el espacio; ocurre en secuencias. El usuario actúa, el sistema responde, algo empieza, algo continúa, algo se detiene, algo termina, algo queda. Y cada momento tiene significado.

Una respuesta inmediata comunica contacto. Una espera sin señal comunica duda; con señal, comunica proceso. Una culminación comunica cierre; una señal persistente, relevancia. Una desaparición demasiado

rápida puede parecer error o pérdida, y una notificación demasiado larga puede convertirse en ruido. El tiempo convierte un cambio en evento.

§ 2

Fundamento psicoperceptivo: tiempo, agencia y espera

El tiempo es uno de los canales más estudiados en interacción humano-computador, aunque a menudo se trate solo como rendimiento técnico. En HCI clásica se han propuesto umbrales temporales aproximados que siguen siendo útiles como orientación (Miller, 1968; Nielsen, 1993). Una respuesta casi inmediata tiende a sentirse como parte del gesto; una demora breve puede mantenerse dentro del flujo de pensamiento. Una espera más larga exige una señal explícita para no convertirse en incertidumbre. Estos umbrales no son leyes absolutas, pero ayudan a entender algo fundamental: el tiempo modifica la interpretación del evento.

Cuando la respuesta llega muy cerca de la acción, el usuario puede atribuirle a su propio gesto, y esa atribución sostiene la sensación de agencia. En una interfaz, el contacto no se comunica solo con un cambio visual, sino con un cambio suficientemente próximo a la acción. Por eso `contact.press` depende tanto de la latencia: si el sistema responde tarde, el evento deja de sentirse como contacto y empieza a sentirse como una reacción posterior del sistema.

La espera activa otro mecanismo. Cuando el usuario actúa y el resultado no llega, necesita saber si el sistema sigue trabajando; la ausencia de señal durante una espera puede interpretarse como bloqueo, fallo o pérdida de control. De ahí la función de `sustain`: no comunicar resultado, sino continuidad.

La sincronía multisensorial también depende del tiempo. Si un movimiento, un sonido y una vibración pertenecen al mismo evento, deben estar suficientemente coordinados para que el usuario los perciba como una unidad; si llegan desfasados, pueden sentirse como eventos separados. El tiempo, por tanto, no es solo duración: es la condición que permite percibir causalidad, continuidad, espera, resolución y unidad multicanal.

§ 3

Latencia

Latencia es el intervalo entre la acción del usuario y la respuesta perceptible del sistema. No es solo una métrica técnica: es una condición de agencia. Cuando el usuario pulsa un botón, arrastra un objeto, escribe una tecla o activa un control, necesita que el sistema responda lo bastante rápido como para que esa respuesta se atribuya a su acción. Si la respuesta llega dentro de una ventana breve, el usuario siente «lo he hecho yo». Si llega demasiado tarde, la lectura cambia:

LECTURA

«El sistema ha hecho algo después.»

Esa diferencia es semántica: separa la acción propia de la reacción ajena.

§ 4

Duración

Duración no es lo mismo que latencia. La latencia pregunta cuándo empieza la respuesta; la duración, cuánto dura la expresión del evento. Un evento demasiado corto puede no ser percibido; uno demasiado largo puede sentirse pesado, teatral o intrusivo. La duración debe corresponder a la función semántica del evento. `contact` debe ser breve; `sustain` puede ser prolongado; `signal` puede necesitar persistencia. `commit` necesita

resolución; emerge necesita entrada y salida; handle necesita continuidad durante el gesto.

§ 5

Espera

La espera es una de las situaciones temporales más delicadas. Cuando el usuario actúa y el sistema no puede dar un resultado inmediato, aparece una pregunta: ¿esto sigue vivo o se ha roto? Ahí nace *sustain*, que no comunica éxito ni error, sino continuidad operativa. Un spinner, una barra de progreso, un skeleton, un texto «guardando...» o una actualización periódica no son elementos decorativos. Son formas de decir:

LECTURA

«*El sistema sigue trabajando.*»

§ 6

Persistencia y huella

No todo evento termina cuando termina su animación. Una selección queda marcada; una alerta permanece hasta ser atendida; un resultado deja estado; una pérdida deja ausencia o undo; un proceso deja progreso. Conviene por eso distinguir duración expresiva, persistencia, huella y caducidad.

La distinción de fondo es la de dos superficies que coexisten: la *ocurrencia*, que vive su ventana perceptiva y se disipa, y el *estado*, que persiste como huella. «Esto acaba de pasar» y «esto es ahora verdad» son lecturas distintas, y colapsarlas en una sola superficie hace que la huella se disipe con la señal.

La persistencia es especialmente importante para la accesibilidad. Una señal efímera puede no ser percibida por todos los usuarios: si la información es importante, debe dejar alguna huella.

§ 7

Secuencia

El significado temporal depende también del orden. No es lo mismo `signal.warn + threat` seguido de `commit.delete + loss` —advertir y luego consumir— que la secuencia inversa. `commit.delete + loss` seguido de `signal.warn + threat` es consumir y advertir después, cuando ya no hay nada que evitar. El orden cambia el sentido.

§ 8

Tiempo e intent

El intent también modula el tiempo. `Affirm` suele ser breve, suave, de baja activación; `fulfill` puede durar algo más porque marca la resolución de una tensión. `risk` necesita persistir lo suficiente para que el usuario lo procese. `threat` debe entrar rápido y con alta saliencia; `loss` puede ser breve, grave y dejar huella.

La posición de un intent en el espacio valencia/activación tiene consecuencias temporales: un `threat`, por su alta activación, debe entrar rápido y con suficiente saliencia; un `loss` no exige la misma urgencia (Capítulo 10).

§ 9

Accesibilidad temporal

El tiempo también es accesibilidad. No todos los usuarios procesan los cambios a la misma velocidad; no todos ven una señal efímera; no todos pueden actuar antes de un `timeout`; no todos interpretan una animación rápida. La accesibilidad temporal incluye duración suficiente de los mensajes importantes, control sobre pausas y `timeouts`, evitar desapariciones críticas, no depender de instantes efímeros y mantener los estados importantes después del evento.

PRINCIPIO

Si un evento importante solo existe durante un instante, muchos usuarios no lo recibirán.

§ 10

Síntesis

El tiempo atraviesa todos los canales. Define si una respuesta se siente causada por la acción del usuario o como evento separado; si una espera parece proceso o bloqueo. Define si una señal se percibe o se pierde; si una culminación cierra una tensión o pasa desapercibida. La tesis:

PRINCIPIO

El tiempo no acompaña al evento. El tiempo forma parte de su significado.

13

Motion: la física percibida del evento

Movimiento, causalidad, continuidad y energía

PREGUNTA DEL CAPÍTULO

¿Qué dice el movimiento sobre la causa, la energía y la continuidad de lo que ocurre?

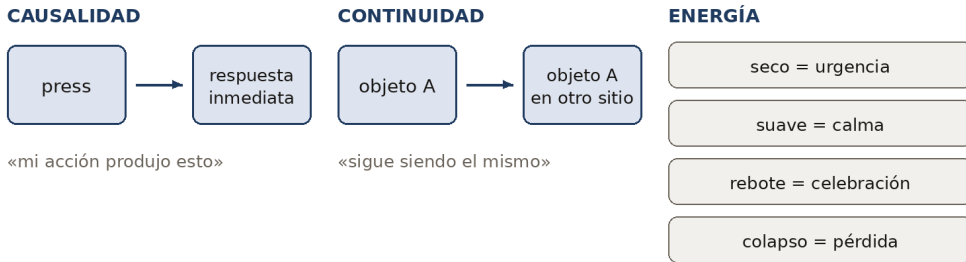
Una ventana que aparece de golpe y otra que se asienta con un leve rebote muestran el mismo contenido, pero no dicen lo mismo. La primera se materializa de la nada; la segunda parece llegar de algún sitio. Un panel que entra deslizándose desde la izquierda indica por dónde se vuelve; el mismo panel que simplemente aparece no deja esa pista. El movimiento, antes de adornar, informa.

Animación es la técnica; motion es la percepción del cambio en movimiento. Una animación puede ser una transición CSS, un transform, un keyframe, un spring, un fade, un slide, un scale, un spinner o una interpolación entre estados. Pero motion, en este libro, no es la propiedad técnica que cambia. Es lo que el usuario percibe cuando algo se desplaza,

se comprime, se eleva, se asienta, se resiste, tiembla, colapsa o entra en el campo. Motion comunica causalidad, continuidad, energía, masa, dirección y régimen.

FIGURA 13.1

Lo que el movimiento comunica



La misma técnica (una animación) puede decir causa, identidad o intensidad: eso es motion.

Motion no decora el cambio: dice cómo ocurre.

§ 1

Fundamento psicoperceptivo: movimiento, causalidad y continuidad

Motion se apoya en mecanismos básicos de percepción del movimiento. Uno de los más relevantes para interfaces es la percepción de causalidad. Los estudios clásicos de Michotte mostraron que ciertas secuencias simples de movimiento se perciben directamente como relaciones causales: un objeto se mueve hacia otro, lo toca, y el segundo empieza a moverse. El observador no percibe simplemente dos desplazamientos sucesivos, sino una relación de causa.

En interfaz ocurre algo parecido, aunque a otra escala. Cuando el usuario pulsa un botón y el botón se comprime inmediatamente, la respuesta se

interpreta como consecuencia del gesto. Si la respuesta llega tarde o con una forma no relacionada, esa atribución se debilita.

Motion también sostiene la continuidad de objeto. Cuando un elemento cambia de posición, se expande o se desplaza hacia otro lugar, el movimiento permite al usuario percibir que sigue siendo el mismo objeto en otra condición. Sin movimiento, el cambio puede sentirse como la desaparición de una cosa y la aparición de otra.

Otro fundamento importante es la predicción sensorimotora. Cuando algo empieza a moverse, el usuario genera expectativas sobre su trayectoria, velocidad y destino. Si el movimiento se corta, salta o contradice la física implícita que sugiere, aparece una sensación de extrañeza.

Finalmente, el movimiento comunica energía y masa mediante claves temporales: velocidad, aceleración, desaceleración, rebote, rigidez, fricción, overshoot o colapso. La interfaz no tiene peso real, pero motion puede producir una lectura de peso, ligereza, resistencia o pérdida de energía. Estos fundamentos no dictan una curva universal, pero sí justifican la tesis central:

DISTINCIÓN

Motion no es decoración; es una forma de comunicar cómo ocurre un cambio.

§ 2

Causalidad

La primera función de motion es preservar causalidad. Cuando el usuario pulsa un botón y el botón se comprime de inmediato, la interfaz comunica:

LECTURA

«Mi acción produjo una respuesta.»

Esa relación es la base de contact.

§ 3

Continuidad

Motion permite que el usuario perciba que algo sigue siendo lo mismo a través del cambio. Una tarjeta que se desplaza hasta otra posición sigue siendo la misma tarjeta; un panel que se expande desde su encabezado sigue perteneciendo a ese origen. Un indicador activo que se desliza de una pestaña a otra conserva la continuidad de la navegación; un objeto arrastrado mantiene identidad durante el gesto. Sin continuidad, el usuario debe reconstruir el cambio con memoria y razonamiento.

§ 4

Energía, masa y activación

Motion comunica energía. Un movimiento rápido y seco puede comunicar urgencia; uno suave, calma. Un rebote puede comunicar elasticidad o celebración; un colapso, pérdida de energía; un shake, discrepancia o alerta; un asentamiento, cierre.

La energía del movimiento puede reforzar la activación del evento. Un threat, situado en una región de alta activación negativa, puede admitir entradas más rápidas, tensas o salientes. Un risk necesita una tensión más moderada. Un affirm debe ser suave y breve; un fulfill puede tener más resolución. Un loss puede usar descenso, contracción o retirada grave, no la energía urgente de una amenaza activa.

§ 5

Dirección

Motion comunica de dónde viene algo y hacia dónde va. Un dropdown que emerge desde su trigger se entiende como dependiente de ese control. Un modal que aparece centrado comunica un marco dominante. Un panel que entra desde un borde comunica procedencia lateral. Una tarjeta que

colapsa hacia una zona de descarte comunica retirada o pérdida. La dirección no es decorativa: es orientación.

§ 6

Motion y familias

Cada familia usa el motion para lo suyo. contact, para la causalidad inmediata; commit, para el asentamiento y el cierre; signal, para llamar la atención; handle, para el control continuo. emerge, para la aparición y la retirada; shift, para el cambio de contexto; y sustain, para la continuidad.

§ 7

Motion y accesibilidad

Motion es poderoso, pero delicado. Puede ayudar a entender causalidad y continuidad; también puede producir mareo, distracción, fatiga o malestar vestibular. Reducir motion no debe significar eliminar semántica. Si motion transportaba contacto, debe migrar a estado visual, sonido opcional o háptica; si transportaba continuidad, a texto o estado. Si transportaba cambio de contexto, a foco, título, estructura y landmarks.

PRINCIPIO

Reducir motion no significa eliminar el tiempo ni el significado. Significa expresarlos sin movimiento problemático.

§ 8

Síntesis

Motion es la física percibida del evento. Su pregunta no es «cómo animar esto», sino «qué clase de cambio necesita percibir el usuario».

14

Presencia y profundidad

Lo que está, lo que aparece y lo que ocupa un plano

PREGUNTA DEL CAPÍTULO

¿Qué distingue estar, aparecer y ocupar un plano en el campo perceptivo de la interfaz?

A veces lo decisivo no es que algo se mueva, sino que algo aparece. Un tooltip entra en el campo. Un dropdown se abre bajo un botón. Un panel lateral se despliega. Un modal se impone sobre la página. Una tarjeta desaparece de una lista. Una barra de undo aparece después de eliminar. Un objeto arrastrado se eleva sobre el layout. En todos esos casos, la interfaz no solo muestra o esconde elementos: reorganiza la presencia y la profundidad del campo perceptivo.

La presencia responde a: ¿esto está aquí, aparece, permanece o se retira? La profundidad responde a: ¿en qué plano ocurre y qué prioridad tiene? Ambos canales son esenciales para entender `emerge`, `shift`, `handle`, `signal` y `commit.delete + loss`.

§ 1

Presencia no es existencia técnica

Un elemento puede existir técnicamente y no estar presente para el usuario: puede estar en el DOM pero oculto, fuera del viewport, colapsado, subordinado, inaccesible por foco; puede existir como estado interno pero no como experiencia. Y al revés: algo puede ser técnicamente transitorio y tener una presencia perceptiva muy fuerte, como un modal, una alerta o un toast.

Por eso, en este libro, presencia no significa existencia técnica: significa existencia perceptiva. La pregunta no es «¿está renderizado?», sino «¿forma parte del campo perceptivo del usuario?».

§ 2

Fundamento psicoperceptivo: figura/fondo, plano y claves de profundidad

Presencia y profundidad se apoyan en la forma en que la percepción organiza el campo visual. La psicología Gestalt mostró que no percibimos elementos aislados, sino relaciones: agrupación, continuidad, figura/fondo, cierre, proximidad, semejanza y destino común. Esto es fundamental para entender por qué un modal, un tooltip o un dropdown no son solo rectángulos que aparecen en pantalla: cada uno reorganiza el campo de una manera distinta.

La distinción figura/fondo es especialmente importante. Cuando un modal aparece sobre una página, el contenido anterior no desaparece necesariamente, pero pasa a funcionar como fondo subordinado; el modal se convierte en figura operativa.

La profundidad se construye mediante claves perceptivas como oclusión, sombra, escala relativa, desenfoque, superposición y tratamiento del fondo. La pantalla es plana, pero esas claves permiten leer relaciones: esto está delante, esto queda detrás, esto flota, esto depende de aquello,

esto domina el marco actual. Por eso `shift.enter-mode` no se comunica solo porque aparece un modal, sino porque cambia la relación de planos. En `handle.pick`, la profundidad comunica otro cambio: el objeto pasa del layout al régimen de manipulación. En `emerge`, la presencia permite leer aparición o retirada. En `loss`, la ausencia misma puede comunicar que algo dejó de estar disponible.

§ 3

emerge, shift y signal: tres formas de aparecer

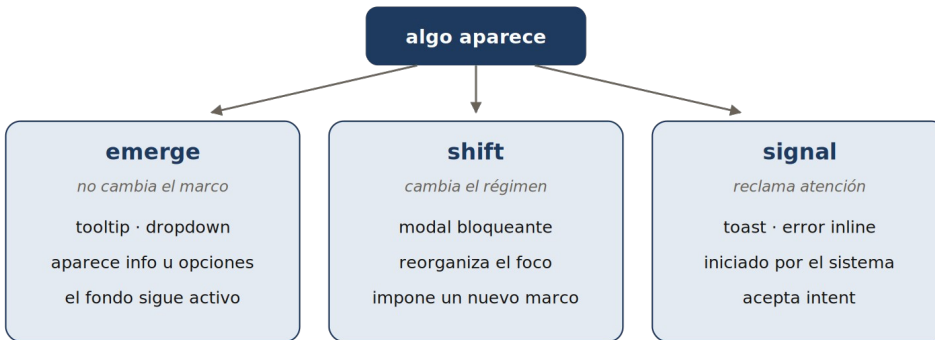
No todo lo que aparece pertenece a la misma familia. Un tooltip, un dropdown, un modal, un toast y un error inline aparecen todos, pero no son el mismo evento:

- **Tooltip** → `emerge.present`: aparece información auxiliar; no cambia el marco ni reclama atención urgente, y debe estar anclado al elemento que explica.
- **Dropdown** → `emerge.open`: aparecen opciones locales; la superficie depende de un trigger, el fondo sigue activo, no hay cambio de régimen operativo completo.
- **Modal** → `shift.enter-mode`: aparece una superficie, sí, pero su función principal no es aparecer, sino cambiar el contexto —subordina el fondo, captura el foco y exige una acción dentro de un nuevo régimen—.
- **Toast** → `signal.notify` + `neutral`: aparece un mensaje iniciado por el sistema; la presencia sirve a la señal.
- **Error inline** → `signal.warn` + `risk`: aparece junto a un campo, pero su función no es solo presencia; reclama atención sobre un problema corregible.

La pregunta clave es doble: ¿algo apareció, o el sistema está reclamando atención? ¿Algo apareció, o cambió el contexto?

FIGURA 14.1

Tres formas de aparecer



Aparecer no es un solo evento: depende de si cambia el marco o si reclama la atención.

§ 4

Profundidad no significa importancia

Un error frecuente es asumir que todo lo que está «más arriba» es más importante. No necesariamente: un dropdown puede flotar sin ser importante, un tooltip puede estar por encima sin ser urgente, un objeto arrastrado puede elevarse porque está bajo control —no porque sea crítico— y un modal puede estar delante porque cambia el marco, no porque su contenido sea peligroso.

La profundidad comunica plano, relación y régimen operativo; la importancia afectiva depende del evento evaluable. Por eso `shift.enter-mode` seguido de `signal.warn + threat` es más preciso que `shift.enter-mode + threat`: el modal cambia el plano; la advertencia porta la amenaza.

DISTINCIÓN

Más profundidad no significa más peligro. Significa otro plano operativo.

§ 5

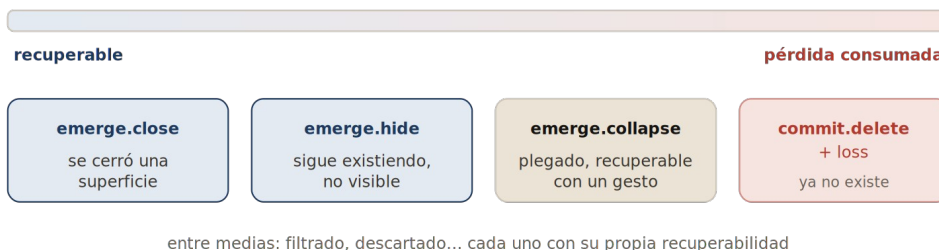
Desaparecer no significa perder

La desaparición visual es ambigua. Una tarjeta puede desaparecer porque se cerró, se ocultó, se filtró, se colapsó, se descartó, se eliminó o se perdió, y si la interfaz no distingue esas diferencias, el usuario debe adivinarlas. La gramática las separa: `emerge.close` dice «se cerró una superficie»; `emerge.hide` dice «sigue existiendo, pero no está visible»; `emerge.collapse` dice «el contenido volvió a estar contenido»; `commit.delete + loss` dice «dejó de estar disponible». La presencia debe hacer legible si algo se retira de la vista o si realmente deja de existir operativamente.

FIGURA 14.2

Desaparecer no significa perder

la misma desaparición visual puede significar cosas muy distintas



Si la interfaz no distingue estas desapariciones, el usuario tiene que adivinar si algo se perdió.

PRINCIPIO

La desaparición visual es ambigua. La semántica debe distinguir retirada, ocultación, colapso y pérdida.

§ 6

Ejemplo narrativo: tarjeta eliminada con undo

Un usuario elimina una tarjeta de una lista. La solución pobre: la tarjeta hace fade out, y el usuario puede no saber si se ocultó, se filtró, se movió o se eliminó. La solución semántica es una secuencia: `signal.warn` + `threat` si la acción es destructiva; al confirmar, `commit.delete` + `loss` —la tarjeta se retira como consecuencia—; y después, `emerge.present` con la barra de deshacer, cerrando con `signal.announce` + `neutral`.

La barra de undo no es la pérdida: la pérdida ya ocurrió en el `commit`. La barra emerge como posibilidad temporal de reparación. En canales: la presencia retira la tarjeta, el motion expresa la retirada, la forma trae la barra, el texto dice «Tarjeta eliminada», el tiempo mantiene el undo lo suficiente, el color refuerza con sobriedad y la accesibilidad anuncia si procede. La lectura final: «esto se eliminó, pero todavía puedes deshacerlo».

§ 7

Ejemplo narrativo: command palette

Una command palette aparece sobre la interfaz. Podría parecer un caso de emerge, porque «aparece algo», pero semánticamente es más fuerte: captura el foco, convierte el teclado en canal principal, permite ejecutar acciones globales, subordina el fondo e introduce un modo temporal. Cambia el régimen operativo. Por tanto es `shift.enter-mode`, no `emerge.open`: aunque la superficie aparezca, su función principal no es presencia local, sino cambio de contexto.

En canales: profundidad de superficie dominante, entrada clara, foco en el input, shell reconocible, motion rápido, y gestión accesible de foco, escape y retorno. La lectura: «ahora estás operando sobre el sistema entero desde otro modo».

§ 8

Accesibilidad de presencia y profundidad

Presencia y profundidad pueden ser claras visualmente y opacas para otros modos de interacción. Un modal visualmente evidente puede fallar si no gestiona el foco; un fondo oscurecido puede parecer subordinado pero seguir activo para el teclado; un toast puede aparecer sin ser anunciado; un dropdown puede flotar sin estar relacionado con su trigger; un cambio de vista puede ser visible sin mover el foco ni actualizar el título.

La accesibilidad exige coherencia entre el plano visual, el foco, el árbol accesible, el orden de lectura y el estado operativo. Si algo emerge visualmente, también debe emerger operativamente; si un fondo queda subordinado visualmente, también debe quedarlo en la interacción.

§ 9

Antipatrones

Profundidad decorativa. Sombras y elevaciones repartidas por estética, sin régimen operativo detrás. Si todo flota, ningún plano significa nada. La profundidad declara jerarquía de régimen: se usa cuando el régimen cambia.

Fondo vivo. El contenido queda visualmente subordinado —oscurecido, desenfocado— pero sigue aceptando interacción. La percepción dice «esto está apartado» y el sistema dice lo contrario. Subordinación visual y subordinación operativa van juntas.

El marco que grita. Cargar la amenaza en el plano elevado —el modal entero teñido, la capa que tiembla— en lugar de en el mensaje evaluable que contiene. El plano organiza; la señal evalúa.

§ 10

Síntesis

Presencia y profundidad comunican existencia perceptiva, plano y régimen. La presencia pregunta si esto está, aparece, permanece o se retira; la profundidad, en qué plano ocurre y qué prioridad tiene. Y las reglas centrales del capítulo caben en cuatro negaciones: aparecer no es advertir; aparecer no es cambiar de marco; flotar no significa ser importante; desaparecer no significa perder.

PRINCIPIO

La interfaz no solo muestra elementos. Hace entrar, salir, elevarse, subordinarse y desaparecer partes del contexto operativo.

TABLA 14.1 Superficies de aparición y familia dominante

SUPERFICIE	FAMILIA DOMINANTE	LECTURA
Tooltip	emerge.present	aparece ayuda auxiliar
Dropdown	emerge.open	aparecen opciones locales
Modal	shift.enter-mode	cambia el contexto
Toast	signal.notify	el sistema comunica algo
Error inline	signal.warn + risk	hay un problema corregible

TABLA 14.2 Antipatrones de presencia y profundidad

ANTIPATRÓN	QUÉ FALLA	CORRECCIÓN
Dropdown como modal	profundidad excesiva	mantener anclaje local
Modal como popover	marco insuficiente	subordinar fondo y foco
Todo lo que aparece parece alerta	exceso de saliencia	separar emerge de signal
Desaparición ambigua	no se sabe si se ocultó o perdió	distinguir hide / delete + loss
Fondo visualmente bloqueado pero activo	incoherencia	gestionar foco e interacción
Toast efímero importante	pérdida de información	persistencia / historial

15

Forma: límite, affordance e iconografía

Lo que se reconoce antes de colorearse, moverse o sonar

PREGUNTA DEL CAPÍTULO

¿Qué comunica la forma de un elemento antes de que el color, el motion o el sonido intervengan?

Antes de que una interfaz use color, motion, sonido o háptica, ya está diciendo algo mediante la forma. Un botón parece pulsable antes de estar coloreado. Un campo parece editable antes de mostrar validación. Un icono sugiere una acción antes de recibir un tooltip. Un borde separa una región antes de que haya profundidad. Una tarjeta agrupa contenido antes de que se eleve.

La forma es uno de los canales más discretos de la interfaz, pero también uno de los más persistentes: no aparece solo cuando ocurre un evento, sino que está ahí continuamente, estructurando lo que puede tocarse, leerse, separarse, agruparse o reconocerse. Responde a preguntas como: ¿esto es accionable?, ¿está separado?, ¿pertenece a este grupo?, ¿es

editable?, ¿está activo?, ¿está enfocado?, ¿representa una acción conocida?

§ 1

Fundamento psicoperceptivo: Gestalt, contorno y affordance

La forma se apoya en dos familias de fundamentos: la organización perceptiva y la percepción de acción posible.

Desde la organización perceptiva, la Gestalt mostró que el sistema perceptivo tiende a agrupar y separar elementos según proximidad, semejanza, continuidad, cierre, región común y figura/fondo. Un borde puede convertir varios elementos en una región común; una forma repetida puede indicar pertenencia; una discontinuidad puede marcar separación; un contorno cerrado puede hacer que algo se perciba como unidad.

Desde la teoría de affordances, especialmente en Gibson, la forma también sugiere posibilidades de acción. Una superficie con relieve, borde, tamaño táctil y posición estable puede sugerir presión; un asa visible, arrastre; un campo rectangular con cursor, escritura; un chevron, expansión; un icono de papelera, eliminación.

La iconografía añade una capa más: condensa acciones o estados en siluetas aprendidas. Pero los iconos no son universales —funcionan cuando el sistema cultural, el contexto y la redundancia los sostienen—. Por eso una X puede ser cerrar, cancelar, borrar o eliminar; un check, seleccionar, confirmar o completar; un triángulo, play, expansión o advertencia; un punto rojo, nuevo, grabando, error o alerta. La forma no es solo lo que algo parece: es parte de lo que el sistema ofrece hacer.

§ 2

Forma como límite

Una de las funciones principales de la forma es establecer límite. Un límite permite saber que esto es una unidad, que pertenece aquí y no allí, que puede manipularse como objeto, que es una región de acción. Sin límites, la interfaz se vuelve ambigua: un bloque de contenido sin borde, sombra, fondo o separación puede no parecer una tarjeta; un grupo de botones sin región común puede parecer una lista de acciones inconexas; un área de drop sin contorno puede no parecer destino; un campo sin forma puede no parecer editable; una alerta sin contenedor puede confundirse con texto ordinario.

El límite puede expresarse con borde, fondo, espacio, radio, sombra, contorno, región común o alineación. No siempre hace falta una línea: a veces el espacio basta, a veces el fondo basta. Pero cuando la estructura lo requiere, debe haber una señal de pertenencia y separación.

§ 3

Forma como affordance

Una forma puede invitar o bloquear acción. Un botón bien formado no necesita explicar demasiado que puede pulsarse; un slider bien formado no necesita decir que puede arrastrarse; un input bien formado no necesita tutorial para indicar escritura; un chip sugiere selección, filtro o etiqueta; un handle visible sugiere que algo puede moverse. La forma participa en las familias semánticas antes de que el evento ocurra:

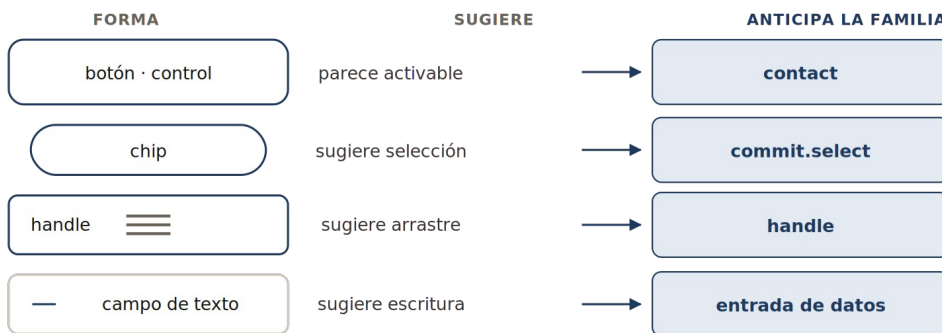
- **En contact**, un control debe parecer activable antes de recibir contacto —button, icon button, toolbar item, card clickable—. Si su forma no sugiere acción, el usuario primero debe adivinar que puede actuar.

- **En commit**, una opción seleccionable debe distinguirse del texto ordinario —checkbox, radio, switch, chip, toggle—: la forma ayuda a decir «esto puede quedar elegido».
- **En signal**, la señal necesita forma reconocible —banner, badge, callout, error inline, toast— para no confundirse con contenido ordinario: la forma marca que eso debe leerse como comunicación del sistema.
- **En handle**, la manipulación necesita affordances explícitas —drag handle, grip dots, drop zone, resize corner—; sin forma que sugiera agarre, la manipulación depende del descubrimiento accidental.
- **En emerge**, un trigger debe sugerir que puede abrir algo —chevron, caret, plus, disclosure—: la forma anticipa el evento de aparición.
- **En sustain**, el proceso necesita una forma que represente continuidad —barra, círculo, skeleton, track, línea de tiempo—: la forma hace visible el tipo de espera.

FIGURA 15.1

La forma como affordance

la forma sugiere qué se puede hacer antes de que el evento ocurra



La forma participa en la familia semántica antes de que ocurra el evento: prepara la lectura.

PRINCIPIO

La forma puede anticipar, sostener o fijar la lectura semántica de un evento.

§ 4

Forma y estado

La forma también comunica estado, y no solo mediante color: muchas veces el estado debe poder reconocerse por forma, contorno, icono, densidad, posición o marca —seleccionado, enfocado, desactivado, expandido, colapsado, inválido, editable, arrastrable, bloqueado—. Si el estado solo cambia de color, la semántica es frágil.

Una opción seleccionada puede necesitar check, borde, fondo, marca persistente o cambio de peso. Un campo inválido puede necesitar icono, mensaje, borde y relación espacial. Un accordion expandido puede necesitar cambio de chevron, contorno y relación con el contenido desplegado. La forma permite que el estado sobreviva cuando el color no basta o cuando el motion se reduce.

§ 5

Forma e intent

La forma no es intent, pero puede reforzarlo. Un risk puede usar forma de advertencia moderada: icono, callout, borde, contenedor local. Un threat, forma más contundente: bloque dominante, icono claro, agrupación crítica. Un loss, formas de ausencia o registro: espacio vacío, estado tachado, undo, recurso retirado. Un affirm, forma pequeña y estable: check discreto, marca de guardado, estado aplicado. Un fulfill, forma más resolutive: check grande, pantalla de completado, progreso al cien por cien. Esto importa porque los intents no deben depender solo del color.

§ 6

Ejemplo narrativo: campo inválido

Un usuario deja vacío un campo obligatorio: `signal.warn + risk`. La solución pobre —`border-color: red;`— cambia el color, pero hace que el evento

dependa casi por completo del canal cromático. La solución semántica combina borde o contorno, icono de advertencia, mensaje vinculado, forma de callout local, relación espacial clara, persistencia hasta la corrección, aria-describedby y, sobre todo, no solo rojo.

La forma ayuda a responder dónde está el problema, a qué campo afecta, si es corregible y qué hacer. El error no debería sentirse como amenaza global si solo es un riesgo local corregible.

§ 7

Ejemplo narrativo: selección de filtro

Un usuario selecciona un filtro en una lista de chips: `commit.select + affirm`. Si solo cambia el color del chip, puede entenderse, pero depende demasiado del canal cromático. La forma puede reforzar el estado con un check interno, un borde más definido, un fondo persistente, un cambio de peso, un botón de remove o la posición dentro de un grupo activo.

La pregunta perceptiva es «¿esto quedó elegido?», y la forma debe dejar huella: un contacto se siente y desaparece; una selección debe permanecer.

§ 8

Ejemplo narrativo: área de drop

Un usuario arrastra un archivo a una zona de subida. Antes de que el archivo entre, la zona debe sugerir que acepta drop: eso es forma como affordance. Los recursos son conocidos: contorno discontinuo, icono de upload, texto breve, tamaño suficiente, fondo suave, cambio de forma al hover, estado válido o inválido. La secuencia completa: `handle.carry`, la affordance de destino, `handle.drop` y `commit.upload + affirm` o `fulfill`.

Si la zona no tiene forma suficiente, el usuario no sabe dónde soltar; si solo se colorea al pasar por encima, quizá llega tarde. La forma debe anticipar la posibilidad de acción.

§ 9

Accesibilidad de la forma

La forma es crucial para la accesibilidad: si todo depende del color, la forma debe sostener el significado; si una animación se reduce, la forma puede dejar estado persistente; si el sonido no está disponible, la forma puede reforzar la señal; si el usuario navega con teclado, el foco debe tener forma clara.

Los problemas frecuentes son reconocibles: foco invisible, iconos sin texto, áreas clicables sin forma clara, contornos demasiado sutiles, targets demasiado pequeños, drop zones inaccesibles por teclado, botones que parecen texto y texto que parece botón. La accesibilidad de la forma no es solo contraste: es reconocimiento de función.

§ 10

Síntesis

La forma es el canal del límite, la affordance y el reconocimiento estructural: comunica dónde empieza y termina algo, qué pertenece junto, qué puede hacerse, qué estado tiene y qué acción representa.

PRINCIPIO

La forma no dice solo cómo se ve algo. Dice qué es, dónde termina y qué permite hacer.

16

Color: refuerzo semántico y valencia

El canal que más hemos confundido con la semántica

PREGUNTA DEL CAPÍTULO

¿Qué puede decir el color de un evento, y qué le hemos pedido decir que no le corresponde?

Para casi uno de cada doce hombres, el rojo de un error y el verde de un acierto se parecen mucho más de lo que el diseñador supone. El daltonismo rojo-verde afecta en torno al 8 % de los hombres — y a una de cada doscientas mujeres—. Su forma más común, la deuteranomalía, a menudo ni se nota hasta que un test la revela. Para todas esas personas, una interfaz que separa «correcto» de «incorrecto» solo por el color no separa nada. Y aun para quien distingue todos los colores, el rojo no significa lo mismo en todas partes: peligro en una cultura, suerte o celebración en otra.

El color es uno de los canales más poderosos de la interfaz, y también uno de los más peligrosos. Durante décadas, buena parte de la semántica

visual se ha delegado en él: rojo para error, amarillo para advertencia, verde para éxito, azul para información. Este capítulo intenta devolverlo a su sitio: el color no es la semántica; es un canal que puede reforzarla.

FIGURA 16.1

El color acompaña; no sostiene

FRÁGIL — solo color

Con deuteranomalía ($\approx 8\%$ de los hombres), este borde no es un campo de correo electrónico» es obligatorio. Icono, mensaje y vínculo con el campo sostienen la lectura.

ROBUSTO — color + forma + icono + texto

saliencia

estado

valencia

identidad

Lo que el color hace bien: orientar, marcar, modular, dar marca — acompañando a otros canales.

El mismo error, sin y con respaldo. El color expresa el intent; no lo define.

§ 1

Fundamento psicoperceptivo: saliencia, contraste y codificación

El color se apoya en varios mecanismos perceptivos, y conviene distinguirlos porque no fallan igual. El primero es la saliencia: un color que contrasta con su entorno puede capturar la atención antes de que el usuario lea nada. Un punto rojo sobre un campo gris se encuentra sin buscarlo. Ese poder de captura es anterior a cualquier significado, y por eso mismo debe administrarse: una pantalla donde todo es saliente es una pantalla donde nada lo es.

El segundo es el contraste. El color no se percibe de forma aislada, sino en relación con el fondo, la luminancia, la saturación y el contexto. El mismo

verde que funciona sobre blanco puede desaparecer sobre gris claro, y un amarillo de advertencia puede volverse ilegible en modo oscuro. El color de una interfaz no es un valor hexadecimal, sino una relación.

El tercero es la codificación aprendida. Rojo como peligro, verde como positivo, amarillo como precaución y azul como información funcionan porque los usuarios los han visto muchas veces, no porque formen por sí solos una teoría del afecto. Son convenciones potentes —sería absurdo no aprovecharlas— pero convenciones al fin: dependen de la cultura, del dominio y del contexto de uso.

El cuarto es la variabilidad perceptiva. No todos los usuarios distinguen los mismos colores, ni en las mismas condiciones. La deuteranomalía del arranque del capítulo es solo el caso más frecuente; hay pantallas mal calibradas, luz solar directa, modos de alto contraste, filtros nocturnos. Por todo ello el color es fuerte, pero incompleto: rápido para orientar, frágil para sostener él solo un significado.

§ 2

Color e intent

El color no define el intent. El intent proviene de la evaluación del evento en el espacio valencia/activación; el color es solo uno de los canales que puede reforzarlo. Rojo, amarillo, verde o azul no son la semántica: son recursos expresivos que pueden ayudar a comunicar amenaza, riesgo, confirmación, resolución, pérdida o neutralidad.

La diferencia entre threat y loss no nace de dos tonos de rojo: nace de la distinción evaluativa que el Capítulo 10 argumentó — lo evitable frente a lo consumado. Si ambos comparten el mismo rojo saturado, el usuario aprende que el rojo «grita» siempre igual, y deja de distinguir cuándo puede actuar y cuándo solo puede aceptar. La separación correcta no pide dos pigmentos, sino dos expresiones. El threat admite un rojo saliente con

presencia dominante; el loss pide gravedad contenida —un rojo desaturado, una ausencia, una huella— sin la energía de la alarma.

La diferencia entre affirm y fulfill tampoco nace de dos verdes. Nace de la diferencia entre confirmación suave y resolución positiva. Un autoguardado no debería estrenar el mismo verde festivo que la culminación de un proceso largo: si todo lo positivo brilla igual, nada culmina. El color puede reforzar esa gradación —un verde discreto para affirm, uno más pleno para fulfill—, pero no debe sustituirla.

§ 3

La herencia del semáforo

El Capítulo 10 desarrolló la crítica al semáforo; aquí solo aplicamos su consecuencia al canal color. La convención heredada funciona como atajo cromático, pero no como teoría semántica completa: danger mezcla threat y loss; success mezcla affirm y fulfill; info mezcla saliencia con evaluación. Mientras el color fue el único vehículo del intent, esas mezclas parecían inevitables. En una gramática multicanal dejan de serlo: el color queda subordinado al evento, no al revés.

§ 4

Lo que el color hace bien

Criticar la sobrecarga del color no es negarle funciones: bien empleado, es insustituible. Orienta la atención más rápido que casi cualquier otro canal visual. Marca estado con una economía envidiable: un campo activo, una pestaña seleccionada, un elemento deshabilitado. Agrupa por categoría cuando hay que distinguir series, etiquetas o capas. Modula la valencia de un evento evaluable reforzando lo que la forma y el texto ya dicen. Y construye identidad: la paleta de un producto es también memoria de marca.

La regla operativa que reúne todo esto: el color acompaña de maravilla y sostiene mal. Puede ser el primero en llegar a la atención, pero no debería ser el único que quede cuando el usuario necesita entender qué ocurrió.

§ 5

Accesibilidad cromática

El color no debe ser el único medio para comunicar información crítica. La lista de fragilidades habituales es conocida: el error señalado solo con un borde rojo; el enlace distinguido solo por el color del texto; la selección marcada solo con un fondo verde. O la gráfica cuyas series solo se separan por tono; la validación que responde solo con verde o rojo. Cada uno de esos casos funciona en el escritorio del diseñador y falla en el mundo: con deuteranomalía, con poco contraste, con luz solar, con una impresión en blanco y negro.

La corrección no es renunciar al color, sino acompañarlo. El error suma icono, mensaje y vínculo con el campo; el enlace suma subrayado; la selección suma marca de verificación y borde. La gráfica suma patrones, etiquetas o formas; la validación suma texto y estado persistente. El color sigue ahí, haciendo su trabajo de refuerzo; pero el significado ya no depende de él.

§ 6

Antipatrones

Color como significado único. El intent vive solo en el pigmento: sin él, el evento enmudece. Corrección: forma, icono, texto y posición cargan el significado; el color refuerza.

Un solo rojo para todo lo negativo. threat y loss comparten expresión, y el usuario no distingue lo evitable de lo consumado. Corrección: separar energía y gravedad.

Un solo verde para todo lo positivo. affirm y fulfill brillan igual, y la culminación pierde valor. Corrección: graduar la intensidad según la resolución.

Azul informativo decorativo. Todo lo que no es error ni éxito recibe azul «info», mezcle o no saliencia con evaluación. Corrección: preguntar primero si el evento es evaluable; si solo necesita ser notado, es cuestión de saliencia, no de intent.

§ 7

Síntesis

El color orienta atención, refuerza estado, modula valencia y construye identidad visual. Pero cuando trabaja solo, la semántica queda expuesta: basta una condición perceptiva distinta para que el evento desaparezca.

PRINCIPIO

El color expresa el intent. No lo define.

17

Sonido: urgencia, valencia y microeventos auditivos

Cuando la interfaz también se oye

PREGUNTA DEL CAPÍTULO

¿Qué añade el sonido a la lectura de un evento, y cuándo sobra?

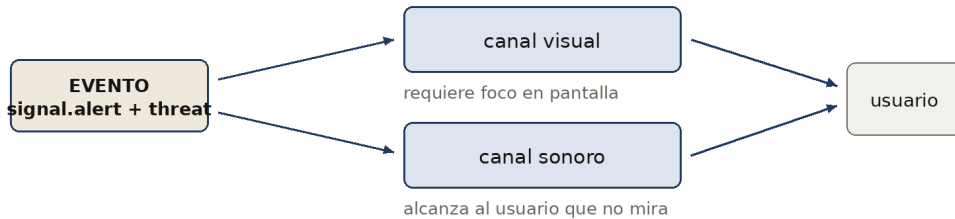
La web ha tratado el sonido con desconfianza durante mucho tiempo, y no sin razón. Durante años, el sonido en interfaces web estuvo asociado a malas prácticas: música automática, anuncios intrusivos, notificaciones insistentes, páginas que sonaban sin permiso. Efectos gratuitos y experiencias imposibles de usar en oficinas, bibliotecas, transporte público o espacios compartidos. La consecuencia cultural fue clara: mejor que la interfaz no suene.

Esa reacción era comprensible, pero produjo una simplificación peligrosa: confundir el mal uso del sonido con la inutilidad del sonido. El problema histórico no ha sido el sonido en sí, sino el sonido sin gramática. Un sonido arbitrario molesta; uno demasiado largo invade; uno inesperado sobresalta; uno repetido fatiga; uno sin relación clara con el evento parece

decoración o ruido. Pero un sonido breve, proporcionado, sincronizado con el evento y opcional para el usuario puede comunicar cosas que otros canales comunican peor.

FIGURA 17.1

Por qué el sonido: alcanza fuera del foco



Precisamente por ese poder: sonido solo para lo raro y relevante, nunca como canal único.

El procesamiento auditivo es el más rápido para la valencia; su coste es la intrusión.

§ 1

Fundamento psicoperceptivo: correspondencias, roughness e integración audiovisual

El sonido se apoya en varios fenómenos perceptivos relevantes para la interfaz.

El primero son las correspondencias crossmodales. Charles Spence ha revisado un amplio cuerpo de investigaciones que muestra que las personas tienden a asociar ciertos rasgos auditivos con rasgos visuales o espaciales. Pitch alto con objetos más pequeños, brillantes o elevados; pitch bajo con objetos más grandes, oscuros o pesados. Estas correspondencias no deben leerse como reglas universales cerradas, pero

sí como orientaciones útiles para diseñar congruencia entre canales. El tono agudo sugiere ligereza y puntualidad; el grave, peso y cierre. El ascenso tonal, apertura y logro; el descenso, cierre, caída o pérdida. El timbre limpio, claridad y baja tensión; el áspero, urgencia y alarma.

El segundo fundamento es la roughness o aspereza acústica: ciertos timbres ásperos o modulados tienden a aumentar la saliencia y la sensación de alarma. Esto no significa que toda alerta deba sonar como alarma; significa lo contrario: la aspereza acústica tiene tanta carga atencional que debe reservarse para eventos donde esa carga esté justificada.

El tercero es la integración audiovisual. Cuando un sonido ocurre en sincronía con un movimiento, una aparición o una vibración, el usuario puede percibirlos como un único evento. Si llega tarde, demasiado pronto o con una cualidad incongruente, se separa de la experiencia y se convierte en ruido añadido.

El cuarto conecta con el sistema de intents: las variaciones en pitch, timbre, ataque y duración pueden modular la activación y la valencia percibida del evento. Un sonido no es «amenazante» por una frecuencia aislada: lo es por la relación entre pitch, timbre, ataque, duración, contexto, volumen, repetición y evento.

§ 2

Qué comunica el sonido

El sonido puede comunicar contacto, cierre, urgencia, valencia, presencia fuera de foco, materialidad, pérdida y resolución. Su fuerza está en que no exige que el usuario mire el lugar donde ocurre el evento. Una señal visual puede pasar desapercibida si la atención está en otra zona; un sonido breve puede alcanzar al usuario igualmente. Eso no significa que deba usarse para todo: significa que, cuando un evento necesita atravesar el campo atencional, el sonido puede ser un canal muy eficaz.

§ 3

Sonido y familias semánticas

- **contact** — Un microclick breve puede reforzar que el sistema recibió la acción, pero no todos los contactos deben sonar: en interfaces de alta frecuencia, sonorizar cada press se vuelve insoportable. El sonido de contact debe ser breve, limpio, opcional y fácil de desactivar.
- **commit** — El sonido puede marcar cierre o resultado: affirm casi imperceptible, fulfill más resolutivo, loss más grave o descendente.
- **signal** — Es donde el sonido es más potente, porque alcanza al usuario fuera del foco visual; y también donde más riesgo hay de invadir. Una notificación informativa no debería sonar como alarma; una amenaza crítica sí puede justificar un sonido más saliente.
- **handle** — Puede reforzar pick y drop, no tanto carry: el carry es una fase de control continuo, y sonorizar el arrastre completo fatiga.
- **emerge** — Normalmente no necesita sonido: un tooltip, dropdown o popover no debería sonar salvo que contenga una señal evaluable.
- **shift** — El cambio de contexto puede tener sonido en casos específicos, pero normalmente profundidad, foco y motion bastan.
- **sustain** — El sonido sostenido es peligroso: un proceso en curso no debería emitir un loop audible salvo en contextos muy particulares. El sonido puede marcar inicio o fin, pero rara vez debe acompañar toda la espera.

§ 4

Sonido e intent

El sonido puede modular el intent con mucha precisión. neutral pide silencio o sonido mínimo; affirm, algo breve, limpio y suave; fulfill, un perfil ascendente y resolutivo. risk, un sonido seco, moderado y contenido; threat, saliencia, aspereza y ataque claro; loss, gravedad, descenso y

sequedad. La diferencia entre threat y loss es crucial: un sonido de amenaza prepara para actuar; un sonido de pérdida registra que algo ya ocurrió. No deben sonar igual.

La misma disciplina vale para nombrar los sonidos del sistema: por su materia —breve, ascendente, áspero, grave—, no por su veredicto. Un contorno descendente no «es» una pérdida: tiende a leerse como cierre o descenso, y por eso sirve a loss; bautizarlo «sonido de error» lo encadena a una sola evaluación.

Ese vocabulario de materia tiene un punto de partida concreto: la paleta sonora inicial del Apéndice B — contornos, registros, asperezas y ganancias orientativas por intent, con números de una implementación real.

§ 5

Ejemplo narrativo: autoguardado

Un editor guarda cambios automáticamente: `commit.save + affirm`. ¿Debe sonar? Depende. Si el autoguardado ocurre cada pocos segundos, probablemente no: sería fatigante. Si ocurre al final de una edición importante, puede haber una confirmación discreta. La solución semántica: texto «guardado», marca visual breve, color positivo discreto, sin sonido por defecto y sonido opcional si el usuario lo activa. El intent es `affirm`, no `fulfill`: no hay tensión fuerte que celebrar, solo normalidad confirmada.

§ 6

Ejemplo narrativo: subida completada

Un usuario sube un archivo grande: `contact.press`, `sustain.progress`, `commit.complete + fulfill`. Aquí el sonido puede tener sentido: durante `sustain`, silencio; al completar, un sonido breve y resolutivo puede marcar

el cierre. El sonido no comunica que el proceso estaba vivo —eso lo hacía el progreso visual—: comunica que la tensión se resolvió.

§ 7

Ejemplo narrativo: alerta crítica

Una alerta crítica —signal.alert + threat— puede justificar sonido saliente, porque necesita atravesar el campo atencional. Pero debe tener límites: no sonar indefinidamente, no ser exageradamente estridente, permitir silencio, tener equivalente visual persistente y no repetirse sin control. La alerta sonora debería decir «atiende ahora», no «entra en pánico».

§ 8

Ejemplo narrativo: eliminación

El usuario elimina una tarjeta. Antes de confirmar —signal.warn + threat— puede haber sonido si el evento es grave y requiere atención. Después de confirmar —commit.delete + loss—, si hay sonido, debería ser distinto: el threat puede tener más ataque y saliencia; el loss, más sequedad, gravedad o descenso. El sonido de pérdida no debe sonar como alarma activa, porque ya no hay nada que evitar: debe registrar consecuencia.

PRINCIPIO

El sonido de amenaza prepara para actuar. El sonido de pérdida registra que algo ya ocurrió.

§ 9

Sonido y web

En la web, el sonido necesita una doble justificación: perceptiva y social. Técnicamente puede producirse audio con Web Audio API, pero hay restricciones reales. Políticas de autoplay, necesidad de gesto del usuario, variación entre navegadores, dispositivos silenciados, entornos

compartidos y un rechazo cultural al audio web que viene de lejos. Por eso en la web el sonido debe ser desactivable, estar bajo control del usuario y no ser crítico como única vía. Debe ser breve, estar vinculado a eventos claros, iniciarse probablemente tras una interacción y respetar el contexto.

§ 10

Accesibilidad sonora

El sonido nunca debe ser el único canal de información crítica. Los usuarios sordos o con hipoacusia pueden no percibirlo, y quienes tienen el sonido desactivado no lo recibirán. En entornos ruidosos puede perderse, y las personas con sensibilidad auditiva pueden rechazarlo. Por tanto, una alerta sonora debe tener equivalente visual; una confirmación sonora, estado visible; una pérdida sonora, huella; una notificación sonora, texto; y el usuario debe poder reducir o desactivar el sonido.

§ 11

Antipatrones

Sonido para todo. Marcar con audio cada contacto y cada notificación trivial. El canal se fatiga y deja de significar: cuando suene algo importante, ya nadie escucha. El sonido se reserva para consecuencia y urgencia real.

Una sola campana. El mismo tono para notify, warn y alert. El oído no puede distinguir lo que la interfaz no distingue. La gravedad necesita diferenciación clara: en timbre, duración o intensidad.

Significado solo audible. Información que existe únicamente como sonido, sin equivalente visible ni anunciante. Quien no oye, pierde el evento entero. El sonido refuerza; nunca es el único portador.

§ 12

Síntesis

El sonido es uno de los canales más peligrosos y más potentes: comunica contacto, cierre, urgencia, valencia, pérdida, materialidad y eventos fuera del foco visual.

PRINCIPIO

La prueba de un sonido es triple: ¿ayuda a reconocer qué ocurrió, cuándo ocurrió o cómo debe evaluarse? Cuando no supera ninguna de las tres, sobra.

TABLA 17.1 Antipatrones del sonido

ANTIPIATRÓN	PROBLEMA	CORRECCIÓN
sonido decorativo	ruido	vincular a evento
sonido largo	invasión	microsonidos
alerta solo sonora	inaccesibilidad	respaldo visual
mismo sonido para todo	indiferenciación	familias + intents
alarma menor	fatiga	separar risk / threat
loop de carga	molestia	silencio durante sustain
sin control	rechazo	preferencias

18

Háptica y vibración

Contacto, impacto, resistencia y encaje

PREGUNTA DEL CAPÍTULO

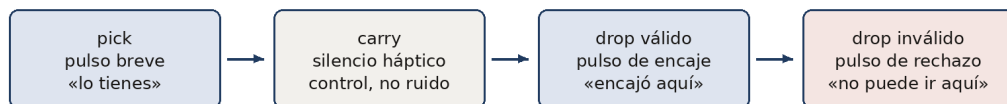
¿Qué puede confirmar el cuerpo que la vista solo puede suponer?

La interfaz no solo puede verse u oírse: también puede sentirse, al menos en algunos dispositivos. Un teléfono puede responder con un pulso breve al tocar una tecla; un reloj puede vibrar para avisar de una notificación. Un mando puede transmitir impacto, fricción o tensión; un trackpad puede simular un click que no existe físicamente.

La háptica es el canal de expresión que comunica mediante sensación corporal: vibración, presión, resistencia, impacto, textura, pulso o ritmo táctil. En la web este canal todavía es limitado, pero esa limitación técnica no elimina el canal como dimensión teórica. La gramática debe distinguir la háptica como canal de expresión de la háptica como capacidad disponible en una plataforma concreta.

FIGURA 18.1

Háptica: pulsos en los bordes del gesto



El cuerpo confirma lo que la vista solo puede suponer: contacto, encaje, rechazo.

Refuerzo corporal en los momentos de decisión; silencio durante el control continuo.

§ 1

Fundamento psicoperceptivo: tacto, cuerpo y acción

La háptica se apoya en un hecho básico: la interacción con el mundo no es solo visual. Tocamos, presionamos, agarramos, soltamos, sentimos resistencia, textura, peso, impacto y vibración. La percepción táctil y propioceptiva participa en cómo entendemos la acción. Cuando una interfaz simula contacto no está inventando una categoría nueva: está apoyándose en una experiencia corporal primaria —tocar algo y recibir una respuesta—. Por eso un pequeño pulso puede reforzar `contact.press`: el usuario no solo ve que el sistema respondió; lo siente.

La háptica también conecta con los esquemas sensoriomotores de la cognición corporeizada —contacto, impacto, resistencia, encaje, pérdida— que tienen base corporal y pueden trasladarse, de forma limitada, a la interfaz. Y la investigación sobre *tactons* ha mostrado que los parámetros hápticos pueden codificar información mediante duración, ritmo, intensidad, localización o frecuencia. Un pulso breve no comunica lo

mismo que una vibración larga, ni un doble pulso lo mismo que uno único, ni un patrón rítmico lo mismo que un impacto seco. Esto permite pensar la háptica como canal semántico, no solo como buzz genérico.

La háptica se integra con motion y sonido. Cuando el usuario suelta un objeto y ve que encaja, un pulso sincronizado puede reforzar la sensación de drop. Si llega tarde, se separa del evento; si es demasiado fuerte, comunica gravedad donde quizá solo había encaje leve.

§ 2

Qué comunica la háptica

La háptica puede comunicar contacto, impacto, encaje, rechazo, resistencia, alerta corporal, pérdida y ritmo. Mientras el motion puede simular la física visual de un evento, la háptica refuerza su dimensión corporal. Un botón que se comprime comunica contacto visual, y un pulso breve, contacto táctil. Un objeto que se asienta comunica encaje visual, y un pequeño pulso al soltar, encaje corporal. Una alerta visual comunica urgencia, y una vibración fuerte o repetida, urgencia corporal.

§ 3

Háptica y familias semánticas

- **contact** — Es la familia háptica más básica: un pulso breve dice «el sistema ha sentido mi acción». Pero no todos los contactos deben vibrar: en interfaces de alta frecuencia, la háptica cansa rápidamente.
- **commit** — Puede reforzar selección, cierre, rechazo o pérdida: un pulso suave marca selección; uno más definido, final de operación; un patrón de rechazo, fallo; una vibración seca, pérdida.
- **signal** — Puede reclamar atención corporal, pero debe reservarse para señales importantes: una vibración por cada notificación menor convierte el cuerpo del usuario en superficie de ruido.

- **handle** — Puede reforzar agarre, encaje, rechazo o destino en pick y drop; el carry debería permanecer sin vibración continua, salvo simulaciones especializadas de fricción.
- **emerge** — Normalmente no la necesita: un tooltip, dropdown o popover no debería vibrar al aparecer.
- **shift** — Puede usarla en cambios de modo muy significativos, con moderación.
- **sustain** — La háptica sostenida es muy delicada: en general, no debería usarse para procesos continuos.

§ 4

Háptica e intent

La háptica puede modular el intent, aunque con menos sutileza que el color, el motion o el sonido en muchas plataformas. neutral pide ausencia o mínimo; affirm, un pulso suave; fulfill, uno más definido o un doble suave. risk, un pulso contenido; threat, una vibración fuerte o doble; loss, un pulso seco y pesado. La diferencia entre threat y loss vuelve a ser importante: la háptica de amenaza puede ser más activadora; la de pérdida, más seca y conclusiva.

De aquí sale una regla de nomenclatura para el vocabulario del canal. Las primitivas hápticas se nombran por cómo se sienten —un tick, un golpe seco, un zumbido áspero—, no por el veredicto que suelen acompañar. Llamar «error» a un patrón de vibración reimporta el semáforo en la capa del canal y acopla la primitiva a un solo intent, ocultando que la misma sensación puede servir a evaluaciones distintas. La primitiva es materia; el veredicto es lectura.

Y también aquí hay punto de partida: la paleta háptica inicial del Apéndice B, con la firma orientativa de cada intent.

§ 5

La web y sus límites hápticos

En la web, la háptica sigue siendo un canal débil. La Vibration API permite patrones básicos en algunos dispositivos, pero no ofrece control fino sobre textura, intensidad, localización o calidad del motor. Su soporte varía entre navegadores y plataformas, y muchos contextos de escritorio no tienen háptica disponible. Esto significa que en la web no podemos asumirla como canal principal. La gramática puede incluirla, pero la implementación debe tratarla como opcional, degradable, nunca única, dependiente de dispositivo, sujeta a permisos y controlable por el usuario.

PRINCIPIO

En web, la háptica debe ser enriquecimiento, no soporte semántico obligatorio.

§ 6

Ejemplo narrativo: teclado móvil

Un teclado móvil puede usar háptica para reforzar cada tecla: `contact.press`. Pero el evento es de frecuencia altísima —el usuario puede pulsar cientos de teclas en pocos minutos—. Así que el pulso debe ser muy breve, de intensidad baja, desactivable, sin variación dramática y sin intent fuerte. Un teclado que vibra demasiado convierte el contacto en fatiga. Aquí la semántica correcta es casi invisible: estoy tocando, el sistema responde, puedo seguir.

§ 7

Ejemplo narrativo: selección de chip

El usuario selecciona un filtro: `commit.select + affirm`. Una háptica suave puede reforzar que la selección quedó aplicada, especialmente en móvil, pero no debe sentirse como logro importante: `affirm`, no `fulfill`. La

expresión posible: pulso único suave, marca visual persistente, color positivo leve, sin sonido por defecto. La háptica aquí refuerza estado, no celebra.

§ 8

Ejemplo narrativo: drag-and-drop táctil

En una interfaz táctil, el usuario arrastra una tarjeta: `handle.pick`, `handle.carry`, `handle.drop`. En el `pick`, un pulso breve puede comunicar agarre —«lo has cogido»—. Durante el `carry`, silencio háptico: el usuario necesita control, no ruido táctil continuo. En el `drop`, la háptica se bifurca: pulso de encaje si es válido —«encajó aquí»—, pulso de rechazo si no —«no puede ir aquí»—. Y si el `drop` es destructivo —`handle.drop` seguido de `commit.delete` + `loss`—, la háptica puede comunicar consecuencia, pero con cuidado: no debe parecer amenaza si la pérdida ya ocurrió.

§ 9

Ejemplo narrativo: alerta en wearable

En un reloj, la háptica puede ser el canal principal de una señal: `signal.alert` + `threat`. Aquí la vibración tiene sentido: la pantalla es pequeña, el usuario puede no estar mirando, el dispositivo está en contacto con el cuerpo, la señal puede requerir atención inmediata. Pero incluso aquí debe haber control: patrón reconocible, duración limitada, posibilidad de silenciar, prioridad bien definida y no usar el mismo patrón para todo. La háptica de wearable puede ser muy efectiva, y precisamente por eso debe ser sobria.

§ 10

Accesibilidad háptica

La háptica puede ayudar a algunas personas y molestar a otras. Puede servir como canal alternativo cuando el sonido está desactivado, reforzar

el contacto o ayudar en wearables y contextos donde la visión está ocupada. Pero también puede ser problemática: sensibilidad táctil, vibración molesta o dolorosa, fatiga por repetición, motores demasiado bruscos, falta de control de intensidad, patrones indistinguibles o interferencia con ayudas de accesibilidad.

Por eso la háptica debe ser desactivable, proporcional, infrecuente cuando es intensa, nunca única para información crítica, ajustable si la plataforma lo permite y coherente con el evento. La accesibilidad háptica no consiste en añadir vibración para «mejorar accesibilidad»: a veces la vibración la reduce.

§ 11

Antipatrones

Vibración ambiental. Háptica en cada toque, cada desplazamiento, cada transición. El cuerpo se habitúa y el canal queda gastado para cuando importe. La háptica marca momentos; no acompaña superficies.

Un solo pulso. La misma vibración para confirmar, advertir y fallar. Sin diferenciación, la háptica solo dice «algo pasó». Los patrones se distinguen por significado, no solo por intensidad.

Castigo háptico. Vibración sostenida o repetida como alarma. Incomoda sin informar, y penaliza justo cuando el usuario necesita pensar. Pulsos breves y diferenciados; la urgencia la sostienen otros canales.

§ 12

Síntesis

La háptica es el canal corporal de la interfaz: comunica contacto, impacto, encaje, rechazo, resistencia, alerta y pérdida mediante sensación táctil. Su fuerza está en que hace sentir el evento en el cuerpo; su riesgo, precisamente ahí: puede invadir, fatigar o excluir si se usa mal.

DISTINCIÓN

La háptica no es vibración añadida. Es la posibilidad de hacer corporalmente perceptible un evento.

PRINCIPIO

Diseñar la semántica para que pueda vivir sin háptica, y enriquecerla con háptica cuando la plataforma y el usuario lo permitan.

TABLA 18.1 Antipatrones de la háptica

ANTIPATRÓN	PROBLEMA	CORRECCIÓN
vibrar por todo	fatiga corporal	reservar para eventos relevantes
mismo buzz para todo	indiferenciación	variar duración, ritmo, intensidad
háptica crítica sin respaldo	inaccesibilidad	canales alternativos
háptica desincronizada	ruido	sincronía con press/drop/señal
intensa y frecuente	rechazo	reducir intensidad
asumir háptica rica en web	inconsistencia	tratar como opcional

19

Accesibilidad perceptiva y reducción de canales

Cuando no todos perciben el mismo evento

PREGUNTA DEL CAPÍTULO

¿Qué queda del significado de un evento cuando uno de sus canales desaparece?

Una gramática multicanal tiene una obligación inmediata: no puede asumir que todos los usuarios perciben los mismos canales, con la misma intensidad, en las mismas condiciones y con la misma tolerancia. La accesibilidad no debería aparecer al final como cumplimiento. En este libro ocupa un lugar más profundo:

PRINCIPIO

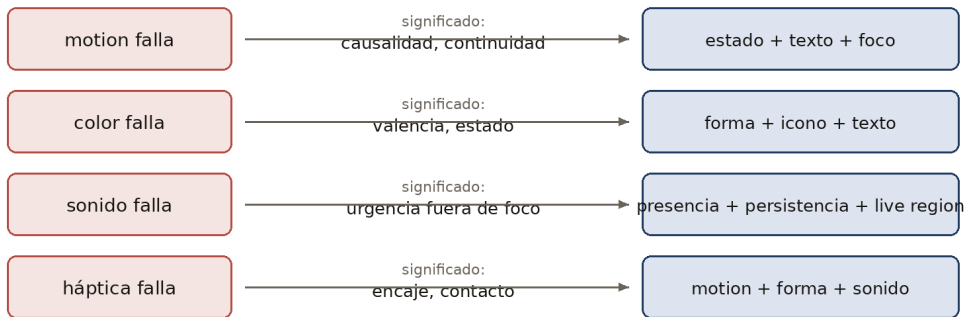
La accesibilidad es la prueba de estrés de una semántica perceptiva.

Este capítulo establece el marco conceptual de esa prueba: qué significa que un canal falte, qué se pierde exactamente cuando falta y cómo se conserva el significado. Su desarrollo práctico llegará en el Capítulo 33,

que aplicará esta lógica a flujos completos; aquí construimos las herramientas que ese capítulo usará.

FIGURA 19.1

Migración semántica: el significado cambia de cuerpo



No se copia el estímulo: se traduce la función.

Cuando un canal falla, migra el significado, no el adorno.

§ 1

Fundamento psicoperceptivo: variabilidad perceptiva y redistribución del significado

La accesibilidad perceptiva se apoya en una idea simple: no todos los usuarios disponen de los mismos canales en las mismas condiciones. La visión puede estar reducida; el color puede no discriminarse; el sonido puede estar desactivado; el movimiento puede producir malestar. La háptica puede no existir; la atención puede estar fragmentada; el tiempo de procesamiento puede variar. Y estas condiciones no son excepcionales ni siempre permanentes. El usuario con auriculares quitados, el móvil bajo el sol, la mano ocupada, el cansancio de última hora son reducciones de canal tan reales como cualquier diagnóstico. Una semántica perceptiva, por tanto, no puede depender de una única vía sensorial.

Los fundamentos del Capítulo 3 ayudan a entender por qué:

- **la Gestalt explica que la organización visual puede fallar si figura, fondo o agrupación no son claras;**
- **las affordances explican que una acción puede no percibirse como disponible si su forma no la sugiere;**
- **la atención explica que una señal puede no alcanzar el foco si no tiene suficiente saliencia;**
- **la valoración del evento explica que una evaluación puede perderse si el evento no comunica riesgo, pérdida o resolución;**
- **el espacio valencia/activación explica que el intent puede perderse si solo vive en color;**
- **la multisensorialidad explica que un evento puede sobrevivir si su significado migra entre canales;**
- **la cognición corporeizada explica que algunos usuarios pueden necesitar otras vías para percibir contacto, encaje o resistencia.**

La pregunta central no es solo si el usuario puede acceder al componente. Es si puede comprender qué evento ocurrió y qué implica. Un botón alcanzable por teclado cuyo resultado nadie anuncia es accesible en el sentido técnico e ilegible en el sentido semántico.

§ 2

Canal único y fragilidad

Si el significado crítico vive en un solo canal, la semántica es vulnerable. Los casos se reconocen enseguida: el error que solo es rojo, el contacto que solo es motion, la alerta que solo es sonido. La pérdida que solo es desaparición, el progreso que solo es spinner, la selección que solo es color. Ninguno de esos diseños es incorrecto por lo que usa, sino por lo que no respalda: cada uno apuesta todo el significado a una sola carta perceptiva. Y basta que esa carta falte —por condición, por contexto o por configuración— para que el evento deje de existir para el usuario.

La fragilidad, además, no se reparte por igual entre familias e intents. Un contact que falla produce un doble click; un signal.alert + threat que falla puede producir una pérdida real. Cuanto más grave es la consecuencia del evento, menos aceptable es que dependa de un canal único.

§ 3

Migración semántica

Cuando un canal se reduce, no basta con quitarlo. Hay que preguntar qué significado transportaba, y darle otro cuerpo. Si motion transportaba causalidad, debe migrar a estado, texto, sonido o háptica. Si el color transportaba valencia, debe migrar a forma, icono, texto o posición. Si el sonido transportaba urgencia, debe migrar a presencia, persistencia o live region. Si la háptica transportaba encaje, debe migrar a motion, forma o sonido.

Nótese que la migración no copia el estímulo: traduce la función. El objetivo no es que la interfaz «se vea parecida» sin el canal, sino que el usuario siga leyendo el mismo evento con la misma familia y el mismo intent. Esa es la diferencia entre adaptar un efecto y conservar una semántica, y es la operación que el Capítulo 33 practicará caso por caso.

§ 4

Accesibilidad y familias

Cada familia tiene su condición de supervivencia. contact debe sobrevivir sin motion: estado pressed, foco visible, respuesta textual si hace falta. commit debe dejar huella: un resultado que solo existió un instante no existió para todos. signal no puede depender solo de color o de sonido: necesita forma, texto y persistencia. handle necesita alternativas a drag: mover también es un evento de teclado. emerge debe sincronizar presencia visual y presencia operativa: lo que aparece en pantalla debe aparecer también para las tecnologías de asistencia. shift debe gestionar

foco y orientación: cruzar un marco sin anunciarlo deja al usuario en territorio desconocido. sustain necesita texto, estado y control: la continuidad no puede ser solo un giro.

§ 5

Síntesis

La accesibilidad perceptiva no es una excepción al sistema: es la prueba de que familias, intents y canales no dependen de una condición ideal de percepción. Una interacción semántica es robusta cuando conserva su significado bajo condiciones variables, y ese criterio —no la lista de atributos cumplidos— es el que este libro adopta como medida.

20

Coordinación multicanal

Cuando varios canales dicen un solo evento

PREGUNTA DEL CAPÍTULO

¿Qué hace falta para que varios canales cuenten un solo evento y no varios?

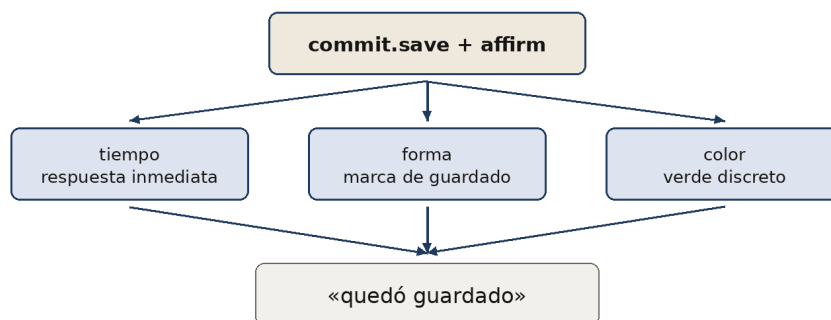
Una interfaz semántica no consiste en activar muchos canales, sino en que, cuando varios participan, digan lo mismo. O, más exactamente:

PRINCIPIO

Todos los canales activos deben participar en la misma lectura del evento.

FIGURA 20.1

Un evento, varios canales, una sola lectura



Congruencia temporal, semántica, afectiva, de intensidad y accesible: cinco planos, una historia.

La coordinación convierte la materia sensible en experiencia unificada.

§ 1

Fundamento psicoperceptivo: integración multisensorial y congruencia

La coordinación multicanal se apoya en un principio general de la percepción: el sistema perceptivo tiende a integrar señales de distintas modalidades cuando parecen pertenecer al mismo evento. Esa integración depende de la proximidad temporal, la correspondencia espacial, la congruencia semántica, la intensidad compatible, la plausibilidad causal, las expectativas previas y la coherencia afectiva. Si un movimiento, un sonido y una vibración ocurren en el momento adecuado y parecen corresponder al mismo tipo de evento, el usuario puede percibirlos como una unidad; si llegan desfasados, con intensidades contradictorias o con significados afectivos distintos, se separan.

La idea de integración temporal explica por qué un sonido de contacto que llega tarde se percibe como añadido, no como parte del press. La

correspondencia crossmodal explica por qué un objeto que se mueve como algo pesado pero suena como algo ligero produce conflicto. Y la teoría de valencia y activación explica por qué los canales deben modular el mismo intent: un mensaje de error con animación festiva no es una excentricidad estilística, es una contradicción evaluativa.

§ 2

La secuencia completa del sistema

La coordinación no empieza en los canales: empieza en el evento. La secuencia completa es evento → familia semántica → verbo / fase → evaluación si procede → intent → canales coordinados. Cuando esa secuencia se rompe, aparecen los errores característicos: canales sin evento, intent sin familia, color sin evaluación, sonido sin necesidad, motion sin semántica, accesibilidad como parche posterior.

Un ejemplo positivo condensa la secuencia. El usuario guarda un documento en un servidor lento: el botón se comprime al instante (contact, canal tiempo y micro motion), aparece una barra con el texto «Guardando...» (sustain, canales presencia, forma y texto), y al terminar la barra se resuelve en una marca de verificación con un verde discreto (commit.save + affirm, canales forma, color y persistencia breve). Tres eventos, siete canales, una sola historia. El mismo flujo descoordinado se reconoce también de inmediato: el botón no responde, la barra aparece tarde, la marca de éxito llega con un rebote triunfal y un sonido de fanfarria por un autoguardado rutinario.

§ 3

Congruencia, redundancia y sobrecarga

La coordinación exige congruencia en cinco planos. Temporal: las señales del mismo evento entran juntas o en un orden que respete su relación causal. Semántica: todos los canales cuentan la misma clase de evento —

si la forma dice advertencia, el motion no puede decir celebración—. Afectiva: todos modulan el mismo intent, sin que un canal grite lo que otro susurra. De intensidad: la energía de cada canal corresponde a la consecuencia del evento, no a las ganas de lucirlo. Y accesible: la lectura sobrevive cuando cualquiera de los canales desaparece.

La coordinación pide además redundancia, pero de un tipo preciso: la redundancia significativa no es repetir por repetir, sino hacer que el significado sobreviva cuando un canal falla. El icono y el texto del error no «duplican» el borde rojo; lo respaldan.

Su exceso también tiene nombre. Una interfaz puede activar demasiados canales: cada click vibra, cada confirmación suena, cada cambio rebota, cada error interrumpe, cada success celebra. El resultado no es claridad: es fatiga. La alternativa a la fragilidad no es la saturación, sino la asignación: cada canal activo con una función, y ningún canal activo sin ella.

§ 4

Síntesis

La Parte II ha separado los canales para poder entenderlos, pero el usuario no los vive separados: la interfaz debe volver a unirlos.

DISTINCIÓN

Una interacción semántica multicanal no es una colección de efectos. Es una composición perceptiva orientada a un único evento.

Cierre de la Parte II

Con este capítulo se cierra la Parte II. Hemos separado los canales para poder entenderlos —tiempo, motion, presencia, profundidad, forma, color, sonido, háptica, la accesibilidad y su coordinación—, pero el usuario no vive esos canales como capas separadas: los vive como un único evento, o

como ruido si la coordinación falla. La tesis de esta parte puede formularse así:

PRINCIPIO

El evento es la unidad de significado. Los canales son la materia sensible. La coordinación convierte esa materia en experiencia perceptiva unificada.

La siguiente parte vuelve al eje semántico del sistema: las familias del evento. Allí veremos cómo contact, commit, signal, handle, emerge, shift, sustain y delegate no son categorías abstractas, sino formas recurrentes de organizar acción, consecuencia, atención, manipulación, presencia, marco, continuidad y reparto de iniciativa.



Familias

Families

Las ocho familias semánticas, una a una: qué nombran, cómo se expresan, dónde están sus fronteras y cómo fallan.

21

Cómo leer los capítulos de familia

La plantilla común de las ocho familias y cómo usarla

PREGUNTA DEL CAPÍTULO

¿Cómo están contruidos los ocho capítulos de familia que siguen, y cómo conviene leerlos?

La Parte II terminó con los canales aprendiendo a decir una sola cosa. La Parte III cambia de plano: deja de construir la gramática general y empieza a recorrer, una por una, las ocho familias que organizan la semántica del evento. Son contact, commit, signal, handle, emerge, shift, sustain y delegate; cada una tiene su capítulo. Este, más breve, explica cómo están contruidos esos capítulos y cómo conviene usarlos.

El Capítulo 8 ya presentó las familias y su reducción desde el campo exploratorio, y el 9 fijó cuáles aceptan evaluación. No hace falta repetir nada de eso; basta re-ancilar la definición que todo lo que sigue da por sabida.

PRINCIPIO

Una familia semántica es una clase de evento perceptivamente recurrente que responde a una misma pregunta del usuario.

§ 1

La ficha de apertura

Los capítulos de contact a sustain abren con una ficha visual (Figuras 22.1 a 28.1) que condensa la familia en tres líneas. La primera es la *pregunta perceptiva* que responde. La segunda, la *secuencia canónica* en la que suele aparecer, escrita en la notación del libro. La tercera, la *frontera clave*: la confusión más frecuente con una familia vecina. La ficha no sustituye al capítulo: es su mapa. Si meses después de leer la Parte III una familia se desdibuja, releer su ficha debería bastar para recuperarla.

delegate, la octava, cierra la parte con la misma plantilla y una particularidad: su frontera —el agente que interpreta— es también una frontera declarada del libro.

§ 2

La plantilla común

Los ocho capítulos comparten una estructura estable. Primero, *qué es y qué no es* la familia: la definición positiva y las confusiones que descarta. Después, su *fundamento psicoperceptivo*: qué estructura de la experiencia —agencia, cierre, atención, control corporal, figura y fondo, cambio de marco, continuidad, autoría— la sostiene. Luego, sus *verbos y canales*: cómo se dice y por dónde se expresa. Y al final, la parte operativa: las *fronteras* con las familias vecinas, la *accesibilidad* y las migraciones cuando falta un canal, los *antipatrones* con su corrección y una *síntesis* con preguntas de validación.

Las tablas de cada capítulo son las de su familia: canales y su función, migraciones, antipatrones, preguntas de revisión. Las tablas transversales

—la síntesis de las ocho familias y los seis intents— no se repiten en cada capítulo: viven una sola vez, como referencia canónica, en el Apéndice B.

§ 3

Qué no son las familias

No son componentes. Un modal no es una familia: puede participar en `shift.enter-mode` o en `signal.alert + threat`. Un dropdown no es una familia: puede ser `emerge.open`. Un botón no es `contact`: puede participar en `contact.press`, pero también en `commit`, `signal` o `shift` según el evento.

No son efectos. Un fade no es `emerge`, un slide no es `shift`, un bounce no es `fulfill`, un shake no es `threat`. Los efectos son realizaciones posibles en canales; la familia es la clase de evento.

No son estados. «Loading», «error» y «success» no son familias: son mezclas de familia, verbo e intent —loading es `sustain.progress`; error es `commit.fail + risk`; success es `commit.complete + fulfill`—.

Estas tres distinciones vuelven una y otra vez en los capítulos que siguen, cada vez encarnadas en la familia concreta. El modal no es familia sino participante; el fade no es `emerge` sino realización; el «loading» no es estado primitivo sino `sustain.progress`.

§ 4

Dos maneras de leer la Parte III

La Parte III admite lectura lineal y lectura de consulta. Lineal: las familias están ordenadas del contacto al resultado, de la atención a la manipulación, de la presencia al proceso, y cierran con la autoría; leerlas seguidas construye el sistema completo. De consulta: cada capítulo es autónomo —ficha, plantilla, síntesis— y puede leerse solo, cuando el trabajo lo pida, sin más requisito que los capítulos 8 y 9.

Para la consulta rápida sin releer capítulos quedan la referencia canónica del Apéndice B y el glosario.

§ 5

Ejemplo transversal

Tomemos una acción simple: eliminar un elemento. La secuencia `contact.press` → `signal.warn` + `threat` → `commit.delete` + `loss` recorre tres familias distintas —acción registrada, advertencia, consecuencia— que responden a tres preguntas distintas: ¿he pulsado?, ¿puedo evitarlo?, ¿qué ocurrió? Si todo se tratara como un solo estado `danger`, se perderían esas distinciones.

Ese es el uso real de la Parte III: no memorizar ocho definiciones, sino reconocer, en cualquier secuencia concreta, qué familia responde a qué pregunta — y detectar cuál falta.

§ 6

Síntesis

Este capítulo no ha añadido gramática: ha entregado el mapa de los ocho que siguen. La definición queda re-anclada, la plantilla explicada, las fronteras señaladas y la referencia localizada.

PRINCIPIO

La interfaz no se organiza solo en componentes. Se organiza en familias de eventos que responden a preguntas perceptivas recurrentes.

22

Contact

La respuesta mínima del sistema a la acción del usuario

PREGUNTA DEL CAPÍTULO

¿Cómo sabe el usuario que el sistema ha sentido su acción?

Pulsar un botón y, durante una fracción de segundo, nada. ¿Ha entrado la pulsación? ¿Ha fallado? Antes de terminar de pensarlo, el dedo ya vuelve a pulsar. Esa duda mínima —no saber si el sistema ha sentido la acción— dura milésimas, pero gobierna la confianza con la que usamos cualquier interfaz.

Antes de que una interfaz confirme, advierta, complete, falle o pierda algo, ocurre un momento previo: el usuario actúa. Pulsa, toca, hace click, arrastra, escribe, activa. Y en ese instante aparece una pregunta inmediata: ¿el sistema ha sentido mi acción? Ese momento es contact.

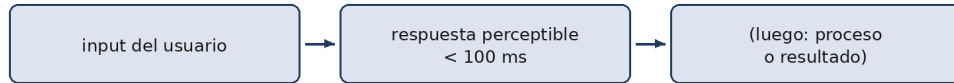
FIGURA 22.1

contact: la ficha de la familia

PREGUNTA PERCEPTIVA

¿El sistema ha sentido mi acción?

SECUENCIA CANÓNICA



FRONTERA CLAVE

contact.press — «te he sentido»

≠

commit.save — «se guardó»

Recibir la acción no es resolverla: contact inicia, commit resuelve.

La respuesta mínima que sostiene la agencia.

§ 1

Qué es contact

contact es la familia semántica de la recepción de la acción. No comunica resultado, ni éxito, ni error, ni estado final; comunica algo más básico y anterior a todo eso.

PRINCIPIO

La acción del usuario ha entrado en el sistema.

Sus verbos recorren las formas de esa recepción: *press*, *tap*, *release*, *activate*, *focus*, *trigger*. Un botón que se hunde ligeramente, una tecla virtual que responde al toque, un switch que muestra su estado pressed, un elemento que empieza a estar bajo el dedo: todo eso pertenece a contact.

§ 2

Qué no es contact

La forma más rápida de delimitar la familia es mirarla desde sus fronteras, porque cada una deja fuera una confusión frecuente:

- **contact no es commit.** `contact.press` $\neq$ `commit.save`: el sistema puede haber recibido la acción sin haber producido todavía resultado.
- **contact no es signal.** `contact.press` $\neq$ `signal.warn`: no reclama atención ni advierte; solo registra acción.
- **contact no es handle.** `contact.press` $\neq$ `handle.pick`: puede iniciar una manipulación, pero no es la manipulación.
- **contact no porta intent fuerte.** `contact.press` + `neutral` suele bastar, porque `contact` es anterior a la evaluación: cuando ocurre, todavía no hay nada que juzgar.

§ 3

Fundamento psicoperceptivo: agencia, causalidad inmediata y esquema táctil

`contact` se apoya en la percepción de agencia: la tendencia del usuario a atribuir un cambio al propio gesto cuando la respuesta ocurre con suficiente proximidad temporal y formal. Cuando el sistema responde de inmediato, la lectura es directa —yo hice esto, el sistema respondió—. Si la respuesta se retrasa, esa relación causal se debilita: yo hice esto... y, en algún momento, algo ocurrió.

Este fundamento conecta con la percepción de causalidad dinámica: ciertos cambios próximos en el tiempo y coherentes en su forma tienden a organizarse como relación causa-efecto. En la interfaz, la microrespuesta de un control preserva esa lectura causal.

También se apoya en esquemas sensorimotrices básicos. Tocar algo y recibir una respuesta es una experiencia corporal primaria —presión,

resistencia, rebote, click, vibración—, y la interfaz digital toma prestada esa estructura para hacer que una acción abstracta se sienta registrada. Contact no es decoración: es la semántica mínima de agencia.

§ 4

La pregunta perceptiva

contact responde a: ¿el sistema ha sentido mi acción? Si la respuesta es clara, el usuario continúa. Si no lo es, aparecen los comportamientos compensatorios que cualquiera reconoce: el doble click, la repetición del toque, la espera insegura, la frustración, el abandono de la operación. Muchos problemas de experiencia de uso no son errores funcionales; son fallos de contact.

§ 5

contact y latencia

El canal crítico de contact es el tiempo. Un contacto correcto encadena input y respuesta inmediata; un contacto fallido intercala un silencio — input, silencio, respuesta— y ese silencio no es un problema estético, sino semántico: durante ese hueco, el usuario no sabe si ha actuado.

PRINCIPIO

Un contacto bonito pero tardío es peor que un contacto simple pero inmediato.

§ 6

Canales y verbos de contact

contact suele apoyarse en pocos canales, y con poco le basta. El tiempo hace el trabajo principal; el microcambio de forma o de movimiento lo hace visible; el color puede acompañar, pero no debe ser el canal

principal, porque el contacto se siente sobre todo por tiempo, microcambio y relación causal.

Sus verbos —press, tap, activate, focus, trigger— no son gestos distintos que exijan semánticas distintas: `contact.press`, `contact.focus` o `contact.trigger` mantienen la misma lógica, acción registrada, no resultado.

§ 7

contact frente a commit

El par de contraste clave del capítulo es `contact.press` frente a `commit.save` + `affirm`: `contact` dice «he pulsado»; `commit` dice «se guardó». El error típico es `contact.press` + `fulfill`, que hace que el sistema diga «ya terminó» cuando solo ha empezado.

PRINCIPIO

contact inicia. commit resuelve.

§ 8

contact frente a signal

`contact.press` y `signal.warn` + `risk` hablan idiomas distintos: el primero dice «he actuado»; el segundo, «atiende esto». No deben mezclarse. Un contacto no debería reclamar atención si todavía no hay nada que evaluar.

§ 9

contact en componentes web

Visto sobre componentes concretos, el reparto queda claro:

- **Botón** → `contact.press`: microcompresión, estado pressed, retorno rápido.
- **Input** → `contact.focus`: cursor visible, contorno, foco.

- **Toggle** → `contact.press` seguido de `commit.toggle` + `affirm`: el `press` registra; el `toggle` fija.
- **Link** → `contact.press` seguido de `shift.navigate`: la presión no es la navegación.
- **Drag** → `contact.press` seguido de `handle.pick`: el contacto inicia el agarre, pero no lo agota.

§ 10

contact e intent

Lo normal es `contact` + `neutral`, o incluso sin `intent` explícito. En acciones graves, la interfaz puede anticipar el peso mediante forma, color o una señal previa, pero el `intent` fuerte no debería vivir en el contacto, sino en la señal o en la consecuencia posterior. La secuencia correcta de un borrado lo muestra: `contact.press`, luego `signal.warn` + `threat`, luego `commit.delete` + `loss`. Cada peso en su evento.

§ 11

contact y accesibilidad

`contact` debe sobrevivir sin `motion`, sin sonido y sin háptica. Si el usuario duda de si ha actuado, el problema no es solo de rendimiento: es de semántica de contacto.

§ 12

Antipatronos

Contacto mudo. El gesto no produce acuse inmediato y el usuario repite la acción: doble click, doble envío, doble pago. Todo contacto se registra de forma perceptible antes de 100 ms, aunque el resultado tarde.

Celebrar el contacto. Animación de éxito al pulsar, antes de que exista resultado. Se promete una consecuencia que aún no ocurrió. El contacto dice «te he sentido»; la celebración, si llega, pertenece al commit.

Acuse desproporcionado. Todo el componente —o la pantalla entera— reacciona a un toque puntual. El acuse pierde localidad y el usuario pierde el hilo entre gesto y respuesta. El acuse ocurre donde ocurrió el gesto.

§ 13

Síntesis

contact es la familia de la agencia mínima. No evalúa, no resuelve, no advierte: responde a una única pregunta —¿el sistema ha sentido mi acción?— y la percepción de agencia que sostiene depende de la proximidad entre acción y respuesta.

PRINCIPIO

Cuanto más frecuente es una acción, más inmediato y más discreto debe ser su contacto.

TABLA 22.1 Latencia y lectura del contacto

Latencia	Lectura
muy baja	contacto
media	respuesta
alta	duda / desconexión

TABLA 22.2 Canales de contact y su función

Canal	Función
Tiempo	respuesta inmediata
Motion	microrespuesta
Forma	estado pressed / foco
Sonido	click breve opcional

Háptica	pulso breve opcional
Color	secundario

TABLA 22.3 Migraciones de contact cuando falta un canal

Canal ausente	Sustituto
Motion	estado pressed, foco visible
Sonido	estado visual
Háptica	tiempo + forma
Visión	anuncio o respuesta auditiva si procede

TABLA 22.4 Antipatronos de contact

Antipatrón	Problema
Contacto tardío	rompe agencia
Contacto exagerado	teatraliza acciones menores
Contacto como éxito	promete resultado antes de tiempo
Sin respuesta	incertidumbre
Solo color	respuesta débil o inaccesible
Sonido en todo	fatiga

23

Commit

Cuando algo queda fijado, seleccionado, completado o perdido

PREGUNTA DEL CAPÍTULO

¿Cómo sabe el usuario qué resultado quedó después de actuar?

Pulsas «Guardar» y la pantalla no cambia. ¿Se ha guardado? Cierras, vuelves a abrir, compruebas. O al revés: pulsas «Eliminar», la fila desaparece al instante, y un segundo después te preguntas si era la que querías —y si hay forma de recuperarla—. Entre «lo he pedido» y «ya está hecho» hay un salto que el usuario necesita ver cruzado.

Si contact responde a «¿el sistema ha sentido mi acción?», commit responde a una pregunta distinta y decisiva: ¿qué ha pasado como consecuencia de esa acción? Aquí es donde la interfaz deja de registrar gestos y empieza a producir realidad operativa. Algo queda seleccionado, algo se guarda, algo se envía, se completa, falla, se descarta, se elimina o se pierde. Ese momento —cuando el sistema fija un estado o produce una consecuencia— es commit.

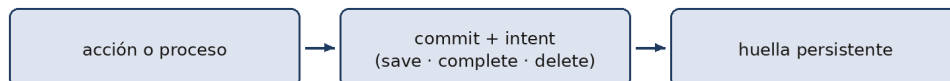
FIGURA 23.1

commit: la ficha de la familia

PREGUNTA PERCEPTIVA

¿Qué resultado quedó después de actuar?

SECUENCIA CANÓNICA



FRONTERA CLAVE

commit — resuelve y deja estado

≠

contact — solo registra el gesto

El resultado necesita huella; el contacto puede desvanecerse.

Lo que queda fijado debe poder verse después.

§ 1

Qué es commit

commit es la familia semántica de la fijación de estado o consecuencia. No es el inicio de la operación, ni su proceso, ni la advertencia que la precede: es el momento en que el sistema dice «esto ya ocurrió». Sus verbos recorren el repertorio de esa fijación: *select*, *save*, *submit*, *complete*, *partial*, *fail*, *block*, *delete*, *restore*, *cancel*, *expire*, *discard*. Y todos comparten una propiedad.

DISTINCIÓN

Después de un commit, el sistema ya no es exactamente el mismo que antes.

§ 2

Qué no es commit

También commit se entiende mejor desde sus fronteras: cada una separa la familia de una confusión frecuente:

- **commit no es contact.** `contact.press` $\neq$ `commit.save`: el sistema puede haber recibido la acción sin haber producido todavía el resultado.
- **commit no es sustain.** `sustain.progress` $\neq$ `commit.complete`: mientras el proceso sigue en curso, nada ha quedado fijado.
- **commit no es signal.** `signal.warn` $\neq$ `commit.fail`: una advertencia habla de lo que puede pasar; un resultado, de lo que ya pasó.
- **commit no es shift.** `shift.navigate` $\neq$ `commit.submit`: cambiar de pantalla no equivale a completar una operación.

§ 3

Fundamento psicoperceptivo: cierre, consecuencia y memoria de estado

commit se apoya en una necesidad fundamental de la acción: saber cuándo una operación produjo consecuencia. El usuario no solo necesita actuar; necesita saber qué quedó después de actuar. Tres dimensiones psicoperceptivas sostienen esa necesidad.

La primera es el cierre cognitivo. Una acción abierta genera incertidumbre, y el usuario necesita percibir cuándo puede dejar de vigilarla: commit cierra la secuencia acción $\rightarrow$ resultado. Sin cierre, la acción queda suspendida.

La segunda es el resultado de la acción: saber si algo cambió en el sistema —una opción quedó marcada, un documento fue guardado, un archivo se eliminó, una operación falló—.

La tercera es la memoria de estado. Un buen commit deja huella: no desaparece como un contacto, sino que puede quedar como estado

seleccionado, mensaje, ausencia, historial, deshacer, marca de completado o error persistente. Por eso commit no solo ocurre: se recuerda.

§ 4

La pregunta perceptiva

commit responde a: ¿qué ha quedado fijado? La pregunta adopta la forma de cada caso concreto —¿se guardó?, ¿se eligió?, ¿se envió?, ¿se completó?, ¿falló?, ¿se perdió?—, pero siempre apunta a lo mismo: la consecuencia que quedó en el sistema.

§ 5

Verbos y tipos de commit

Los verbos de commit definen el tipo de consecuencia, y se agrupan en tres familias de lectura:

- **Fijación de estado** — `commit.select`, `commit.toggle`, `commit.save`: algo queda aplicado o guardado.
- **Resolución de proceso** — `commit.complete`, `commit.submit`, `commit.fail`: un proceso terminó, bien o mal.
- **Consecuencia irreversible o crítica** — `commit.delete`, `commit.discard`, `commit.expire`: algo deja de estar disponible.

A ellos se suman `restore`, `cancel` y `reset`, que devuelven o anulan estado con la misma lógica de fijación.

§ 6

commit, huella e intent

A diferencia de `contact`, `commit` suele requerir huella; y es, además, una de las familias donde el `intent` vive con más fuerza, porque casi toda consecuencia se evalúa:

- `commit.save` + **affirm** → «todo va bien».
- `commit.complete` + **fulfill** → «objetivo logrado».
- `commit.fail` + **risk** → «algo no se completó: revisa».
- `commit.delete` + **loss** → «esto ya se perdió».

El `intent` no cambia el evento; cambia cómo se evalúa su consecuencia.

La huella tiene una consecuencia arquitectónica que conviene nombrar. Todo evento tiene dos superficies perceptivas que coexisten: la *ocurrencia*, que vive durante la ventana perceptiva del evento y luego se disipa, y el *estado*, que persiste como su huella. La *ocurrencia* dice «esto acaba de pasar»; el *estado* dice «esto es ahora verdad». Un aviso ocurriendo y un campo siendo inválido se leen distinto, y no deberían compartir superficie: si la comparten, retirar la señal obliga a borrar el estado, o conservar el estado congela la señal.

Una gramática robusta las mantiene separadas: los canales transitorios se suscriben a la *ocurrencia*; la huella vive en el *estado*. Así el `commit` puede dejar rastro duradero mientras su señal se apaga.

PRINCIPIO

La ocurrencia dice «esto acaba de pasar». El estado dice «esto es ahora verdad». Colapsarlas hace que la huella se disipe con la señal.

§ 7

Las fronteras temporales: `contact` y `sustain`

Las dos confusiones más costosas de `commit` son temporales. Con `contact`, el error común es `contact.press` + `fulfill`: se celebra antes de saber el resultado, y si la operación falla después, la interfaz mintió. Con

sustain, la secuencia correcta es sustain.progress → commit.complete: sustain mantiene, commit cierra, y confundirlos produce incertidumbre — el usuario no sabe si la operación sigue viva o ya terminó—.

§ 8

Ejemplos prácticos

Las secuencias habituales muestran el reparto:

- **Guardar** → contact.press · sustain.progress · commit.save + affirm
- **Enviar** → contact.press · sustain.progress · commit.submit + fulfill
- **Error** → contact.press · commit.fail + risk
- **Eliminación** → signal.warn + threat · commit.delete + loss
- **Toggle** → contact.press · commit.toggle + affirm

§ 9

Accesibilidad

commit debe ser visible, persistente y comprensible, y debe sobrevivir sin color, sin sonido y sin motion. Si el usuario no sabe qué ocurrió después de actuar, el commit falló.

§ 10

Antipatrones

Success prematuro. Confirmar como hecho lo que solo se ha pedido. Si la operación falla después, la interfaz mintió. Affirm y fulfill llegan cuando el resultado queda, no cuando se solicita.

Pérdida sin huella. Algo desaparece con un desvanecimiento y no queda rastro, registro ni deshacer. El usuario no puede verificar qué pasó ni repararlo. Todo loss deja huella perceptible y, si es posible, camino de vuelta.

Resultado mudo. La acción se aplica pero nada lo dice: el usuario guarda y se queda mirando, sin saber si guardó. Todo commit relevante tiene expresión propia, aunque sea mínima.

§ 11

Síntesis

commit es la familia donde la interfaz produce consecuencias: comunica qué resultado quedó y qué cambió en el sistema. Su tesis cabe en una frase.

DISTINCIÓN

El significado de la interfaz no está en el click. Está en lo que queda después.

PRINCIPIO

Todo evento importante debe dejar una huella perceptible de su consecuencia.

TABLA 23.1 Tipos de commit y su lectura

Tipo	Ejemplo	Lectura
Estado	commit.select	esto queda elegido
Proceso	commit.complete	esto terminó
Consecuencia	commit.delete	esto ya no está

TABLA 23.2 Canales de commit y su función

Canal	Función
Forma	estado persistente
Presencia	aparición o retirada
Motion	asentamiento o cierre
Color	refuerzo de estado o valencia
Sonido	cierre opcional

Háptica	impacto o encaje opcional
Tiempo	marca el momento de resolución

TABLA 23.3 Migraciones de commit cuando falta un canal

Canal ausente	Sustituto
Color	forma + texto
Motion	estado persistente
Sonido	mensaje visible
Háptica	estado + motion

TABLA 23.4 Antipatronos de commit

Antipatrón	Problema
Sin commit visible	incertidumbre
Commit efímero	no deja huella
Todo es success	trivializa resultados
Todo es danger	mezcla pérdida y amenaza
Celebración constante	fatiga
Sin diferencia entre affirm y fulfill	pérdida de matiz
Resultado solo en color	inaccesible

TABLA 23.5 Preguntas de validación de commit

Dimensión	Pregunta
Estado	¿qué quedó aplicado?
Resultado	¿qué ocurrió?
Cierre	¿puedo continuar?
Consecuencia	¿qué cambió en el sistema?
Evaluación	¿cómo debo interpretarlo?

24

Signal

Cuando el sistema reclama atención

PREGUNTA DEL CAPÍTULO

¿Cómo reclama el sistema atención sin confundir aviso, amenaza y pérdida?

Un punto rojo se posa sobre un icono. Una pestaña parpadea. Algo, en algún borde de la pantalla, dice «mírame». No has hecho nada para provocarlo: es el sistema quien reclama. Y la pregunta inmediata no es qué quiere, sino cuánta urgencia trae: si es un aviso que puede esperar o una alarma que no.

Si contact responde a «¿el sistema ha sentido mi acción?» y commit a «¿qué ha pasado como consecuencia?», signal responde a una pregunta distinta: ¿debo atender esto? Aquí el flujo se invierte: no es el usuario quien actúa primero, sino el sistema quien orienta, interrumpe o reclama la atención. Una advertencia antes de eliminar, un error en un campo, un badge que indica algo nuevo, una notificación, un aviso de expiración de sesión, una validación: todo eso pertenece a signal.

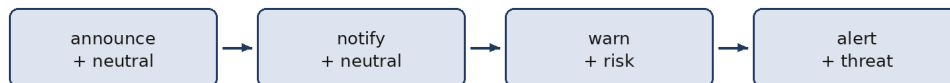
FIGURA 24.1

signal: la ficha de la familia

PREGUNTA PERCEPTIVA

¿Debo atender esto ahora?

SECUENCIA CANÓNICA



FRONTERA CLAVE

signal.warn + risk — «revisa esto»

≠

signal.alert + threat — «actúa ahora»

Una escala de atención: aviso, corrección posible y urgencia no se expresan igual.

No todas las señales son alertas; la escala protege a la alarma.

§ 1

Qué es signal

signal es la familia semántica de la orientación y reclamación de atención. No ejecuta una acción, no fija un resultado, no cambia el marco por sí sola: hace algo más básico.

PRINCIPIO

Un signal dirige la percepción del usuario hacia algo relevante.

Sus verbos gradúan esa reclamación: *announce*, *notify*, *warn*, *alert*, *inform*, *emphasize*, *remind*.

§ 2

Qué no es signal

Cada frontera de la familia deja fuera una confusión frecuente:

- **signal no es commit.** `signal.warn` $\neq$ `commit.fail`: una advertencia no es un resultado. Un resultado puede necesitar señal, pero no son lo mismo.
- **signal no es contact.** No es respuesta a una acción; puede aparecer sin gesto directo del usuario.
- **signal no es emerge.** `emerge.open` $\neq$ `signal`: algo puede aparecer sin reclamar atención.
- **signal no es shift.** El modal no es la señal; el mensaje que contiene puede serlo.

§ 3

Fundamento psicoperceptivo: atención, saliencia y carga cognitiva

signal se apoya en la orientación atencional. La percepción no procesa todo por igual: el sistema atencional decide qué entra en foco, qué queda en fondo, qué se pospone y qué interrumpe. Una señal puede orientar, informar, advertir, interrumpir, mantener algo en vigilancia —o, si se abusa de ella, fatigar—.

El concepto clave es la saliencia: la capacidad de un estímulo para destacar del fondo y reclamar procesamiento. Y la saliencia tiene coste: demasiada produce ruido; demasiado poca, invisibilidad. Por eso signal no es simplemente «mostrar algo», sino regular cuánta atención se pide. La familia se apoya en los mecanismos de orientación atencional y en la distinción entre estímulos que simplemente aparecen y estímulos que exigen procesamiento.

§ 4

La pregunta perceptiva

signal responde a: ¿debo notar esto? Y, en los casos de mayor urgencia, a su forma intensificada: ¿debo actuar ahora?

§ 5

Tipos de señal

Los cuatro verbos centrales forman una escala de urgencia, cada uno con su lectura:

- **signal.announce + neutral** → «esto existe», sin urgencia.
- **signal.notify + neutral / affirm** → «algo ocurrió», pero no requiere acción inmediata.
- **signal.warn + risk** → «revisa esto».
- **signal.alert + threat** → «atiende ahora».

§ 6

Canales, texto e intent

signal es una familia multicanal fuerte, y en ella el texto tiene un papel especial: es el canal que puede explicar con precisión qué ocurre y qué debe hacer el usuario.

La relación con el intent es central, porque la escala de urgencia y la escala evaluativa van de la mano: announce con neutral no carga afecto; notify con affirm es confirmación ligera; warn con risk señala un problema corregible; alert con threat trae urgencia. loss es menos común en signal, porque la pérdida suele vivir en commit, pero una señal puede anunciar una pérdida ya ocurrida —signal.announce + loss— cuando lo que el sistema hace es comunicar una consecuencia previa.

§ 7

signal frente a emerge y commit

Las dos fronteras más finas de la familia son estas. Con `emerge`: `emerge` hace que algo aparezca; `signal` hace que eso que aparece importe. El error común es `emerge.open + danger`, que carga el contenedor en lugar del evento evaluable. Con `commit`: la señal puede ocurrir antes o durante; el `commit` ocurre como resultado. `signal.warn + risk` dice «revisa este campo»; `commit.fail + risk` dice «la operación no se completó». La primera aún puede evitar la segunda.

§ 8

Ejemplos prácticos

- **Error de formulario** → `signal.warn + risk`: forma inline, color moderado, texto claro, persistente hasta la corrección.
- **Notificación informativa** → `signal.notify + neutral`: presencia leve, no interrumpe.
- **Alerta crítica** → `signal.alert + threat`: presencia fuerte, posible sonido, persistente hasta la acción.
- **Información contextual** → `signal.announce + neutral`: no es el intent «info» de la convención heredada, sino una señal neutra.

§ 9

Accesibilidad

`signal` es la familia más crítica en accesibilidad. Una señal accesible no depende solo del color, lleva texto claro, persiste si importa, usa `live region` cuando procede, no es solo sonora, no desaparece demasiado rápido y no interrumpe sin necesidad.

§ 10

Antipatronos

Todo es urgente. Usar alert para lo que merece notify. La urgencia constante se convierte en ruido de fondo, y la alarma real llega devaluada. La gravedad de la señal corresponde a la gravedad del evento; la mayoría de los eventos no son urgentes.

Insistencia sin memoria. La señal reaparece aunque ya fue vista y atendida. El sistema no registra la atención del usuario y lo castiga por haberla prestado. Una señal atendida cambia de estado o se retira.

Señal sin salida. Se reclama atención pero no se ofrece acción: qué revisar, qué corregir, dónde ir. La atención convocada no tiene dónde aterrizar. Toda señal que pide actuar dice cómo.

§ 11

Síntesis

signal es la familia de la atención. No actúa, no fija, no transforma el sistema directamente: dirige la percepción del usuario, y comunica que algo merece atención y con qué grado de urgencia.

PRINCIPIO

Una interfaz no solo responde a acciones. También guía la atención.

TABLA 24.1 Tipos de signal, intent y lectura

Tipo	Intent	Lectura
announce	neutral	existe
notify	neutral / affirm	ocurrió algo
warn	risk	revisa
alert	threat	actúa ahora

TABLA 24.2 Canales de signal y su función

Canal	Función
-------	---------

Presencia	aparición visible
Forma	separar de contenido
Color	valencia y saliencia
Texto	significado explícito
Tiempo	persistencia
Motion	entrada / énfasis
Sonido	opcional, fuera de foco
Háptica	opcional, alertas

TABLA 24.3 Migraciones de signal cuando falta un canal

Problema	Solución
solo color	añadir texto/icono
solo sonido	añadir visual
efímero	persistencia
fuera de foco	live region
exceso de alerta	graduar

TABLA 24.4 Antipatrones de signal

Antipatrón	Problema
Todo es alert	fatiga
Info como intent	confusión
Warning para todo	pierde significado
Señales efímeras	no se procesan
Exceso de motion	ruido
Modal para todo	interrupción

TABLA 24.5 Preguntas de validación de signal

Dimensión	Pregunta
Atención	¿debo mirar esto?
Urgencia	¿debo actuar ahora?

Saliencia	¿qué destaca?
Contexto	¿qué significa esto?

25

Handle

Cuando el usuario manipula directamente un objeto

PREGUNTA DEL CAPÍTULO

¿Cómo se percibe que un objeto está bajo el control directo del usuario?

Arrastras un archivo hacia una carpeta. Mientras lo mueves, el archivo te sigue, una zona se ilumina para decir «aquí cabe» y otra se resiste. Tienes algo cogido, y lo sabes con la mano antes que con la cabeza. Soltarlo en el sitio correcto, o fuera, será una decisión que se siente física.

Hasta ahora hemos visto familias donde la interacción suele ser discreta: `contact` registra una acción, `commit` fija una consecuencia y `signal` reclama atención. Pero hay un tipo de interacción donde el evento no cabe en un instante. El usuario no solo pulsa, ni solo confirma, ni solo observa: manipula. Arrastra, reordena, redimensiona, desliza, mueve, agarra, suelta. Aquí aparece otra pregunta —¿estoy controlando directamente este objeto?— y esa pregunta define `handle`.

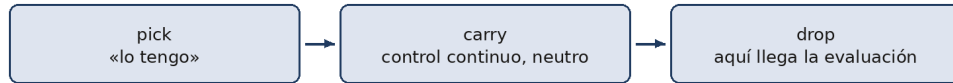
FIGURA 25.1

handle: la ficha de la familia

PREGUNTA PERCEPTIVA

¿Está este objeto bajo mi control directo?

SECUENCIA CANÓNICA



FRONTERA CLAVE

handle — control motor continuo

≠

contact — registro puntual

El carry pide silencio y fluidez; la evaluación se concentra al soltar.

El control vive en el gesto; la consecuencia, en el drop.

§ 1

Qué es handle

handle es la familia semántica de la manipulación directa. No es una acción puntual, ni un resultado, ni una señal: es una relación continua entre el usuario y un objeto. Es lo que la investigación clásica llamó manipulación directa —operar sobre una representación continua del objeto, con acciones reversibles y respuesta inmediata (Shneiderman, 1983; Hutchins, Hollan y Norman, 1985)—, a la que esta gramática añade su lectura como evento perceptivo con estructura temporal propia.

Sus verbos nombran esa relación: *pick*, *carry*, *drop*, *drag*, *resize*, *rotate*, *reorder*, *scroll*, *zoom*. A diferencia de otras familias, handle no es un instante: es una secuencia sostenida en el tiempo.

§ 2

Qué no es handle

Cada frontera deja fuera una confusión frecuente:

- **handle no es contact.** `contact.press` $\neq$ `handle.pick`: `contact` inicia, `handle` continúa.
- **handle no es commit.** `handle.drop` $\neq$ `commit.delete`: soltar no es necesariamente el resultado; el resultado ocurre después o en composición.
- **handle no es sustain.** `sustain.progress` $\neq$ `handle.carry`: un proceso automático no es control directo.
- **handle no es shift.** Mover un objeto no es cambiar de vista.

§ 3

Fundamento psicoperceptivo: control directo, esquema corporal y continuidad

`handle` se apoya en la percepción de control directo sobre un objeto. Cuando el usuario arrastra algo, espera que ese objeto responda de inmediato, siga su gesto, conserve su identidad, se comporte de forma coherente y, al soltarlo, se asiente, encaje o sea rechazado. Esto conecta con esquemas sensoriomotores básicos —agarrar, trasladar, soltar—: el usuario no piensa en eventos discretos, piensa «estoy moviendo esto», y la interfaz debe sostener esa ilusión.

Si el objeto se retrasa, el control se rompe. Si se mueve de forma extraña, el usuario pierde confianza. Si no se sabe qué se está manipulando, hay ambigüedad. `handle` es una de las familias más corporales del sistema, porque convierte la interfaz en espacio de acción continua.

§ 4

Las fases de `handle`

handle tiene tres fases fundamentales, cada una con su pregunta y su lectura:

- **Pick** — `handle.pick` responde a «¿he agarrado esto?». Se expresa por profundidad, motion, forma y háptica opcional, y su lectura es «esto está bajo mi control».
- **Carry** — `handle.carry` responde a «¿sigue bajo mi control?». Exige motion continuo, latencia mínima y coherencia espacial; su lectura es «estoy moviendo esto».
- **Drop** — `handle.drop` responde a «¿qué ocurrió al soltar?». Es la fase donde handle se conecta con otras familias —`commit.reorder + affirm`, `commit.delete + loss`, `signal.warn + risk`— y su lectura se abre: «esto se colocó», «esto no encaja», «esto se perdió».

§ 5

Canales y evaluación

handle es una de las familias más exigentes en canales: motion y profundidad son centrales, porque el objeto debe sentirse separado del layout mientras está en control directo.

Con el intent, la regla es de fase: el carry normalmente no debe cargar intent, porque durante el carry el usuario planifica y controla; la evaluación aparece sobre todo en el drop. Así, `handle.pick` → `handle.carry` → `handle.drop + risk` si el destino es inválido, o `handle.pick` → `handle.carry` → `commit.delete + loss` si el drop produce eliminación. Esto evita inyectar evaluación en una fase que aún no tiene resultado.

Hay un problema de rango que solo tienen los eventos continuos. Durante un carry, el sensor emite muestras a un ritmo muy superior al que el ojo o el oído resuelven: docenas o cientos por segundo. Expresar cada muestra como señal completa inunda los canales y aplana el gesto a un tono constante.

La regla es muestrear al ritmo de la percepción, no del sensor: las señales del movimiento se agrupan al refresco perceptivo y se les impone un suelo mínimo por debajo del cual repetir no añade nada. Las entradas discretas a ritmo humano —una tecla que se repite— no se muestrean: cada repetición es ya un valor comprometido y merece su evento.

§ 6

Ejemplos prácticos

- **Drag & drop** → `contact.press · handle.pick · handle.carry · handle.drop · commit.move + affirm`
- **Reordenar una lista** → `handle.pick · handle.carry · handle.drop · commit.reorder + affirm`
- **Resize** → `handle.resize · commit.resize + affirm`
- **Slider** → `handle.drag · commit.set + affirm`

§ 7

Accesibilidad

handle es la familia más difícil de hacer accesible, porque suele depender de gesto continuo, precisión motora y control espacial. Las soluciones son conocidas: alternativa por teclado, botones de subir y bajar, el patrón selección + acción, zonas explícitas, respuesta textual, estado visible del objeto, cancelación clara —y, sobre todo, no depender solo del drag-and-drop—.

PRINCIPIO

La accesibilidad no exige el mismo gesto. Exige el mismo resultado operativo.

§ 8

Antipatrones

Arrastre ciego. No se ve qué quedó cogido, ni qué destinos aceptan, ni dónde caerá. El usuario manipula a oscuras. El régimen de manipulación es visible de principio a fin: qué tengo, dónde puede ir, qué pasará al soltar.

Drop mudo. Se suelta y nada confirma el encaje ni el rechazo. La manipulación termina sin desenlace legible. El drop es evaluable: el encaje o el rechazo se expresan en el momento.

Manipulación sin escape. Una vez cogido, no hay forma perceptible de cancelar. El error se ve obligado a consumarse. Siempre existe una salida legible —soltar fuera, una tecla de escape— y se percibe que existe.

§ 9

Síntesis

handle es la familia de la manipulación directa: comunica que el objeto está bajo el control del usuario y qué puede hacerse con él. No es un evento puntual, sino una secuencia sostenida —pick, carry, drop— que convierte la interfaz en espacio de acción continua.

PRINCIPIO

El objeto debe sentirse bajo control en todo momento.

TABLA 25.1 Fases de handle y su lectura

Fase	Evento	Lectura
Pick	handle.pick	agarro
Carry	handle.carry	controlo
Drop	handle.drop	suelto / evalúo

TABLA 25.2 Canales de handle y su función

Canal	Función
-------	---------

Tiempo	latencia mínima
Motion	continuidad
Profundidad	régimen de manipulación
Forma	objeto activo / asa
Color	destino válido/inválido
Sonido	opcional en pick/drop
Háptica	opcional en pick/drop

TABLA 25.3 Antipatrones de handle

Antipatrón	Problema
Lag en drag	pierde control
Sin elevación	no se entiende pick
Objeto rígido	no sigue gesto
Drop ambiguo	no se sabe resultado
Sin alternativa	inaccesible
Respuesta continua	fatiga
Intent durante carry	ruido evaluativo

TABLA 25.4 Preguntas de validación de handle

Dimensión	Pregunta
Control	¿esto está bajo mi acción?
Continuidad	¿sigue siendo el mismo objeto?
Movimiento	¿lo estoy moviendo?
Resultado	¿qué ocurre al soltar?

26

Emerge

Cuando algo entra o sale del campo perceptivo

PREGUNTA DEL CAPÍTULO

¿Cómo entra y sale algo del campo perceptivo sin cambiar el lugar donde se trabaja?

Pulsas un icono y un menú se despliega. Pasas el cursor y asoma una explicación. Cierras un panel y la pantalla vuelve a respirar. Nada se ha confirmado y nada ha cambiado de estado: solo ha cambiado lo que está disponible a la vista. Es el movimiento más callado de una interfaz, y uno de los que más se diseñan mal.

Hasta ahora hemos visto familias en las que el sistema responde, fija consecuencias, reclama atención o permite manipular objetos. Pero hay otra clase de evento más silenciosa y estructural: algo aparece, se revela, se despliega, se oculta, se retira; algo entra en el campo perceptivo o deja de estar presente, pasando a ocupar —o a abandonar— una figura reconocible en el campo. Ese evento no siempre comunica éxito, error, riesgo o pérdida: muchas veces solo reorganiza la presencia. La pregunta de emerge es: ¿algo ha entrado o salido del campo perceptivo?

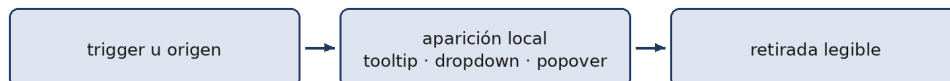
FIGURA 26.1

emerge: la ficha de la familia

PREGUNTA PERCEPTIVA

¿Qué acaba de entrar (o salir) del campo?

SECUENCIA CANÓNICA



FRONTERA CLAVE

emerge — aparece algo

≠

shift — cambia el marco

Si reorganiza el foco o bloquea el fondo, ya no es una aparición: es otro régimen.

Presencia local, sin cambiar el lugar donde se trabaja.

§ 1

Qué es emerge

emerge es la familia semántica de la aparición y retirada perceptiva. No nombra el contenido que aparece, ni su evaluación, ni necesariamente una decisión del usuario: nombra el hecho de que algo pasa a estar presente o deja de estarlo. Sus verbos recorren ese movimiento: *present*, *dismiss*, *open*, *close*, *expand*, *collapse*, *reveal*, *hide*. emerge no dice todavía «esto es bueno», «esto es malo», «esto es urgente» ni «esto se perdió».

PRINCIPIO

Algo entra en el campo perceptivo; algo sale del campo perceptivo.

§ 2

Qué no es emerge

Cada frontera deja fuera una confusión frecuente:

- **emerge no es signal.** `emerge.open` $\neq$ `signal.warn`: algo puede aparecer sin reclamar atención urgente. Si lo que aparece contiene una advertencia, hay composición —`emerge.open` seguido de `signal.warn` + `risk`—: `emerge` organiza presencia; `signal` reclama atención.
- **emerge no es shift.** `emerge.open` $\neq$ `shift.enter-mode`: un elemento puede aparecer sin cambiar el contexto.
- **emerge no es commit.** `emerge.dismiss` $\neq$ `commit.delete` + `loss`: cerrar un panel no es perder un recurso.

§ 3

Fundamento psicoperceptivo: figura/fondo, presencia y organización del campo

`emerge` se apoya en la organización perceptiva del campo. La percepción no recibe la pantalla como una lista plana de elementos: organiza lo que aparece como figura, fondo, región, objeto, superficie, contorno, relación y prioridad. En esa organización, la aparición de una nueva figura modifica el campo perceptivo.

Un tooltip que aparece junto a un término no solo añade un rectángulo: convierte una información auxiliar en figura local. Un dropdown que se abre no solo añade opciones: revela una región dependiente de un control. Un accordion que se expande no solo aumenta la altura de una sección: convierte contenido oculto en parte del campo procesable. `emerge` suele ser transicional: reorganiza la presencia para que algo pueda verse, leerse, elegirse o evaluarse.

§ 4

La pregunta perceptiva

emerge responde a: ¿qué acaba de entrar o salir del campo? Y arrastra las preguntas que la acompañan: ¿de dónde viene?, ¿a qué pertenece?, ¿puedo interactuar con ello?, ¿es temporal o persistente?, ¿se ha ocultado o se ha perdido?

§ 5

Verbos de emerge

Los verbos de emerge nombran el movimiento de entrada o salida del campo perceptivo:

- **emerge.open / emerge.close** — algo se abre o se cierra con marco propio: menú, popover, panel.
- **emerge.expand / emerge.collapse** — algo contenido se despliega o se contrae en su propio sitio: accordion, sección expandible.
- **emerge.present / emerge.dismiss** — el sistema ofrece algo, o el usuario lo despide: barra de deshacer, sugerencia, tarjeta auxiliar.
- **emerge.reveal / emerge.hide** — algo oculto se hace visible, o se aparta de la vista sin dejar de existir.

El verbo nombra qué ocurrió. El tipo, en la sección siguiente, nombra cómo se organiza perceptivamente esa aparición o retirada.

§ 6

Tipos de emerge

- **Aparición anclada** — emerge.open: dropdown, tooltip, popover. Lectura: «esto viene de aquí».
- **Expansión interna** — emerge.expand: accordion, details, sección desplegable. Lectura: «esto estaba contenido aquí».
- **Retirada reversible** — emerge.dismiss, emerge.close, emerge.hide. Lectura: «se fue de la vista, pero no se perdió».

- **Aparición auxiliar** — `emerge.present`: la barra de deshacer tras `commit.delete + loss` es el caso canónico.

§ 7

Canales e intent

En canales, la regla de emerge es de tono: debe ser legible, no necesariamente saliente. Presencia, motion anclado al origen y profundidad hacen el trabajo; el color es secundario y el sonido, normalmente, sobra.

Con el intent, la regla es estructural: emerge normalmente no lo acepta, porque su función es introducir presencia; lo evaluable será lo que aparece dentro o lo que ocurre después. Incorrecto: `emerge.open + threat`. Más preciso: `emerge.open` seguido de `signal.warn + threat` —la amenaza está en la advertencia, no en el hecho de abrir—. Con una excepción estrecha, desarrollada en el Capítulo 9: la aparición cuya única razón de ocurrir es advertir. Si al quitarla la advertencia no sobrevive en ningún otro sitio, la aparición puede portar el tono —declarándolo.

§ 8

Las fronteras: signal, shift y commit

Tres pares de contraste delimitan la familia. Frente a `signal`: el tooltip es `emerge.present`; la alerta inline es `signal.warn + risk`; y su composición —`emerge.present` con `signal.warn + risk` dentro— se lee «algo aparece, y ese algo contiene una advertencia». Frente a `shift`: el dropdown es `emerge.open` y el modal es `shift.enter-mode`; el primero introduce presencia local, el segundo cambia el régimen operativo. Y frente a `commit`: `emerge.dismiss` significa «se retiró de la vista»; `commit.delete + loss` significa «dejó de estar disponible». No es lo mismo ocultar que perder, y la semántica debe distinguir retirada, ocultación, colapso y pérdida.

§ 9

Ejemplos prácticos

- **Dropdown** → `contact.press · emerge.open · commit.select + affirm · emerge.close`
- **Tooltip** → `contact.focus · emerge.present · emerge.dismiss`
- **Accordion** → `contact.press · emerge.expand · emerge.collapse`
- **Eliminación con undo** → `signal.warn + threat · commit.delete + loss · emerge.present (undo)`

§ 10

Accesibilidad

emerge tiene riesgos importantes, y todos comparten la misma raíz: que algo aparezca visualmente pero no exista para una parte de los usuarios.

PRINCIPIO

Si algo emerge visualmente, también debe emerger operativamente.

§ 11

Antipatrones

Aparición con amenaza. `emerge.open + threat`: cargar el intent en la apertura. Abrirse no es peligroso; lo evaluable está dentro. La forma correcta: `emerge.open → signal.warn + threat` — salvo que la aparición sea, ella misma, la única advertencia (la excepción declarada del Capítulo 9).

Emergencia a medias. Algo aparece visualmente pero no operativamente: no recibe foco, no se anuncia, no entra en el orden de navegación. Para una parte de los usuarios, no ha aparecido. Si algo emerge visualmente, emerge operativamente.

Retirada que se lleva el contexto. Lo que se cierra desaparece con información o estado que el usuario aún necesitaba, sin dejar huella. La retirada es reversible o deja constancia de lo retirado.

§ 12

Síntesis

emerge es la familia de la presencia: comunica qué ha pasado a estar disponible y qué se ha retirado de la vista. Su tesis es de límites: emerge no dice qué significa lo que aparece; dice que algo ha pasado a estar presente.

PRINCIPIO

La aparición debe ser proporcional al régimen que introduce.

TABLA 26.1 Verbos de emerge y su lectura

Verbo	Lectura
present	algo se muestra
dismiss	algo se retira de la atención
open	una superficie o región se abre
close	una superficie o región se cierra
expand	una región crece y revela contenido
collapse	una región se contrae
reveal	algo oculto se hace visible
hide	algo sigue existiendo pero deja de verse

TABLA 26.2 Canales de emerge y su función

Canal	Función
Presencia	hacer visible o retirar
Motion	mostrar origen, entrada, salida
Profundidad	indicar plano flotante o dependencia
Forma	definir superficie o región

Tiempo	duración de aparición y retirada
Color	secundario
Sonido	normalmente no
Háptica	normalmente no

TABLA 26.3 Riesgos de accesibilidad de emerge y su soporte

Riesgo	Soporte necesario
contenido aparece pero no se anuncia	estado abierto/cerrado
tooltip inaccesible	alternativa textual / descripción
dropdown sin teclado	navegación y escape
accordion visual solamente	aria-expanded, relación trigger-panel
popover flotante	foco y cierre claro
contenido oculto aún accesible	coherencia DOM/árbol accesible

TABLA 26.4 Antipatrones de emerge

Antipatrón	Qué falla	Corrección
Todo lo que aparece parece alerta	exceso de saliencia	separar emerge de signal
Dropdown como modal	profundidad excesiva	mantener anclaje local
Tooltip inaccesible	información perdida	descripción accesible
Cerrar como eliminar	confusión de consecuencia	separar dismiss y delete + loss
Fade sin origen	dependencia ambigua	motion anclado
Contenido oculto pero accesible	incoherencia operativa	sincronizar DOM/ARIA/visibilidad

TABLA 26.5 Preguntas de validación de emerge

Dimensión	Pregunta
Aparición	¿qué acaba de entrar en el campo?
Retirada	¿qué acaba de salir?

Origen	¿de dónde viene?
Dependencia	¿a qué pertenece?
Reversibilidad	¿se ocultó o se perdió?
Campo	¿cómo se reorganiza lo visible?

27

Shift

Cuando cambia el contexto

PREGUNTA DEL CAPÍTULO

¿Cómo se percibe que el contexto de trabajo ha cambiado, sin que el usuario se pierda?

El usuario toca un elemento y, de pronto, las reglas son otras: lo que antes se leía ahora se edita, los botones de antes han desaparecido y han aparecido otros. No se ha movido una pieza del tablero: ha cambiado el tablero entero. Y si nada avisa de que se ha cruzado una frontera, quien usa la interfaz se queda un instante sin saber dónde está.

Hasta ahora hemos visto eventos que ocurren dentro de un mismo contexto: el sistema responde, algo queda fijado, algo reclama atención, el usuario manipula, algo aparece o desaparece. Pero hay un tipo de evento distinto, que no cambia solo un elemento ni hace aparecer solo una superficie: cambia el lugar desde el que todo lo demás se interpreta. El usuario pasa a otro estado del sistema. Ese evento es shift.

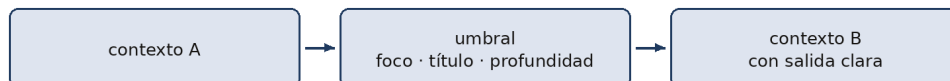
FIGURA 27.1

shift: la ficha de la familia

PREGUNTA PERCEPTIVA

¿Sigo en el mismo marco o ha cambiado el régimen?

SECUENCIA CANÓNICA



FRONTERA CLAVE

shift — cruza un umbral operativo

≠

emerge — solo aparece algo

Cruzar exige orientación: saber dónde se está y cómo volver.

Un cambio de contexto sin orientación deja al usuario sin mapa.

§ 1

Qué es shift

shift es la familia semántica del cambio de contexto. No describe un contenido, ni una señal, ni una acción puntual: describe algo más profundo.

PRINCIPIO

El contexto en el que el usuario actúa ha cambiado.

Sus verbos recorren las formas de ese cruce: *enter-mode*, *exit-mode*, *navigate*, *step*, *route*, *context*, *return*. Abrir un modal, entrar en edición, ir a otra pantalla, avanzar en un flujo, cambiar de vista, activar un modo especial: todo eso es shift.

§ 2

Qué no es shift

Cada frontera deja fuera una confusión frecuente:

- **shift no es emerge.** `emerge.open` $\neq$ `shift.enter-mode`: un dropdown aparece; un modal cambia el contexto.
- **shift no es commit.** `shift.navigate` $\neq$ `commit.submit`: ir a otra pantalla no implica que la operación haya terminado.
- **shift no es signal.** Un cambio de vista no es una advertencia.
- **shift no es handle.** Manipular un objeto no es cambiar el marco.

§ 3

Fundamento psicoperceptivo: cambio de contexto, orientación y modelo mental

shift se apoya en un fenómeno clave: el usuario opera siempre dentro de un modelo mental activo, y cuando ese modelo cambia, la percepción cambia con él. El cambio no es solo visual: es semántico. Editar frente a visualizar, navegar frente a confirmar, formulario frente a resultado, lista frente a detalle, selección simple frente a modo bulk: en cada uno de esos cruces cambia qué puedo hacer, qué significa cada acción, qué es relevante, qué está permitido, qué queda subordinado y cómo vuelvo.

Por eso el usuario necesita reorientarse. Si el sistema no comunica el cambio de contexto, aparece la desorientación con sus preguntas características: ¿dónde estoy?, ¿qué ha cambiado?, ¿puedo volver?, ¿qué puedo hacer ahora? shift no es opcional: es necesario para mantener la coherencia cognitiva.

§ 4

La pregunta perceptiva

shift responde a: ¿sigo en el mismo contexto o estoy en otro? Y trae consigo las preguntas de la reorientación: ¿puedo hacer lo mismo aquí?, ¿cómo vuelvo?, ¿qué cambió?, ¿qué es ahora lo importante?

§ 5

Tipos de shift

- **Navegación** — shift.navigate: lista → detalle, página A → página B. Lectura: «estoy en otro lugar».
- **Cambio de modo** — shift.enter-mode / shift.exit-mode: visualizar → editar, normal → selección múltiple. Lectura: «estoy en otro régimen».
- **Flujo paso a paso** — shift.step: paso 1 → paso 2 → paso 3. Lectura: «avanzo en un proceso».
- **Cambio de contexto local** — shift.context: filtro activo, modo bulk, vista agrupada. Lectura: «el significado del campo cambió».

§ 6

Canales de shift

shift es altamente multicanal: la profundidad marca el nuevo plano, la presencia introduce el nuevo contexto, el motion orienta la transición y la forma estructura el nuevo estado. Pero hay un canal que manda sobre los demás: el foco.

PRINCIPIO

Si cambia el contexto, debe cambiar el foco.

§ 7

shift y familias relacionadas

Tres pares delimitan la familia. Con emerge, la diferencia es de alcance: emerge hace que aparezca algo; shift cambia el sistema de acción. El dropdown es emerge.open; el modal es shift.enter-mode —ambas

superficies aparecen, pero solo una cambia el contexto—. Con `commit`, la relación es de secuencia: una operación puede terminar (`commit.submit + fulfill`) y después llevar a otra vista (`shift.navigate`); el cierre y el traslado son eventos distintos. Y con `signal`, la relación es de composición: un modal puede cambiar el marco (`shift.enter-mode`) y contener una advertencia (`signal.warn + threat`); el marco no carga el intent, el mensaje sí.

§ 8

Ejemplos prácticos

- **Navegación** → `contact.press · shift.navigate`
- **Modal de confirmación** → `contact.press · shift.enter-mode · signal.warn + threat`
- **Wizard** → `shift.step` en cada avance
- **Modo edición** → `shift.enter-mode · ediciones dentro del modo (contact, commit.set) · commit.save + affirm · shift.exit-mode`

§ 9

Accesibilidad

`shift` es crítico en accesibilidad. Un cambio de contexto accesible mueve el foco, anuncia el cambio, actualiza el título, define landmarks, permite volver, evita la desorientación y bloquea correctamente lo que ya no es operativo.

PRINCIPIO

Si el usuario no sabe dónde está, el `shift` falló.

§ 10

Antipatrones

Teletransporte. El contexto cambia sin transición ni orientación: el usuario aparece en otro régimen sin saber cómo llegó ni cómo volver. El shift se percibe como cruce —de dónde, hacia dónde—, no como corte.

Shift disfrazado. El régimen entero cambia, pero la expresión es la de un panel más que aparece. El usuario cree que sigue donde estaba. Si cambian las reglas, la expresión declara el cambio de marco, no una aparición local.

Vuelta con amnesia. Al regresar, el estado anterior se perdió: el scroll, la selección, lo escrito. El viaje de ida tenía billete; el de vuelta no. Salir y volver restituye el contexto que se dejó.

§ 11

Síntesis

shift es la familia del cambio de contexto: comunica dónde está ahora el usuario y bajo qué reglas. Su tesis: shift redefine el espacio de acción del usuario.

PRINCIPIO

Todo cambio de contexto debe ser explícito y orientado.

TABLA 27.1 Canales de shift y su función

Canal	Función
Profundidad	nuevo plano
Presencia	aparición del nuevo contexto
Motion	transición orientadora
Tiempo	duración suficiente
Forma	estructura del nuevo estado
Foco	punto de entrada
Color	apoyo de modo
Sonido	ocasional

TABLA 27.2 Dropdown frente a modal: la frontera emerge/shift

Propiedad	Dropdown	Modal
Cambia contexto	no	sí
Bloquea fondo	no	sí
Requiere decisión	normalmente no	a menudo sí
Foco cambia	local	dominante

TABLA 27.3 Fallos de orientación en shift y su efecto

Problema	Efecto
foco no cambia	confusión
título no cambia	desorientación
fondo activo	incoherencia
navegación invisible	pérdida
modo sin anuncio	acciones mal interpretadas

TABLA 27.4 Antipatrones de shift

Antipatrón	Problema
Todo es modal	exceso de cambio
Cambio sin foco	desorientación
Animación sin orientación	ruido
Navegación sin contexto	pérdida
Modal sin bloqueo	incoherencia
Shift invisible	confusión
Modo sin salida clara	ansiedad

TABLA 27.5 Preguntas de validación de shift

Dimensión	Pregunta
Contexto	¿dónde estoy?

Acción	¿qué puedo hacer?
Continuidad	¿de dónde vengo?
Orientación	¿cómo vuelvo?

28

Sustain

Cuando algo sigue ocurriendo sin haber terminado

PREGUNTA DEL CAPÍTULO

¿Cómo se sostiene la percepción de que un proceso sigue vivo sin haber terminado?

Hay eventos que no se resuelven en un instante. El usuario pulsa guardar, pero el guardado tarda. Envía un archivo, pero la subida avanza lentamente. Abre una vista, pero los datos aún no llegan. Inicia una búsqueda, pero el sistema sigue procesando. En todos esos casos, el evento principal no es todavía éxito, error, pérdida ni confirmación; tampoco es simplemente contacto. Algo está ocurriendo, pero aún no ha terminado. Esa es la región de sustain.

La pregunta perceptiva de sustain es: ¿esto sigue ocurriendo? O, dicho desde la experiencia del usuario: ¿el sistema sigue vivo o se ha quedado bloqueado?

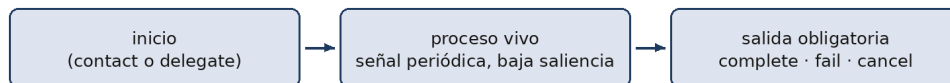
FIGURA 28.1

sustain: la ficha de la familia

PREGUNTA PERCEPTIVA

¿Sigue vivo el proceso?

SECUENCIA CANÓNICA



FRONTERA CLAVE

sustain — continuidad, sin juicio

≠

commit — el resultado que la cierra

La espera no es el resultado; pero toda espera debe terminar en uno.

Cuanto más dura una espera, más información debe ofrecer.

§ 1

Qué es sustain

sustain es la familia semántica de la continuidad operativa: comunica que una operación, estado o proceso sigue activo aunque todavía no haya producido una consecuencia final. Sus verbos recorren las formas de esa continuidad: *start*, *progress*, *loading*, *waiting*, *syncing*, *streaming*, *processing*, *pending*, *upload*, *retrying*, *end*.

Un spinner puede ser sustain; también una barra de progreso, un skeleton, un texto «guardando...» o un indicador de sincronización. Pero sustain no es el spinner, ni la barra, ni el skeleton, ni el texto: es la clase de evento que esos recursos intentan hacer perceptible.

PRINCIPIO

Una operación continúa sin haber llegado aún a su resultado.

§ 2

Qué no es sustain

Cada frontera deja fuera una confusión frecuente:

- **sustain no es contact.** `contact.press` $\neq$ `sustain.progress`: `contact` confirma que el sistema recibió la acción; `sustain` confirma que la operación iniciada sigue viva.
- **sustain no es commit.** `sustain.progress` $\neq$ `commit.complete`: un proceso en curso no es culminación.
- **sustain no es signal.** `sustain.progress` $\neq$ `signal.warn`: un proceso no es alerta por el hecho de durar.
- **sustain no es emerge.** `emerge.present` $\neq$ `sustain.progress`: puede aparecer un indicador, pero la aparición no es el proceso.

§ 3

Fundamento psicoperceptivo: continuidad temporal, incertidumbre y confianza

`sustain` se apoya en una necesidad básica de la interacción: cuando no hay resultado inmediato, el usuario necesita saber si el sistema sigue funcionando. La ausencia de respuesta prolongada no se percibe como neutral, sino como ambigua, y el usuario empieza a preguntarse: ¿se ha bloqueado?, ¿lo he hecho mal?, ¿debo esperar?, ¿debo repetir?, ¿perderé lo que estaba haciendo? `sustain` existe para reducir esa incertidumbre.

Trabaja con tres dimensiones. La continuidad temporal: percibir que el proceso no se ha detenido. La predicción: si hay progreso medible, el usuario puede anticipar el final. Y la confianza operativa: una señal proporcional permite sostener la confianza sin vigilancia constante — porque sin señal el proceso parece fallo, y con demasiada señal parece amenaza—. Por eso `sustain` no es un detalle visual: es una forma de mantener abierta una operación en la mente del usuario.

§ 4

La pregunta perceptiva

sustain responde a: ¿esto sigue ocurriendo? Y despliega las preguntas de la espera: ¿está vivo?, ¿está bloqueado?, ¿avanza?, ¿cuánto falta?, ¿puedo hacer otra cosa?, ¿puedo cancelar?, ¿debo esperar?

§ 5

Tipos de sustain

No todo sostenimiento tiene la misma forma temporal.

Progreso acotado. El proceso tiene medida y avance perceptible: una barra que crece, un porcentaje que sube. Es `sustain.progress` con magnitud, y es el sostenimiento más amable: promete un final y muestra el camino.

Continuidad indeterminada. El proceso sigue, pero no puede medirse: solo cabe afirmar que está vivo. Es el territorio del spinner, y por eso el que más texto honesto necesita.

Sostenimiento en segundo plano. El proceso continúa mientras el usuario hace otra cosa: sincronización, subida, exportación. Necesita presencia mínima pero localizable: poder comprobarlo sin tener que vigilarlo.

Estado vivo. Algo ocurre de forma continua y deliberada: grabación, emisión, conexión activa. Aquí el sostenimiento no es espera: es vigilancia compartida, y su interrupción es un evento de pleno derecho.

Cada tipo sostiene la misma promesa: el sistema sigue contigo. Lo que cambia es cuánta información puede dar mientras tanto.

§ 6

Canales de sustain

El texto es muchas veces el canal más honesto de sustain. Un spinner sin texto dice «algo pasa»; un texto dice «esto es lo que pasa».

§ 7

sustain e intent

sustain normalmente no acepta intent por sí mismo: un proceso en curso no es positivo o negativo por estar en curso. Puede volverse problemático, y entonces se compone (sustain.progress con signal.warn + risk); puede terminar bien (sustain.progress hacia commit.complete + fulfill); puede fallar (sustain.progress hacia commit.fail + risk). El intent aparece en la señal o en el resultado, no en la continuidad misma.

El modelo de evento puntual —una señal, una ventana perceptiva, fin— no cubre la continuidad larga. ¿Cómo se dice «sigue vivo» durante ocho segundos? No repitiendo la señal: un pulso reemitido choca con el muestreo perceptivo y con la memoria de frecuencia. La continuidad no es un pulso: es un estado.

El ciclo de vida de sustain tiene, por tanto, tres momentos: la transición de entrada, que sí es un evento y se puntúa; el estado sostenido, presencia persistente mientras dura el proceso —barra, texto, actualización discreta—; y la transición de salida hacia una resolución. La gramática puntúa las transiciones; la continuidad la lleva el estado.

PRINCIPIO

Sustain no es un pulso repetido: es un estado sostenido con dos transiciones puntuadas — la entrada y la salida.

§ 8

Las fronteras: commit y signal

Con commit, la relación es de relevo: en la secuencia contact.press · sustain.progress · commit.save + affirm, sustain mantiene abierta la

operación y commit la cierra. Si el sistema muestra «guardado» mientras todavía está guardando, rompe la confianza; si mantiene «guardando...» después de haber guardado, también.

Con signal, la relación es de tono: sustain acompaña; signal reclama atención. Si un proceso tarda más de lo esperado, se compone (sustain.processing con signal.warn + risk); si falla, se cierra (commit.fail + risk). No todo proceso lento debe convertirse en alerta, pero todo proceso que se aleja de las expectativas debe comunicarlo.

§ 9

Ejemplos prácticos

- **Guardado remoto** → contact.press · sustain.progress · commit.save + affirm
- **Subida de archivo** → contact.press · sustain.progress · commit.complete + fulfill
- **Búsqueda** → contact.submit · sustain.processing · emerge.present (resultados)
- **IA generando respuesta** → contact.submit · sustain.streaming · commit.complete + affirm / fulfill
- **Sincronización offline** → sustain.syncing · commit.save + affirm — o, si falla, sustain.syncing · commit.fail + risk

El tercero de estos ejemplos —la respuesta que fluye de un modelo generativo— es más difícil de lo que la línea sugiere. El Apéndice E lo examina entre los casos donde la gramática suda.

§ 10

Accesibilidad

sustain debe ser accesible porque la espera genera incertidumbre. La accesibilidad de sustain consiste, en el fondo, en hacer soportable la espera: estado anunciado, progreso legible, salida disponible.

§ 11

Antipatrones

Spinner eterno. Continuidad sin información: no se sabe cuánto queda, si avanza, ni cómo salir. La espera se vuelve sospecha. sustain dice qué pasa y, cuando puede, cuánto falta; y siempre deja una salida.

Proceso fantasma. Algo sigue ocurriendo en segundo plano sin ninguna presencia perceptible. Cuando termina —o falla—, el usuario nunca supo que existía. Lo que se sostiene tiene presencia proporcional, aunque sea mínima.

Final mudo. El proceso termina y la continuidad simplemente desaparece, sin commit que registre el desenlace. La historia queda sin última página. Todo sustain se cierra con un evento de resolución perceptible.

§ 12

Síntesis

sustain es la familia de la continuidad operativa. No responde a «¿se recibió mi acción?» —eso era contact— ni a «¿ya terminó?» —eso será commit—. Responde a «¿esto sigue ocurriendo?», y su función es mantener la confianza durante el intervalo entre acción y resultado.

PRINCIPIO

Todo proceso que no termina de inmediato debe indicar que sigue vivo, sin parecer alarma ni celebración.

Cierre del bloque de familias

Con estos capítulos queda consolidado el bloque de las familias semánticas. La gramática ya no se limita a nombrar ocho palabras: cada familia queda vinculada a una pregunta perceptiva, un fundamento psicoperceptivo, una relación con el intent, unos canales dominantes, ejemplos concretos, criterios de accesibilidad y antipatronos. Las familias no son fichas técnicas: son estructuras perceptivas del evento.

TABLA 28.1 Tipos de sustain y su lectura

Tipo	Qué comunica	Ejemplo	Riesgo
loading	carga indeterminada	skeleton, spinner	bloqueo aparente
progress	avance medible	barra, porcentaje	progreso falso
syncing	alineación de estado	guardando en nube	confundir con commit
processing	cálculo interno	búsqueda, IA, exportación	opacidad
streaming	flujo continuo	respuesta en vivo	no distinguir final
waiting	dependencia externa	esperando servidor	frustración

TABLA 28.2 Canales de sustain y su función

Canal	Función
Tiempo	sostener continuidad
Presencia	mantener visible el estado
Forma	representar proceso
Motion	indicar actividad o avance
Texto	explicar qué ocurre
Color	bajo, como estado auxiliar
Sonido	normalmente evitar
Háptica	normalmente evitar

TABLA 28.3 Fragilidades de sustain y su corrección

Problema	Riesgo	Corrección
solo spinner	ambiguo	texto de estado
motion continuo	malestar	reduced motion
sin progreso	ansiedad	barra o explicación
sin cancelación	bloqueo	control del usuario
sin final claro	duda	commit visible
anuncios excesivos	ruido	live regions dosificadas
skeleton largo	sensación de vacío	mensaje o alternativa

TABLA 28.4 Antipatrones de sustain

Antipatrón	Qué falla	Corrección
spinner eterno	ansiedad	progreso, texto, control
spinner sin texto	ambigüedad	estado explícito
progreso falso	desconfianza	progreso real o indeterminado
sustain como alarma	sobreactuación	neutralidad + signal si procede
resultado sin cierre	duda	commit claro
espera invisible	fallo percibido	indicador de continuidad
loop sonoro	fatiga	silencio

TABLA 28.5 Preguntas de validación de sustain

Dimensión	Pregunta
Continuidad	¿sigue vivo?
Progreso	¿avanza?
Tiempo	¿debo esperar?
Estado	¿qué está haciendo?
Control	¿puedo cancelar o seguir?
Cierre	¿cuándo pasó a commit?

29

Delegate

Cuando el sistema actúa por el usuario

PREGUNTA DEL CAPÍTULO

¿Quién actúa cuando actúa el sistema, y cómo se hace legible esa autoría?

La bandeja de correo amanece más limpia de lo que quedó anoche: un par de mensajes archivados, una reunión movida, una respuesta esperando como borrador que el usuario no escribió. No lo ha hecho él: lo ha hecho el asistente mientras no miraba. Y antes de saber si lo ha hecho bien, necesita saber algo más básico: ¿quién está actuando aquí, y hasta dónde puede llegar?

Las familias anteriores describen buena parte de las interfaces reactivas: el usuario actúa, el sistema responde, algo aparece, algo se sostiene, algo reclama atención, algo cambia de contexto o algo produce un resultado.

Pero muchas interfaces actuales introducen una pregunta adicional. El sistema ya no solo responde. Puede sugerir, planificar, generar, aplicar cambios, pedir revisión, esperar autorización o devolver el control.

En esos casos el usuario necesita saber algo muy concreto:

¿quién actúa ahora?

Esa pregunta no pertenece del todo a `contact`, `commit`, `signal`, `handle`, `emerge`, `shift` o `sustain`. Por eso esta gramática incorpora `delegate`.

§ 1

Dónde se sitúa el actor delegado

Antes de describir cómo se expresa `delegate`, hay que responder una pregunta que las otras siete familias no necesitaban plantear: cuando el sistema actúa por el usuario, ¿dónde queda ese actor? ¿Es parte del sistema? ¿Una capa entre el sistema y la interfaz? ¿El motor que la genera?

La respuesta de este libro es deliberadamente conservadora, y por eso mismo sólida: el actor delegado es un actor más dentro del contexto operativo. No introduce un nivel nuevo en la arquitectura. Introduce una nueva clase de actor dentro del nivel que ya existe.

Conviene recordar qué es el contexto operativo. No es solo el usuario frente a la interfaz: es todo lo que rodea a la acción y le da sentido. Y ese contexto ya contenía, desde antes de la IA, actores que no son el usuario. El servidor que responde a una petición; otro usuario que edita el mismo documento; una regla programada que archiva correos a medianoche. El usuario nunca fue el único que producía cambios en el contexto. Solo era el único que los producía deliberadamente desde la interfaz.

Un actor delegado —un asistente, un agente, una automatización— es otro de esos productores de cambio. Lo que lo distingue no es su naturaleza, sino su grado: actúa con más iniciativa que un servidor, con más alcance que una regla y, a menudo, con más opacidad que otro usuario. Pero son diferencias de grado, no de ontología. La interfaz sigue siendo lo que siempre fue: la dimensión perceptible del contexto. Lo único que ha

cambiado es que ese contexto ahora contiene acciones cuya autoría no es del usuario.

DISTINCIÓN

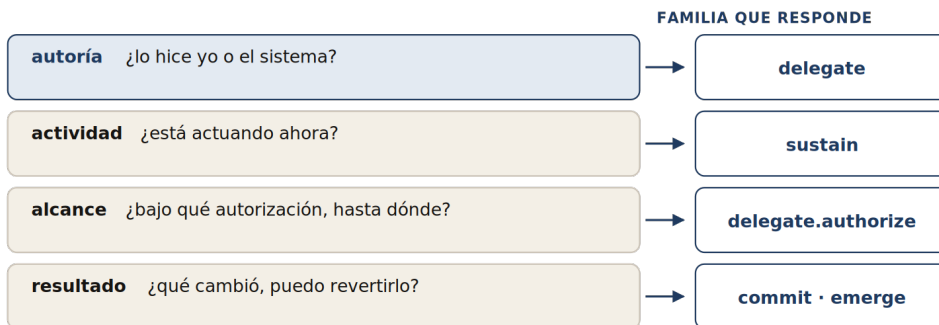
La IA no es una capa nueva entre el sistema y la interfaz. Es un actor nuevo dentro del contexto que la interfaz hace perceptible.

Esto tiene una consecuencia directa. Si el actor delegado vive en el contexto, lo que delegate debe hacer perceptible es exactamente lo que se vuelve incierto cuando alguien que no es el usuario actúa en su nombre: la autoría (¿lo hice yo o el sistema?), la actividad (¿está actuando ahora?), el alcance (¿bajo qué autorización, hasta dónde puede llegar?) y el resultado (¿qué cambió, y puedo revisarlo o revertirlo?).

FIGURA 29.3

Las cuatro preguntas que delegate hace perceptibles

lo que se vuelve incierto cuando alguien que no es el usuario actúa en su nombre



Delegate no inventa familias nuevas: organiza el reparto de iniciativa que las precede.

Estas cuatro preguntas no exigen una gramática nueva. Las familias que ya tenemos saben responderlas: la actividad se sostiene (sustain), el resultado se fija y deja huella (commit), lo que aparece para revisión emerge (emerge). delegate no sustituye a esas familias: organiza el reparto de iniciativa que las precede. Sus verbos —offer, authorize, act, review, return— no nombran efectos perceptivos nuevos, sino quién tiene el turno de actuar en cada momento.

Ahora bien: situar al actor delegado en el contexto es una decisión, y conviene nombrar las posiciones que un actor automático puede ocupar, porque no todas caen igual respecto a la interfaz. Hay tres, en una progresión ordenada. La del actor, que actúa dentro del contexto y produce cambios: es la posición que este capítulo desarrolla como núcleo, la más tratable, porque la autoría de una acción todavía puede señalarse. La del intérprete, que ya no solo actúa, sino que lee el contexto por el usuario —resume, filtra, decide qué merece volverse perceptible—; ahí la interfaz deja de ser la dimensión perceptible del contexto y pasa a serlo de una lectura del contexto. Y la del autor, que genera la propia dimensión perceptible: interfaces compuestas en tiempo de ejecución.

De estas tres posiciones, solo la del autor queda fuera de este libro, y por una razón precisa: generar la interfaz es un paso previo a la interacción, no parte de ella. Genere quien genere la superficie —un equipo, una plantilla o un modelo—, lo que llega al usuario sigue siendo una interfaz con la que tiene que interactuar y que debe ser legible; respecto a ella, esta gramática no cambia de objeto, cambia de papel: se vuelve el contrato que ese generador debe respetar para que el resultado siga siéndolo. El actor y el intérprete, en cambio, actúan sobre la interfaz con la que el usuario ya opera, y por eso ambos pertenecen a este capítulo: el actor como su núcleo, que las próximas secciones desarrollan; el intérprete como su frontera, que abordaremos al final, cuando lleguemos a lo que un agente decide *no* mostrar.

FIGURA 29.1

Las tres posiciones del actor automático



El actor y el intérprete operan sobre la interfaz; el autor la genera, y por eso queda fuera de este libro.

§ 2

Fundamento psicoperceptivo: atribución de agencia y autoría

Cada familia de este libro se apoya en algo que la percepción humana hace de forma espontánea: contact, en la percepción de causalidad; signal, en la captura de la atención; emerge, en la organización de figura y fondo. delegate se apoya en un fenómeno igual de básico y menos evidente: atribuimos agencia. Percibir que algo actúa por sí mismo, con iniciativa propia, no es una inferencia deliberada, sino una lectura perceptiva inmediata, anterior al razonamiento.

Conviene situar la palabra. *Agencia* viene del latín *agere* —hacer, actuar, poner en movimiento—, de donde deriva *agens*, «el que actúa»; pero su acepción moderna añade lo decisivo: una agencia es la entidad que actúa **en nombre de otro**, la relación entre un principal y un agente facultado para operar en su representación. Ese segundo sentido es el que importa aquí. La interfaz no actúa: es la dimensión perceptible donde la acción se vuelve legible. Quien actúa es un agente —el usuario, el sistema o un

actor delegado— que opera sobre ella, y cuando no es el propio usuario, actúa siempre *por cuenta de* alguien. Una interfaz *agentiva* es, en este sentido adjetival, una interfaz formada y dirigida por esa relación de agencia, no una interfaz que sea ella misma el agente. Por eso la pregunta de delegate tiene dos caras y no una: no solo «¿quién actúa?», sino «¿por cuenta de quién?».

PRINCIPIO

La agencia no se declara, se percibe. La interfaz no actúa: hace legible quién actúa sobre ella y por cuenta de quién.

Vemos intención en el movimiento

En 1944, Fritz Heider y Marianne Simmel mostraron a varios observadores una breve animación de tres figuras geométricas —dos triángulos y un círculo— moviéndose por la pantalla. Casi todos la describieron en términos de intenciones: un triángulo perseguía, otro protegía, el círculo intentaba escapar. No había personajes; había timing y trayectoria. Bastaron el ritmo y la dirección del movimiento para que la percepción asignara propósito.

Esto conecta con algo que el libro ya usó. Michotte, al estudiar la percepción de la causalidad, mostró que vemos a un objeto causar el movimiento de otro por su contigüidad en espacio y tiempo. La atribución de agencia es el paso siguiente: cuando el patrón de movimiento no se explica por una causa externa visible —cuando algo se mueve como si decidiera—, la percepción le atribuye iniciativa propia.

La consecuencia para delegate es directa: el usuario no percibe dónde está el sistema; percibe pistas de agencia. No le llega un dato técnico sobre qué proceso ejecutó qué. Le llegan señales —timing, autonomía aparente del movimiento, presencia o ausencia de un gesto suyo previo— y a partir de ellas su percepción decide, automáticamente, quién actuó. Una fila de una tabla que se reordena sola un segundo después de que el

usuario deje de teclear se lee como decisión del sistema, aunque nadie lo haya anunciado. La interfaz que no gobierna esas pistas deja la autoría librada a la interpretación espontánea —y a menudo errónea— del usuario.

PRINCIPIO

La pregunta de diseño no es dónde situar al sistema, sino qué pistas perceptivas asignan la autoría de cada cambio.

La autoría propia se siente, y se puede romper

Existe una sensación, normalmente silenciosa, de ser uno el autor de las propias acciones: el sentido de agencia. La investigación sobre cómo se forma —el modelo del comparador y sus revisiones posteriores— converge en una idea aprovechable: sentimos que una acción es nuestra cuando su resultado coincide con lo que anticipábamos; muevo el cursor y algo se mueve conmigo, cuando y como esperaba. Cuando esa correspondencia falla —cuando ocurre algo que no predije—, la sensación de autoría se debilita.

Cuentas más recientes del sentido de agencia (Synofzik, Vosgerau y Newen) matizan que esa sensación no se construye de una sola fuente, sino integrando varias pistas según su fiabilidad: la predicción sensoriomotora, sí, pero también el contexto, las claves previas a la acción y el resultado observado. Para una interfaz esto es una buena noticia: significa que la autoría es diseñable, porque se apoya en pistas que la interfaz controla.

Cuando el sistema actúa por el usuario, esa correspondencia se rompe por diseño: ocurre algo que el usuario no hizo. Ese es, exactamente, el momento que delegate debe hacer legible. Compárense dos correos:

Un asistente que mueve un correo a una carpeta sin más: el mensaje desaparece de la bandeja. El usuario, que no hizo ese gesto, percibe una

ausencia inexplicable. ¿Lo archivé yo y no recuerdo? ¿Un fallo? La autoría quedó muda, y con ella el control.

El mismo archivado, pero con una breve presencia que declara «Archivado por tu asistente · Deshacer»: el evento se atribuye a su actor, queda revisable y reversible. La acción fue la misma; lo que cambia es que la autoría se hizo perceptible.

DISTINCIÓN

Cuando el sistema actúa, no basta con mostrar qué cambió. Hay que hacer perceptible quién lo cambió.

Cuanto mejor funciona, menos se vigila

Hay una asimetría que delegate no puede ignorar. Parasuraman y Manzey, revisando décadas de estudios sobre uso de automatización, describieron lo que llamaron complacencia inducida por automatización: cuando un sistema automático funciona bien de forma sostenida, el operador le presta cada vez menos atención y tarda más en detectar sus fallos. No es pereza ni falta de formación —aparece igual en novatos y en expertos, y no se corrige con práctica simple—: es una consecuencia estructural de cómo repartimos una atención limitada. Confiamos en lo que rara vez falla, y dejamos de mirarlo.

El problema es que el error de un agente fiable llega justo cuando el usuario ha bajado la guardia. Y en la era de los agentes generativos, ese patrón se ha trasladado de las cabinas y las salas de control al trabajo de conocimiento cotidiano: el asistente que acierta noventa y nueve veces educa al usuario para no revisar la número cien.

De aquí sale una exigencia de diseño que distingue a delegate de las demás familias: la legibilidad de la acción delegada no puede decaer con la confianza. Un commit que el usuario hizo puede confirmarse y desvanecerse, porque él sabe que ocurrió. Una acción que hizo el sistema necesita dejar rastro revisable aunque —sobre todo porque— el usuario ya

no la esté vigilando. El historial de acciones del agente, la posibilidad de revisar en lote lo que hizo mientras el usuario no miraba, el deshacer que sobrevive al momento: no son cortesías, son lo que mantiene al usuario en posición de intervenir.

Cuándo el usuario confía de más o de menos —la calibración de la confianza— es un problema por derecho propio que excede esta gramática. Aquí basta retener que la interfaz es la principal herramienta para mantener esa confianza ajustada a lo que el sistema realmente hace.

De la psicología a la práctica

Las directrices de interacción humano-IA recogidas por Amershi y colaboradores formulan, desde la práctica, lo mismo que la percepción exige: dejar claro qué puede hacer el sistema y con qué fiabilidad, mostrar cuándo está actuando, permitir corregir y deshacer, y escalar al usuario cuando la confianza es baja. Traducidas a esta gramática, no son consejos sueltos: son afirmaciones sobre qué eventos de delegate deben hacerse perceptibles, y con qué intent. El resto de este capítulo las desarrolla como lo que son —eventos, canales y composiciones—, no como buenas intenciones.

§ 3

Qué es delegate

delegate es la familia de los eventos donde cambia quién lleva la iniciativa o quién ejecuta parte de la acción.

No es una familia de IA. Una IA puede producir eventos delegate, pero no todo evento delegate requiere IA. Una macro, un flujo automatizado, una regla programada, una aprobación delegada o un sistema de workflow también pueden producir delegate.

La pregunta de delegate es:

¿el usuario actúa directamente, el sistema actúa por él, o ambos comparten la acción?

§ 4

Verbos principales

Los verbos de *delegate* describen momentos distintos del reparto de control:

delegate.offer → el sistema ofrece encargarse de algo

delegate.plan → el sistema propone un plan o curso de acción

delegate.authorize → el usuario concede permiso o alcance

delegate.act → el sistema actúa por el usuario

delegate.review → el sistema espera revisión antes de aplicar

delegate.escalate → el sistema pide intervención humana

delegate.return → el control vuelve al usuario

Vistos sobre un caso —un asistente que ordena facturas en el correo—, dejan de ser abstractos:

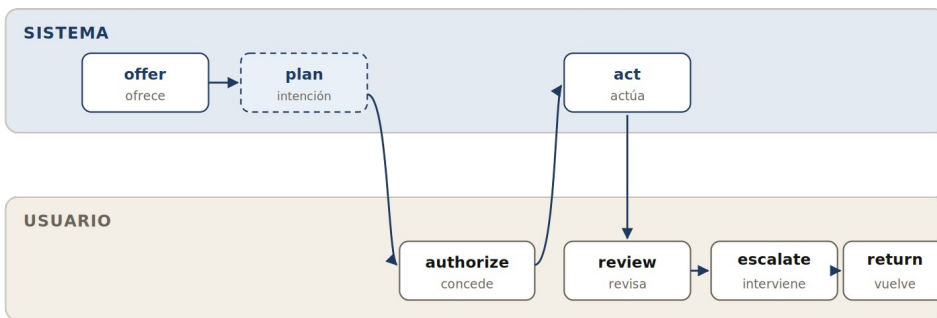
- *delegate.offer* → «He encontrado 3 facturas sin archivar. ¿Quieres que me encargue?»
- *delegate.plan* → «Voy a crear la carpeta *Facturas 2026* y mover las tres ahí.» (hace legible su intención antes de tocar nada)
- *delegate.authorize* → el usuario acepta ese plan concreto, no una autorización en abstracto
- *delegate.act* → el agente mueve las facturas
- *delegate.review* → «Hecho. ¿Reviso el resultado antes de vaciar la bandeja?»

- *delegate.escalate* → «Una factura tiene un importe anómalo; decide tú.»
- *delegate.return* → el control vuelve al usuario

Seis de estos verbos reparten el turno de actuar; *plan* hace otra cosa, y por eso conviene no confundirlo con los demás: no dice quién actúa, sino qué se va a hacer antes de hacerlo. Es el verbo de la legibilidad anticipada —lo que convierte una autorización en una decisión informada en lugar de un cheque en blanco—.

FIGURA 29.2

Los verbos de delegate: el reparto del turno



Seis verbos reparten el turno de actuar; plan hace algo distinto: vuelve visible la intención antes de actuar.

§ 5

Qué no es delegate

delegate no es sustain. sustain comunica que algo sigue ocurriendo. delegate comunica quién actúa o bajo qué autorización.

delegate no es signal. signal reclama atención. delegate puede pedir revisión, pero la reclamación de atención pertenece a signal.

delegate no es commit. commit comunica resultado. delegate puede terminar en commit, pero no es el resultado.

delegate no es shift. shift cambia el contexto. delegate cambia el reparto de acción dentro de ese contexto.

§ 6

delegate e intent

delegate no tiene intent por defecto. Que el sistema actúe por el usuario no significa automáticamente que algo sea positivo, negativo, urgente o perdido.

El intent aparece cuando la acción delegada produce una señal, un resultado, un riesgo, una amenaza o una pérdida.

Ejemplos:

delegate.act

→ sustain.processing

→ commit.apply + affirm

delegate.review

→ signal.warn + risk

delegate.act

→ commit.fail + risk

delegate.act

→ commit.delete + loss

§ 7

Canales de delegate

delegate casi no tiene canales propios. Los toma prestados de las familias que desencadena: la actividad se expresa como sustain, el resultado como commit, lo que aparece para revisión como emerge. Pero delegate añade sobre todos ellos una capa que ninguna otra familia necesita y que ningún canal sensorial transporta por sí solo: la atribución. Decir quién actúa no es cuestión de color ni de movimiento. Es, antes que nada, una cuestión de lenguaje.

Texto. Es el canal primario de delegate, más que en ninguna otra familia. La autoría y el alcance son proposicionales: «tu asistente», «3 correos», «puedes deshacerlo» son cosas que se afirman, no que se insinúan. Un icono indica que algo automático ocurre; solo el texto dice qué, de quién y hasta dónde. Por eso un evento de autorización —«Puedo enviar esta respuesta a 3 personas. ¿Autorizas?»— vive o muere en su redacción: el alcance que no se enuncia, no se percibe.

Presencia. Delegate necesita una presencia de dos tiempos. Una momentánea, que acompaña la acción mientras ocurre. Y otra persistente —un historial, un registro revisable— que sobrevive a la acción. Esta segunda no es un lujo: es la consecuencia directa del fundamento. Si la vigilancia del usuario decae con la confianza, el rastro de lo que el sistema hizo mientras nadie miraba es lo único que le devuelve el control. La presencia persistente de delegate es la memoria de la autoría.

Tiempo. El ritmo declara agencia. Una acción que se ejecuta en el mismo instante en que el usuario la roza se percibe como suya; la misma acción precedida de un breve compás —«Voy a archivar esto...»— se percibe como del sistema. Ese compás no es lentitud: es el espacio donde la autoría se vuelve legible y donde cabe la cancelación. La secuencia entera de delegate —ofrecer, actuar, devolver— es una figura temporal, y su legibilidad depende de que esos tres momentos no se colapsen en uno.

Motion. Secundario y delicado. El movimiento puede señalar el lugar donde el sistema está actuando —dónde mira, qué toca—, pero es justo el

canal que dispara la atribución de agencia espontánea: algo que se mueve solo se lee como decidido por alguien. Usado sin atribución explícita, motion produce la inquietud que delegate venía a resolver. Acompaña; no sustituye al texto.

Lo que delegate no debe usar para sí. Color, sonido y háptica pertenecen a las consecuencias, no al reparto de iniciativa. delegate no tiene intent por defecto; teñir de verde o de rojo el hecho de que el sistema actúe es cargar valencia donde todavía no la hay. Si la acción delegada produce un riesgo o una pérdida, son signal y commit —no delegate— quienes lo expresan, con sus canales.

PRINCIPIO

delegate toma prestados los canales de las familias que desencadena, y les añade una capa irreducible: la atribución. Un indicador de actividad que no dice quién actúa no es delegate; es solo sustain.

§ 8

Casos no basados en IA

Un sistema de aprobación puede delegar una acción en otro usuario o rol.

Una regla automática puede archivar mensajes bajo condiciones definidas.

Una macro puede aplicar una secuencia de cambios en una herramienta profesional.

Un workflow puede ejecutar pasos cuando una tarea cambia de estado.

En todos esos casos lo importante no es la inteligencia del sistema. Lo importante es que la acción ya no está completamente en manos del usuario en ese instante.

§ 9

Riesgos de diseño

delegate es una familia delicada porque afecta control y responsabilidad. Si se expresa mal, el usuario puede no saber si el sistema solo sugiere, si ya está actuando, si espera autorización o si el resultado ya se aplicó.

Los errores más comunes son:

mostrar un loader genérico cuando el sistema está actuando por el usuario;

no explicar el alcance de la acción;

no ofrecer salida o cancelación;

no diferenciar sugerencia, plan, autorización y ejecución;

no dejar claro cuándo el control vuelve al usuario.

§ 10

Accesibilidad de delegate

La accesibilidad de delegate sigue la regla del libro —cuando un canal falla, migra el significado, no el adorno—, pero parte de una desventaja propia. Buena parte de la atribución que un usuario vidente recibe gratis —el movimiento autónomo, el compás antes de actuar, el cambio que aparece «solo»— son pistas visuales y temporales que no tienen equivalente para quien no ve la pantalla. La agencia que la vista infiere, el resto de los sentidos necesita que se la declaren.

DISTINCIÓN

La pista de agencia que el usuario vidente percibe en el movimiento no existe sin la vista. Por eso, en delegate, la autoría no se muestra: se enuncia.

Esto convierte las cuatro preguntas de delegate en cuatro exigencias concretas.

Autoría. Que el sistema ha actuado debe anunciarse, no quedar confiado a un icono o a un color. Para quien navega sin ver, un cambio de estado

no atribuido es aún más desorientador: algo cambió en el documento y nada dice que no fue él. La atribución viaja en el texto accesible del evento, no en su apariencia.

Actividad. Que el sistema está actuando ahora se comunica mediante regiones activas, pero dosificadas: el antipatrón del anuncio incesante de sustain se aplica igual aquí. Se anuncia el inicio, los hitos relevantes y el cierre; no cada microestado.

Alcance. El punto de autorización debe ser alcanzable y operable sin vista antes de que la acción se consume. Una acción que se ejecuta «salvo que la detengas» dentro de una ventana de tiempo solo captable visualmente no es autorización: es una trampa para quien no llega a verla. El alcance se enuncia y la decisión se ofrece como un control real, no como un plazo.

Resultado y reversibilidad. El rastro revisable y el deshacer deben sobrevivir y ser alcanzables sin vista. Aquí el argumento de la confianza se duplica: si para el usuario vidente el registro es lo que le permite recuperar el control cuando ha dejado de vigilar, para el usuario no visual ese registro es, a menudo, la única vía de acceso a lo que ocurrió. No puede depender de una notificación efímera.

Hay además una dimensión cognitiva. *delegate* toca control y responsabilidad, y la automatización sin alcance claro es, en sí misma, una barrera: abruma y arrebató la sensación de control. Por eso *delegate.escalate* —el momento en que el sistema devuelve la decisión a la persona— tiene que ser perceptible y no quedar enterrado: pedir intervención humana solo sirve si la petición se percibe.

PRINCIPIO

En delegate, accesibilidad y legibilidad de la autoría son el mismo problema: hacer que quien no ve la pantalla sepa, en todo momento, quién acaba de actuar y qué puede hacer al respecto.

El agente que interpreta: una frontera

El fundamento de esta familia dejó planteada una pregunta —no solo «¿quién actúa?», sino «¿por cuenta de quién?»—. Una acción delegada tiene una dirección: el agente puede actuar por cuenta del sistema —una fila que se reordena sola— o por cuenta del usuario —un asistente que rellena un formulario en su nombre—. «El sistema hizo algo sobre mi espacio» y «mi agente hizo algo por mí» son atribuciones distintas, y delegate debe hacer perceptible de qué lado viene la iniciativa. La tensión no es nueva. Pattie Maes defendió que la sobrecarga vuelve inevitable delegar en agentes que actúan por el usuario; Ben Shneiderman, la manipulación directa que mantiene al usuario en control. Eric Horvitz mostró que la oposición era falsa: en la iniciativa mixta, usuario y agente negocian quién actúa en cada momento. La dirección de la acción es, precisamente, lo que esa negociación reparte.

Hay un punto en que esa dirección se complica, y marca la frontera de este libro. Cuando el agente actúa por cuenta del usuario, a menudo no se limita a actuar: lee el contexto por él y selecciona qué merece volverse perceptible. Un agente que compra por ti recorre las tres posiciones de un tirón. Lee el catálogo y descarta la mayoría sin mostrarla. Decide y actúa. Y al presentar «esto es lo que te conviene» entrega una vista que ha construido, no la tienda tal cual. La selección final la gramática ya sabe describirla: hay un commit, puede haber `delegate.act` o `delegate.review`. Lo que las familias todavía no saben nombrar es el paso previo: el recorte del campo. No hay una familia para «esto que ves es un subconjunto que alguien eligió por ti».

Y ahí aparece lo más difícil de hacer legible: lo que el agente descarta no se ve. La omisión es invisible por definición — un producto más barato que encajaba, una marca no considerada, un sesgo en «acorde a tus gustos». No es una hipótesis. Eli Pariser lo describió en 2011 como la burbuja de filtro: la personalización aísla al usuario de lo que no encaja con su perfil, y

sin mecanismos transparentes. Y los estudios de Motahhare Eslami y Karrie Karahalios sobre los muros de noticias encontraron algo aún más revelador: la mayoría de los usuarios ni siquiera sabía que había una curación operando.

Frente a esto, una línea de la investigación ofrece una dirección de diseño, y este libro la suscribe. Es lo que Mark Weiser llamó diseño con costuras —seamful, frente a seamless—: en lugar de ocultar las juntas del sistema, mostrarlas. Esconderlas deja al usuario incapaz de corregir los fallos de la máquina. Matthew Chalmers lo formuló para los sistemas ubicuos. Eslami demostró que exponer el razonamiento del algoritmo mejora la comprensión y a veces cambia la conducta. Y Nava Tintarev y Judith Masthoff nombraron la propiedad que aquí importa, la escrutabilidad: que el usuario pueda decirle al sistema que se ha equivocado. Trasladado a delegate, esto da un principio.

PRINCIPIO

El filtrado debe ser legible y reversible. Cuando un agente recorta el campo por el usuario, la interfaz debe hacer perceptible que hubo recorte y permitir ver lo que quedó fuera.

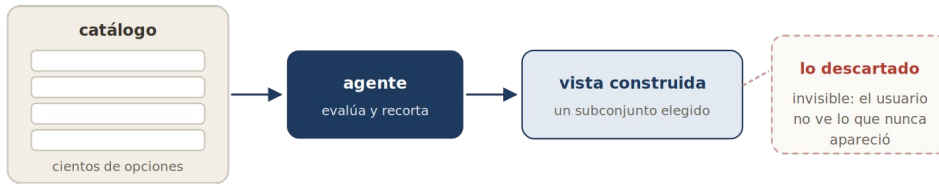
FIGURA 29.4

La dirección y la frontera del intérprete

¿POR CUENTA DE QUIÉN ACTÚA EL AGENTE?



LA FRONTERA · cuando el agente lee y filtra por el usuario



El filtrado debe ser legible y reversible:

la interfaz debe hacer perceptible que hubo recorte y permitir ver lo que quedó fuera.

Queda una pregunta que este libro no puede cerrar todavía, y prefiere dejarla abierta antes que fingir que la resuelve. Una omisión, ¿tiene intent? ¿Qué tono porta un evento que no aparece, algo que el agente decidió no mostrar? La investigación se ha acercado por dos lados sin llegar del todo. Bilgic y Mooney distinguieron en 2005 entre explicaciones que informan y explicaciones que promueven — el intent de la omisión en embrión, tratado como problema de honestidad, no de percepción. Y el trabajo sobre seamful XAI de Upol Ehsan y sus colaboradores introdujo la pregunta del «por qué no» como vía hacia la contestabilidad. Pero ninguno trata la omisión como un evento que el usuario deba percibir, con su propio tono. Esa es la frontera que esta gramática alcanza y no cruza: nombrar la omisión como evento es posible; asignarle un intent estable, todavía no.

Vale la pena decirlo sin rodeos: el intérprete da para mucho más de lo que cabe en una sección. Filtrado, personalización, curación, explicabilidad: un campo entero al que aquí solo nos asomamos para marcar dónde termina el actor. Tratarlo a fondo pediría su propia gramática — la de la omisión. Queda como semilla declarada, no como deuda saldada.

§ 12

Resumen operativo

Qué es: delegate nombra los eventos donde cambia quién lleva la iniciativa o quién ejecuta parte de la acción.

Qué no es: no es IA, no es proceso, no es señal y no es resultado.

Para qué sirve: sirve para que el usuario entienda si actúa él, el sistema o ambos, y bajo qué autorización.

Regla de uso: usa delegate cuando el problema principal sea quién actúa, no simplemente qué se muestra o qué resultado se produjo.

§ 13

Síntesis

delegate completa el sistema de familias al cubrir una región cada vez más importante: el reparto de acción entre usuario y sistema.

No aparece porque la IA esté de moda. Aparece porque muchas interfaces ya no solo esperan órdenes; también sugieren, planifican, ejecutan, escalan o devuelven control.

La gramática necesita hacer visible ese reparto.

Cierre de la Parte III

Con este capítulo se cierra la Parte III. Las ocho familias han quedado descritas una a una: qué pregunta perceptiva responde cada una, con qué verbos y canales se expresa, dónde están sus fronteras y cuáles son sus antipatrones capitales. El recorrido termina con delegate, la familia que la era de los agentes vuelve urgente: la que hace legible quién actúa.

Conocer las familias por separado no basta. Una interfaz real las compone: encadena contact con sustain, resuelve en commit, interrumpe con signal. La Parte IV convierte la gramática en herramienta: composición, matrices, rangos, accesibilidad aplicada, antipatronos, validación y, finalmente, sistema.

IV

Sistema

System

De la gramática a la práctica: composición, matrices, rangos, accesibilidad aplicada, antipatronos, validación y sistema de diseño.

30

Composición de eventos semánticos

Secuencias, reglas y solapamientos

Hasta ahora hemos separado. Hemos separado las familias semánticas — contact, commit, signal, handle, emerge, shift, sustain y delegate—, los intents —neutral, affirm, fulfill, risk, threat, loss— y los canales: tiempo, motion, presencia, profundidad, forma, color, sonido y háptica. Esa separación era necesaria: sin ella todo se mezcla, y un click parece éxito, un modal parece amenaza, un spinner parece error, una desaparición parece pérdida.

Pero una interfaz real no ocurre por piezas aisladas. El usuario no vive una familia semántica cada vez: vive flujos. Pulsa, espera, recibe una señal, corrige, confirma; el sistema procesa, algo queda fijado, algo se pierde, algo aparece como reparación, algo cambia de marco. Una interacción real suele ser una composición temporal de eventos semánticos, y la pregunta de este capítulo es cómo se construye esa composición sin perder claridad.

PREGUNTA DEL CAPÍTULO

¿Cómo se combinan las familias semánticas sin que la interfaz pierda claridad?

FIGURA 30.1

Una interacción es una composición temporal

guardar un documento remoto no es un solo evento, sino una secuencia



Una operación no se entiende por un único efecto, sino por la relación entre sus eventos.

§ 1

Por qué necesitamos composición

Una familia semántica aislada explica un tipo de evento, pero una operación de interfaz casi nunca es un solo evento. Guardar no es solo `commit.save`; eliminar no es solo `commit.delete`; arrastrar no es solo `handle.carry`; abrir un modal no es solo `shift.enter-mode`. Cada operación contiene fases. Guardar un documento remoto, por ejemplo, puede componerse así:

contact.press → *sustain.progress* → *commit.save + affirm*

La pulsación fue recibida, el sistema trabajó, el resultado quedó guardado. Si omitimos `contact`, el usuario duda de si actuó; si omitimos `sustain`, la espera parece bloqueo; si omitimos `commit`, no hay cierre. La operación no se entiende por un único efecto, sino por la relación entre eventos.

DISTINCIÓN

Una interacción real no es una familia aislada. Es una composición temporal de eventos.

§ 2

Secuencias básicas

Algunas composiciones son tan frecuentes que conviene tenerlas como repertorio de partida.

Acción inmediata con resultado simple

contact.press → commit.toggle + affirm

El usuario activa un switch y la lectura es doble y casi simultánea: «me ha sentido» y «el estado quedó aplicado». Cuando la operación es inmediata no hace falta *sustain*: introducir una espera visible donde no hay espera real solo añade ruido.

Acción con proceso intermedio

contact.press → sustain.progress → commit.save + affirm

El usuario guarda cambios en un servidor. La lectura se despliega en tres tiempos: «me ha sentido», «sigue trabajando», «quedó guardado». Cada evento responde una duda distinta, y por eso ninguno puede sustituir a los otros.

Acción con culminación

contact.press → sustain.progress → commit.complete + fulfill

El usuario exporta un vídeo, sube un archivo o completa un flujo importante. Aquí *fulfill* tiene más sentido que *affirm*, porque no se aplica simplemente un estado: se resuelve una tensión que venía acumulándose durante la espera.

Acción destructiva con advertencia

contact.press → signal.warn + threat → commit.delete + loss

El usuario elimina un proyecto. La amenaza vive antes de la consecuencia —todavía puede evitarse—; la pérdida vive después, cuando ya solo cabe registrarla. Si la secuencia respeta ese orden, cada intent aparece en el único momento en el que tiene sentido.

Eliminación con reparación

signal.warn + threat → commit.delete + loss → emerge.present undo

El usuario confirma la eliminación de una tarjeta y aparece una barra de deshacer. El undo no borra la pérdida: la recontextualiza como parcialmente reversible. Por eso su aparición debe leerse como reparación disponible, no como una segunda alarma.

Manipulación directa

contact.press → handle.pick → handle.carry → handle.drop → commit.reorder + affirm

El usuario reordena una lista. La lectura encadena cinco momentos: «me ha sentido», «lo agarro», «lo controlo», «lo suelto», «quedó reordenado». La evaluación se concentra al final; durante el trayecto, lo que importa es la sensación de control.

§ 3

Las reglas no son dogmas

Las reglas de composición que siguen no deberían parecer mandamientos arbitrarios. No decimos que *contact* va antes de *commit* porque sí: decimos que, si el usuario actúa, necesita percibir que su acción fue recibida antes de poder interpretar el resultado. No decimos que *threat* debe ir antes de *loss* porque lo dicte la taxonomía: decimos que una

amenaza convoca acción antes de la consecuencia, mientras que una pérdida registra la consecuencia después. No decimos que todo sustain debe terminar en commit por afán de simetría: decimos que un proceso abierto necesita algún tipo de cierre —resultado, cancelación, fallo o persistencia explicada—.

Cada regla, en definitiva, debe poder derivarse de una necesidad perceptiva. Cuando no puede, no es una regla de la gramática: es una costumbre.

§ 4

Reglas de composición

Regla 1 — Contact precede al resultado cuando hay acción directa

Si el usuario actúa directamente, la interfaz debe comunicar recepción antes de comunicar consecuencia: *contact.press* → *commit.save* + *affirm*. El fundamento es la agencia y la causalidad acción-respuesta. El error correspondiente es un *commit.save* + *affirm* sin contacto perceptible: el usuario no sabe si su acción fue registrada o si el estado cambió por otra razón.

Regla 2 — Sustain precede al commit cuando hay espera

Si la operación no termina de inmediato, debe sostenerse antes de resolverse: *contact.press* → *sustain.progress* → *commit.complete* + *fulfill*. El fundamento es la continuidad temporal y el mantenimiento de la confianza. El error es el silencio intermedio —contact, nada, resultado—: la operación puede funcionar técnicamente, pero la espera se percibe como bloqueo.

Regla 3 — Threat precede a loss

Threat vive antes o durante una posible consecuencia; loss vive después: *signal.warn + threat → commit.delete + loss*. El fundamento quedó argumentado en el Capítulo 10: prevenir y registrar son actos distintos. El error es *commit.delete + threat*: amenaza cuando ya no hay nada que evitar.

Regla 4 — Emerge no debe absorber el intent del contenido

Una superficie puede aparecer sin ser evaluable: *emerge.open → signal.warn + risk*. El fundamento es la distinción figura/fondo entre evento estructural y evento evaluable: el marco no es el mensaje. El error es *emerge.open + risk*: la apertura no es el riesgo; el riesgo está en lo que aparece dentro. La excepción, heredada del Capítulo 9, es la aparición-que-advierte: cuando la única razón de la superficie es hacer perceptible la situación evaluable, puede portar el tono — declarado y justificado.

Regla 5 — Shift debe orientar el cambio de contexto

Cuando cambia el contexto, el usuario necesita orientación: *shift.enter-mode* debe acompañarse de foco, título, profundidad, estructura y salida clara. El fundamento es el cambio de modelo operativo: al cruzar un marco, las reglas de acción son otras. El error típico es el modal que aparece visualmente mientras el foco permanece detrás.

Regla 6 — Handle concentra la evaluación en el drop

Durante *handle.carry* el usuario está controlando; la evaluación aparece sobre todo al soltar: *handle.pick → handle.carry → handle.drop + risk*. El fundamento es el control motor continuo y la economía atencional. El error es teñir todo el arrastre de amenaza porque el usuario pasa cerca de una zona destructiva; mejor reservar *signal.warn + threat* para cuando esa zona se vuelve destino probable.

Regla 7 — Todo proceso abierto necesita salida

Un sustain no puede quedar suspendido indefinidamente. Sus salidas posibles son conocidas: `commit.complete + fulfill`, `commit.save + affirm`, `commit.fail + risk`, `signal.warn + risk` o `commit.cancel + neutral`. El fundamento es el cierre cognitivo: la memoria necesita saber cómo terminó lo que estaba abierto. El error tiene nombre propio: el spinner eterno.

El orden dentro del evento: señal, mutación y ventana

Las siete reglas ordenan eventos entre sí. Queda un orden más fino, dentro de cada evento: el de su señal perceptiva respecto de la mutación que la acompaña. Ese orden tiene tres valores. *Anticipado*: la señal precede a la mutación — la retirada de una superficie se anuncia antes de desaparecer, porque después ya no habría dónde percibirla. *Posterior*: la mutación se asienta primero y la señal la sigue — el resultado de un gesto se fija y luego se celebra; una superficie que aparece primero existe y luego se siente. *Indivisible*: señal y mutación son un solo momento — la manipulación continua, donde el usuario es el valor cambiando.

Elegir mal el orden produce dos fallos simétricos: anticipar la señal de una aparición retrasa la llegada de la superficie hasta que la señal termina; posponer la señal de una retirada la deja sin lugar donde ocurrir. Y un principio engloba los tres valores: la señal debe completar su ventana perceptiva antes de que el cambio estructural la pise. Un evento cuyo estado lo atropella a mitad de vida no llega a leerse.

§ 5

Ejemplo largo: subida con fallo parcial y recuperación

Veamos una composición más compleja. El usuario sube un archivo grande para enviarlo a revisión; durante la subida la conexión falla

parcialmente; el sistema conserva parte del progreso, muestra una advertencia y permite reintentar; el usuario reintenta y la subida termina correctamente.

Acción inicial

contact.press — el usuario pulsa «Subir». Canales: respuesta inmediata, estado pressed, micro motion opcional.

Subida en curso

sustain.upload — el sistema empieza a subir. Canales: barra de progreso, texto «Subiendo archivo...», presencia persistente, baja saliencia.

Advertencia corregible

signal.warn + *risk* — la conexión se interrumpe parcialmente. No es threat, porque no hay urgencia inmediata; no es loss, porque nada se perdió definitivamente. Canales: callout, color moderado de advertencia, texto descriptivo, presencia persistente.

Reintento automático o disponible

sustain.retrying — el sistema intenta continuar. Canales: indicador de reintento, texto «Reintentando...», baja saliencia.

Resultado parcial

commit.partial + *risk* — parte de la operación no se completó. Este commit no es fulfill ni loss completo: es un resultado problemático pero corregible. Canales: estado de progreso parcial, mensaje descriptivo, color de advertencia.

Aparición de recuperación

emerge.present recovery — aparece la opción de reintentar o continuar. La aparición no es el riesgo: la opción emergente ofrece reparación. Canales: botón de reintentar, presencia clara.

Reintento del usuario

contact.press → *sustain.progress* — el usuario pulsa «Reintentar» y el sistema vuelve a trabajar. Canales: barra de progreso restaurada, texto «Subiendo...».

Cierre positivo

commit.complete + *fulfill* — la subida termina. Aquí *fulfill* tiene sentido porque hubo tensión, espera y recuperación. Canales: motion resolutivo, marca de completado, color positivo, sonido breve opcional.

§ 6

Eventos simultáneos y solapados

Hasta ahora hemos hablado de secuencias lineales, pero una interfaz real también tiene simultaneidad. Un proceso puede seguir en curso mientras aparece una advertencia (*sustain.progress* + *signal.warn* + *risk*); un usuario puede estar arrastrando mientras se aproxima a una zona destructiva (*handle.carry* + *signal.warn* + *threat*); un modal puede cambiar el marco mientras dentro aparece una señal (*shift.enter-mode* + *signal.alert* + *threat*); un sistema puede estar sincronizando mientras muestra un estado parcial (*sustain.syncing* + *commit.partial* + *risk*); una notificación puede llegar mientras el usuario manipula un objeto (*handle.carry* + *signal.notify* + *neutral*).

La composición, por tanto, no siempre es una cadena. A veces es una superposición, y las superposiciones plantean una pregunta que las secuencias no tenían: ¿quién manda?

§ 7

Regla de dominancia atencional

Cuando dos eventos se solapan, uno debe dominar la atención y el otro debe mantenerse como contexto. La jerarquía de dominancia sigue tres criterios.

Primero, el evento evaluable domina sobre el estructural: un *signal.warn + risk* que aparece durante un *sustain.progress* debe convertirse en figura, y el proceso queda como contexto. Segundo, el evento con mayor activación domina sobre el de menor activación: un *signal.alert + threat* pasa por delante de un *signal.notify + neutral*, porque la amenaza reclama atención y la notificación puede esperar. Tercero, entre eventos del mismo nivel domina el más reciente: si durante un *sustain.progress* aparece un *commit.partial + risk*, el resultado parcial toma prioridad sobre el indicador de proceso.

Los tres criterios se ven bien sobre casos. En *sustain.progress + signal.warn + risk*, el proceso sigue pero la advertencia debe ser figura; la lectura correcta es «el sistema sigue trabajando, pero hay algo que revisar», no «el proceso falló». En *handle.carry + signal.warn + threat*, el usuario sigue manipulando pero la zona destructiva reclama atención: «sigo controlando el objeto, pero si lo suelto ahí habrá consecuencia». Y en *handle.carry + signal.notify + neutral* domina *handle*, porque el usuario está en control motor activo: la notificación no debe interrumpir ni reclamar foco; espera, o aparece sin competir.

§ 8

Jerarquía dentro de una composición

Una composición necesita jerarquía: no todos los eventos deben expresarse con la misma intensidad. En una operación larga — *contact.press* → *sustain.progress* → *commit.complete + fulfill*—, *contact*

debe ser inmediato y discreto, sustain debe sostener sin fatigar y commit debe cerrar con claridad. Si los tres tienen la misma energía, la secuencia se vuelve plana. En una operación destructiva —*signal.warn* + *threat* → *commit.delete* + *loss* → *emerge.present undo*—, *threat* debe destacar antes, *loss* debe registrar después y el *undo* debe permanecer como reparación, sin parecer otra amenaza.

La graduación sigue tres principios. El evento evaluable recibe más intensidad que el estructural: *commit.complete* + *fulfill* merece más intensidad que el *sustain* que lo precedió. El evento con mayor activación recibe más intensidad que el de menor: un *threat* necesita más saliencia que un *affirm*, un *fulfill* más resolución que un *neutral*, un *loss* más gravedad que un *risk*. Y el evento de cierre recibe más presencia que los intermedios: en *contact* → *sustain* → *commit*, el *commit* es el momento perceptivo más claro de la secuencia; el *sustain* acompaña y el *contact* inicia.

Cuando todos los eventos compiten por la misma atención, la composición se convierte en ruido. Cuando están bien graduados, la secuencia se lee como una historia perceptiva coherente.

§ 9

Antipatrones de composición

Commit anticipado. La interfaz muestra éxito antes de terminar: *contact.press* → *commit.complete* + *fulfill* → *sustain.progress*. Promete un resultado que aún no existe. Corrección: devolver el *commit* al final de la cadena.

Threat después de loss. *commit.delete* + *loss* → *signal.alert* + *threat*. Comunica urgencia cuando ya no hay prevención posible. Corrección: la advertencia va antes de la consecuencia.

Emerge como alert. Un dropdown aparece con la energía de una alerta, y la aparición local se confunde con una señal urgente. Corrección: `emerge.open` sin `intent`, salvo que lo que aparece contenga un `signal`.

Sustain sin salida. Un spinner queda indefinido: proceso sin cierre. Corrección: todo `sustain` desemboca en `commit.complete`, `commit.fail`, `signal.warn` o cancelación.

Handle evaluado todo el tiempo. Todo el arrastre está teñido de amenaza y el usuario pierde la sensación de control. Corrección: mantener el `carry` neutro y señalar solo destinos o `drop`.

Notificación que interrumpe la manipulación. Una notificación reclama foco mientras el usuario arrastra, rompiendo el control motor activo. Corrección: la notificación espera, o aparece sin competir con `handle`.

§ 10

Síntesis

La composición convierte familias aisladas en interacción real. Una interfaz no debe preguntarse solo «¿qué evento es este?»; debe preguntarse también qué ocurrió antes, qué viene después, qué evento domina la atención, cuál queda como contexto, qué resultado cierra la secuencia y qué huella queda.

PRINCIPIO

Toda composición debe preservar la pregunta perceptiva de cada evento sin hacerlos competir innecesariamente.

La composición es donde la gramática deja de ser vocabulario y empieza a comportarse como lenguaje.

FIGURA 30.2

Seis secuencias frecuentes

INMEDIATA



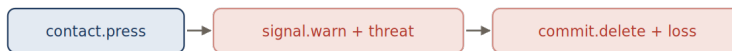
CON PROCESO



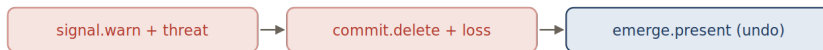
CON CULMINACIÓN



DESTRUCTIVA CON AVISO



CON REPARACIÓN



MANIPULACIÓN DIRECTA



La amenaza vive antes de la consecuencia; la pérdida, después. El orden de los eventos es parte del significado.

FIGURA 30.3

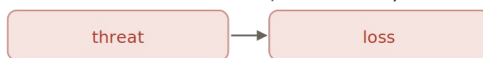
Tres reglas de orden en la composición

1 · CONTACT PRECEDE AL RESULTADO



primero «me ha sentido», luego el resultado

2 · LA AMENAZA VIVE ANTES; LA PÉRDIDA, DESPUÉS



antes de la consecuencia / después de ella

3 · EL PROCESO NO ES EL RESULTADO



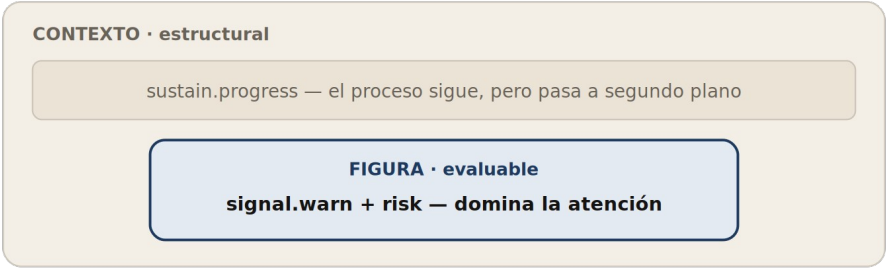
la espera no debe celebrarse como cierre

Las reglas no son dogmas, pero el orden temporal de los eventos transporta significado.

FIGURA 30.4

Dominancia atencional en el solapamiento

cuando dos eventos coinciden, uno es figura y el otro queda como contexto



El evento evaluable domina sobre el estructural: el aviso se vuelve figura; el proceso, contexto.

TABLA 30.1 Operaciones frecuentes y su secuencia semántica

OPERACIÓN	SECUENCIA SEMÁNTICA	LECTURA
Activar switch	contact.press → commit.toggle + affirm	acción recibida, estado aplicado
Guardar remoto	contact.press → sustain.progress → commit.save + affirm	acción, proceso, cierre
Completar flujo	contact.press → sustain.progress → commit.complete + fulfill	acción, espera, objetivo cumplido
Eliminar	contact.press → signal.warn + threat → commit.delete + loss	acción, advertencia, pérdida
Deshacer eliminación	commit.delete + loss → emerge.present undo	pérdida con reparación temporal
Reordenar	contact.press → handle → commit.reorder + affirm	agarre, control, resultado

TABLA 30.2 Las siete reglas de composición y su fundamento

REGLA	FUNDAMENTO
contact precede a resultado	agencia y causalidad
sustain precede a commit si hay espera	continuidad temporal

threat precede a loss	antes/después de consecuencia
emerge no absorbe intent	figura/fondo; marco ≠ mensaje
shift debe orientar	cambio de modelo operativo
handle.drop concentra evaluación	control continuo vs resultado
sustain necesita salida	cierre cognitivo

TABLA 30.3 El ejemplo largo, momento a momento

MOMENTO	FAMILIA	INTENT	LECTURA
Pulsar subir	contact.press	—	el sistema me ha sentido
Subiendo	sustain.upload	—	sigue trabajando
Problema de red	signal.warn	risk	hay algo corregible
Reintento	sustain.retrying	—	intenta recuperarse
Resultado parcial	commit.partial	risk	algo quedó incompleto
Opción de recuperación	emerge.present recovery	—	aparece una reparación
Reintentar	contact.press	—	recibió mi acción
Subida completada	commit.complete	fulfill	objetivo cumplido

TABLA 30.4 Criterios de dominancia atencional

CRITERIO	REGLA
Evaluabilidad	lo evaluable domina sobre lo estructural
Activación	lo más activado domina sobre lo menos activado
Recencia	si el nivel es similar, domina lo más reciente

TABLA 30.5 Composiciones solapadas: contexto y figura

COMPOSICIÓN	EVENTO DE CONTEXTO	EVENTO DOMINANTE
sustain.progress + signal.warn	proceso en curso	advertencia
handle.carry + signal.threat	manipulación	zona peligrosa
shift.enter-mode + signal.alert	nuevo marco	alerta interna
sustain.syncing +	sincronización	resultado parcial

commit.partial		
emerge.present + signal.notify	aparición	mensaje del sistema
handle.carry + signal.notify	manipulación activa	handle domina, notificación espera

31

Matrices semánticas

Familias, intents y canales

El capítulo anterior mostró que una interacción real no suele ser una familia aislada. Una operación de interfaz encadena eventos: contacto, proceso, señal, consecuencia, aparición, cambio de contexto o manipulación. La gramática se vuelve útil cuando permite leer esas secuencias sin confundirlas.

Pero para diseñar, revisar o documentar una interfaz hace falta otro paso. No basta con saber que existen las ocho familias y los seis intents: hay que saber qué canales expresan mejor cada familia y cómo cambia esa expresión cuando aparece un intent. Este capítulo introduce dos herramientas para eso —la matriz de familias $\times$ canales y la matriz de intents $\times$ canales—. No son tablas cerradas ni recetas universales: son mapas de decisión.

PREGUNTA DEL CAPÍTULO

¿Qué canal debe dominar cada evento y qué función cumple en él?

§ 1

Para qué sirve una matriz semántica

Una matriz semántica convierte la gramática en una herramienta de trabajo. Permite pasar de frases generales como «este evento necesita feedback» a preguntas más precisas: ¿es contact o commit? ¿Hay proceso intermedio? ¿Hay intent evaluativo? ¿El canal dominante debe ser tiempo, motion, forma, presencia o sonido? ¿Qué ocurre si el usuario reduce motion? ¿Qué queda después del evento?

Un botón «Guardar», por ejemplo, no debería resolverse diciendo «añade una animación de success», sino leyendo la secuencia: contact.press, sustain.progress si hay espera, commit.save + affirm. Y entonces la matriz ayuda a decidir: contact.press necesita tiempo y micro motion; sustain.progress necesita presencia, texto y progreso; commit.save + affirm necesita marca discreta, quizá color suave, quizá sonido opcional. La matriz obliga a separar lo que muchas veces se decide junto.

PRINCIPIO

Una matriz semántica no dice qué efecto usar. Dice qué función debe cumplir cada canal dentro del evento.

§ 2

Cómo leer las matrices

Las matrices de este capítulo no usan solo valores como «alto», «medio» o «bajo». Esa forma puede ser útil, pero es insuficiente: decir que motion es «alto» en contact no explica qué hace motion ahí. Es mejor decir «motion en contact → causalidad inmediata». Decir que color es «alto» en signal tampoco basta; es mejor decir «color en signal → saliencia y valencia». Por eso las celdas describen función, no solo intensidad, y la pregunta para cada celda es siempre la misma: ¿qué aporta este canal a esta familia o a este intent?

§ 3

Matriz de familias y canales

La primera matriz cruza las ocho familias semánticas con los canales de expresión principales. No pretende decir que todos los eventos de una familia usen siempre los mismos canales; pretende mostrar qué canales suelen dominar y qué función suelen cumplir.

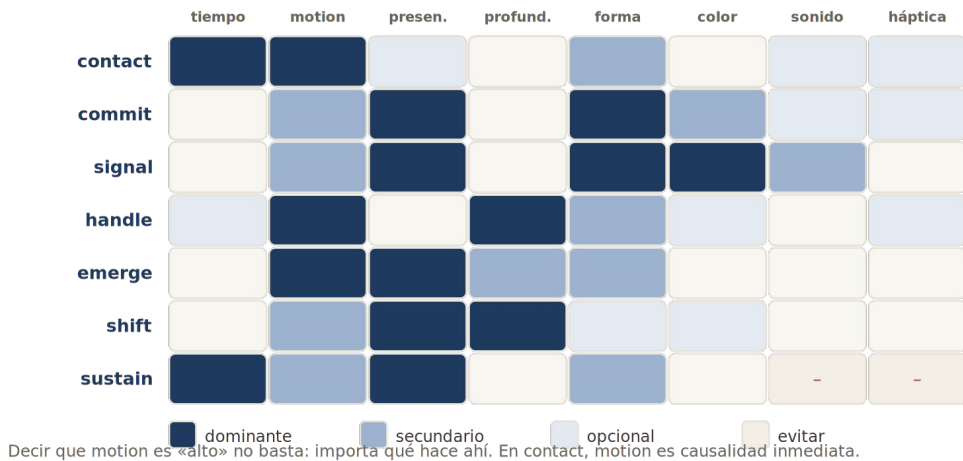
TABLA 31.1 Matriz familias × canales

FAMILIA	TIEMPO	MOTION	PRESENCIA / PROFUNDIDAD	FORMA	COLOR	SONIDO	HÁPTICA
contact	causalidad inmediata	microrespuesta al gesto	casi nula	estado pressed / foco	secundario	click opcional	pulso breve
commit	momento de cierre	asentamiento / retirada	huella o consecuencia	marca persistente	refuerzo evaluativo	cierre opcional	impacto / encaje
signal	persistencia suficiente	entrada o énfasis	figura atencional	callout / badge / alerta	saliencia y valencia	fuera de foco	alerta corporal
handle	baja latencia	seguimiento continuo	elevación / plano activo	grip / zona destino	válido / inválido	pick/drop opcional	pick/drop
emerge	entrada y salida	origen / retirada	presencia local	superficie / contenedor	apoyo menor	normalmente no	normalmente no
shift	transición orientada	cruce de marco	plano dominante / fondo	estructura del nuevo contexto	modo o estado	raro	raro
sustain	continuidad	actividad o progreso	estado persistente	barra / skeleton / indicador	bajo	evitar loop	evitar

FIGURA 31.1

Matriz de familias y canales

qué canal hace el trabajo principal en cada familia



§ 4

Lectura por familia

Contact

contact depende principalmente del tiempo: si la respuesta llega tarde, pierde su función aunque el motion sea bonito. El canal dominante es la latencia —el usuario debe percibir que el sistema ha sentido su acción—; motion puede ayudar con una microcompresión, la forma puede reforzar con un estado pressed, y sonido y háptica son opcionales. La función dominante es tiempo + motion → agencia inmediata, y el riesgo, convertir cada contacto en espectáculo.

Commit

commit necesita dejar huella. No basta con que algo cambie por un instante: el usuario debe percibir que algo quedó fijado, resuelto, eliminado, cancelado o restaurado. Motion puede comunicar asentamiento

o retirada; forma y presencia sostienen el estado; el color puede reforzar el intent; sonido o háptica pueden marcar cierre en eventos relevantes. La función dominante es forma + presencia + motion → consecuencia perceptible, y los riesgos, el commit efímero sin huella y la sobrecelebración de lo rutinario.

Signal

signal necesita entrar en la atención. Sus canales fuertes son presencia, forma, color y texto; el sonido es útil cuando la señal debe alcanzar al usuario fuera del foco visual; motion puede introducir o enfatizar, pero dosificado. La función dominante es presencia + forma + texto + color → atención orientada, y los riesgos, depender solo del color o convertir todo en alerta.

Handle

handle requiere baja latencia, continuidad y profundidad. Motion es central porque sostiene el control; la profundidad indica que el objeto salió del flujo normal; la forma define objeto, agarre y zonas de destino; el color puede indicar válido o inválido, pero no debe dominar el carry. La función dominante es motion + profundidad + forma → control directo, y los riesgos, perder agencia por lag, destino ambiguo o falta de alternativa accesible.

Emerge

emerge necesita hacer legible la entrada o salida de algo en el campo perceptivo. Presencia y motion son centrales; la profundidad ayuda si la superficie flota; la forma define el tipo de superficie; color, sonido y háptica suelen ser secundarios. La función dominante es presencia + motion + forma → aparición reconocible, y el riesgo, que una aparición local se sienta como alerta o como cambio de contexto.

Shift

shift necesita orientar un cambio de contexto. Sus canales principales son profundidad, presencia, foco, estructura y motion; el color puede reforzar el modo activo; sonido y háptica son raros. La función dominante es profundidad + foco + estructura → cambio de contexto, y el riesgo, que el usuario no sepa dónde está, qué cambió o cómo volver.

Sustain

sustain necesita sostener continuidad sin fatigar. El canal principal es el tiempo; la presencia mantiene visible el estado; la forma representa el proceso; motion puede indicar actividad, pero con baja demanda; sonido y háptica deben evitarse. La función dominante es tiempo + presencia + forma → proceso vivo, y los riesgos, el spinner eterno, la espera silenciosa o el sustain que parece alarma.

Delegate

delegate no tiene canales propios: toma prestados los de las familias que desencadena y les añade una capa de autoría. Sus canales característicos son el texto y la forma de atribución —una marca de «automático», una mención del agente—, una presencia suficiente para que la acción del sistema no pase inadvertida y una saliencia moderada que no la convierta en alarma. La función dominante es forma + texto + presencia → autoría legible, y los riesgos, que las acciones del sistema parezcan del usuario o que ocurran sin dejar rastro.

§ 5

Matriz de intents y canales

La segunda matriz cruza los intents con los canales. Aquí la pregunta cambia: ya no preguntamos qué familia es, sino cómo debe evaluarse

afectivamente el evento. Los intents no definen el evento; lo modulan. Por eso esta matriz no se usa sola: siempre debe leerse junto a una familia.

TABLA 31.2 Matriz intents × canales

INTENT	TIEMPO	MOTION	PRESENCIA	FORMA	COLOR	SONIDO	HÁPTICA
neutral	funcional	mínimo	baja	estructura clara	sin carga	normalmente no	normalmente no
affirm	breve	asentamiento suave	baja / discreta	marca ligera	positivo discreto	breve opcional	pulso suave
fulfill	cierre más marcado	resolución visible	media	final claro	positivo claro	ascendente breve	pulso definido
risk	persistente	tensión moderada	local y estable	callout / borde / icono	advertencia moderada	seco contenido	pulso contenido
threat	entrada rápida	saliente / tenso	dominante	bloque crítico	alta saliencia	urgente opcional	vibración fuerte
loss	breve + huella	retirada / descenso	ausencia visible	undo / registro / tachado	grave / desaturado	seco / grave	pulso seco

§ 6

Lectura por intent

neutral es ausencia de juicio afectivo fuerte, no ausencia de estructura. Puede necesitar forma y texto, pero no carga intensa de color, sonido o motion. Su riesgo es hacer invisible lo que necesita ser notado.

affirm confirma suavemente, con baja activación positiva: marca ligera, color discreto, asentamiento breve. No dice «objetivo logrado»; dice «todo va bien». Su riesgo es celebrar lo rutinario.

fulfill es resolución positiva, con mayor activación que affirm porque cierra una tensión: motion resolutivo, color positivo visible, sonido

ascendente opcional. Sus riesgos son trivializar el logro o convertirlo en espectáculo.

risk señala un problema corregible, en la región negativa moderada: callout local, color de advertencia, icono, texto, persistencia hasta la corrección. Dice «revisa esto», no «actúa ahora». Su riesgo es expresarse como amenaza.

threat es amenaza activa, de alta activación negativa: presencia dominante, alta saliencia, forma crítica, sonido opcional. Dice «atiende ahora». Su riesgo es usarse para todo y destruir su fuerza.

loss es pérdida consumada: comparte valencia negativa con threat, pero con menor activación y otra temporalidad. Se expresa con retirada, ausencia, huella, undo y color grave; dice «esto ya se perdió», sin alarma sostenida. Su riesgo es sonar como amenaza cuando ya no hay prevención posible.

§ 7

Cómo usar las matrices juntas

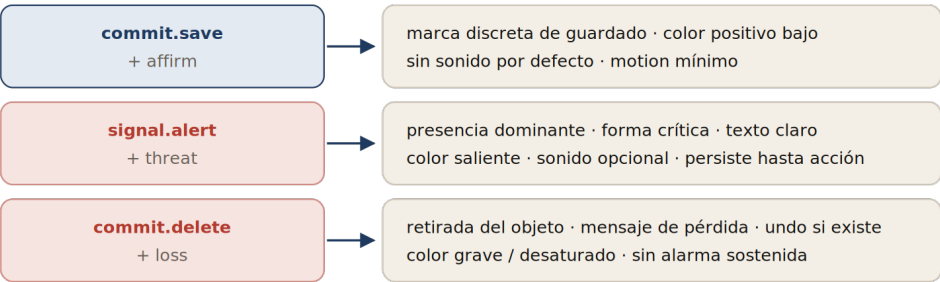
Las dos matrices deben cruzarse: no basta con elegir familia ni basta con elegir intent; la lectura final surge de la combinación. En `commit.save + affirm`, la familia pide huella, cierre y estado persistente, y el intent pide baja activación y confirmación suave; el resultado es una marca discreta de guardado, color positivo bajo, sin sonido por defecto, motion mínimo.

En `signal.alert + threat`, la familia pide atención, presencia, texto y forma clara, y el intent pide alta activación y acción inmediata; el resultado es presencia dominante, forma crítica, texto claro, color saliente, sonido opcional y persistencia hasta la acción.

FIGURA 31.2

Cómo usar las matrices juntas

la lectura final surge de cruzar familia (qué evento) e intent (cómo se evalúa)



La familia pide huella, atención o consecuencia; el intent decide la energía. El resultado es la combinación.

En **commit.delete + loss**, la familia pide consecuencia, huella y cierre, y el intent pide gravedad con activación contenida; el resultado es la retirada del objeto, un mensaje de pérdida, undo si existe, color grave o desaturado, y ninguna alarma sostenida.

§ 8

Celdas prohibidas o raras

No todas las combinaciones tienen el mismo sentido, y algunas casi nunca lo tienen. **emerge.open + threat** es normalmente incorrecta: la amenaza está en el contenido o en la señal, no en la apertura; mejor **emerge.open → signal.warn + threat**, con una excepción estrecha: la aparición que es la única advertencia (Capítulo 9). **sustain.progress + fulfill** confunde el proceso con el resultado; mejor **sustain.progress → commit.complete + fulfill**. **handle.carry + loss** adelanta una pérdida que solo ocurre en el resultado; mejor **handle.drop → commit.delete + loss**. Y **contact.press + fulfill** convierte el contacto en culminación; mejor **contact.press → sustain.progress → commit.complete + fulfill**.

TABLA 31.3 Combinaciones problemáticas y alternativas

COMBINACIÓN	PROBLEMA	ALTERNATIVA
emerge.open + threat	evalúa el marco, no el contenido	emerge.open → signal.warn + threat
sustain.progress + fulfill	confunde proceso y resultado	sustain.progress → commit.complete + fulfill
contact.press + loss	confunde recepción y consecuencia	contact.press → commit.delete + loss
handle.carry + loss	evalúa antes del drop	handle.drop → commit.delete + loss
shift.enter-mode + risk	evalúa el cambio de contexto	shift.enter-mode → signal.warn + risk

§ 9

Matrices y accesibilidad

Las matrices también sirven para revisar accesibilidad: si un evento depende de un solo canal, la matriz lo revela. Un signal.warn + risk que solo se expresa con color amarillo está diciendo que le faltan forma, texto, persistencia, icono y relación con el campo. Un commit.select + affirm que solo se expresa con verde carece de check, borde, estado persistente y forma seleccionada. Un sustain.progress que solo es un spinner necesita texto, estado, quizá progreso real y anuncio controlado.

La matriz obliga a preguntar, canal por canal: ¿cuál domina? ¿Cuál respalda? ¿Cuál falla? ¿Cuál sobra?

§ 10

Antipatrones de matriz

Usar todos los canales porque la matriz los menciona. La matriz no obliga a activar todo: muestra posibilidades y funciones, y también sirve para quitar. Si un commit.save + affirm ya tiene marca visual persistente y color positivo, añadir sonido, háptica y motion resolutivo probablemente

sobra. La pregunta no es solo «¿qué canal puedo añadir?», sino «¿qué canal puedo omitir sin perder la lectura?».

Interpretar la matriz como receta fija. Cada contexto modifica la decisión: una app médica, una herramienta creativa y un dashboard financiero no deberían modular igual.

Confundir canal dominante con canal único. Dominante no significa exclusivo: el canal que lidera necesita respaldo de al menos otro.

Usar intent sin familia. risk, por sí solo, no dice qué ocurrió. Hace falta la familia que lo ancle: signal.warn + risk, commit.fail + risk o handle.drop + risk.

Usar familia sin canal suficiente. Un commit.delete + loss sin huella, sin texto y sin ausencia significativa puede sencillamente no entenderse.

§ 11

Síntesis

Las matrices semánticas convierten la gramática en herramienta. La de familias responde a qué tipo de evento es y qué canales suelen hacerlo legible; la de intents, a cómo debe evaluarse y qué modulación perceptiva necesita. Usadas juntas permiten pasar de una decisión vaga —«pon feedback de success»— a una decisión semántica: commit.save + affirm → huella discreta → color positivo leve → sin sonido por defecto → persistencia breve.

PRINCIPIO

Una matriz semántica no prescribe efectos. Organiza funciones perceptivas.

PRINCIPIO

No conviene preguntar primero qué canal usar, sino qué evento debe percibirse y qué función debe cumplir cada canal.

32

Rangos psicoperceptivos orientativos

Duración, intensidad, persistencia y fatiga

El capítulo anterior convirtió las familias, los intents y los canales en matrices de decisión. Pero una matriz todavía deja abierta una pregunta práctica:

PREGUNTA DEL CAPÍTULO

¿Cuánto debe durar, persistir o intensificarse cada evento para conservar su lectura?

Cuánto debe durar un `contact.press`. Cuánta presencia necesita un `signal.warn + risk`. Cuánta intensidad puede tener un `signal.alert + threat`. Cuánto tiempo debe permanecer un `undo`. Cuánto `motion` empieza a fatigar. Cuándo un `sustain.progress` deja de tranquilizar y empieza a desesperar. Cuándo un sonido breve ayuda y cuándo invade.

Esta pregunta es peligrosa. Si damos valores demasiado concretos, caemos en falsa precisión; si no damos ninguno, dejamos al lector igual que antes. Por eso este capítulo no propone leyes: propone rangos

psicoperceptivos orientativos. Un rango no dice «usa siempre 120 ms»; dice algo más útil. Esta clase de evento suele necesitar una temporalidad breve para preservar su función perceptiva. O esta señal debe persistir mientras el usuario pueda actuar sobre ella. O esta intensidad debe mantenerse baja porque el evento ocurre con mucha frecuencia. Los rangos son regiones de diseño, no números sagrados.

§ 1

Por qué hablar de rangos

La semántica perceptiva no solo depende de qué canal usamos; también depende de cuánto lo usamos. Un motion correcto puede fallar por exceso de duración; un color adecuado, por exceso de saturación; una señal necesaria, por desaparecer demasiado pronto. Un sonido útil, por repetirse demasiado; un loader, por durar sin explicar nada; una vibración, por ser demasiado intensa o demasiado frecuente. El canal no trabaja solo por presencia o ausencia: trabaja por intensidad, duración, ritmo, persistencia, frecuencia y contexto.

DISTINCIÓN

Un rango psicoperceptivo no es una ley. Es una zona en la que un evento conserva mejor su lectura.

§ 2

Fundamento: umbrales, atención y fatiga

Los rangos se apoyan en tres ideas ya desarrolladas en la Parte II.

Ventanas de percepción

El Capítulo 12 explicó que el tiempo modifica la lectura del evento: una respuesta muy próxima a la acción se percibe como contacto; una más tardía, como reacción; una espera prolongada exige señal explícita. Por eso contact necesita vivir cerca del gesto.

Activación atencional

Los intents ocupan posiciones distintas del espacio valencia/activación que el Capítulo 10 cartografió — de la baja activación de affirm a la alta de threat. Por eso no todos deben tener la misma intensidad, duración o persistencia.

Fatiga sensorial

Un canal puede funcionar bien una vez y volverse insoportable por repetición. Un sonido de confirmación puede ser agradable una vez y molesto cincuenta veces. Una microvibración puede reforzar el contacto en un gesto aislado y fatigar en un teclado; un motion de éxito puede resultar expresivo una vez y ridículo si ocurre cada minuto. La frecuencia modifica el rango aceptable.

FIGURA 32.1
De qué dependen los rangos



Los rangos no son valores universales: salen de cómo percibimos el tiempo, la activación y la repetición.

§ 3

Rangos temporales orientativos

No hay valores universales, pero sí regiones útiles. La tabla siguiente las resume por evento.

TABLA 32.1 Regiones temporales orientativas por evento

EVENTO	REGIÓN TEMPORAL ORIENTATIVA	FUNDAMENTO
contact.press	dentro de la ventana de atribución causal	agencia acción-respuesta
commit.save + affirm	breve, sin interrumpir flujo	confirmación suave
commit.complete + fulfill	breve-media, más resolutivo	cierre de tensión
emerge.tooltip	breve-media	aparición auxiliar
emerge.dropdown	breve	opciones locales
shift.enter-mode	media, con orientación	cambio de contexto
signal.warn + risk	persistente hasta corrección	permitir acción correctiva
signal.alert + threat	entrada rápida + persistencia hasta acción	urgencia
commit.delete + loss	breve + huella	consecuencia consumada
sustain.progress	mientras dure el proceso	continuidad
handle.carry	baja latencia continua	control directo

Lo importante no es memorizar la tabla, sino entender su lógica. Un `contact` demasiado largo deja de ser contacto; un `signal.warn + risk` demasiado breve no permite corregir. Un `sustain.progress` sin continuidad parece bloqueo; un `loss` sin huella puede parecer simple desaparición.

§ 4

contact: el rango de la agencia

`contact` es la familia más sensible al tiempo. Su pregunta —¿el sistema ha sentido mi acción?— solo puede responderse dentro de la ventana de atribución causal descrita en el Capítulo 12. No necesita durar mucho: necesita empezar pronto. Un buen `contact.press` puede ser casi instantáneo y muy breve —estado `pressed`, microcompresión, cambio de foco, pulso háptico leve, click opcional—; su función no es expresar belleza, sino preservar agencia.

Si el sistema no puede ejecutar la operación inmediatamente, debe al menos responder al contacto y después sostener el proceso: `contact.press`

→ sustain.progress → commit.save + affirm. El antipatrón correspondiente es el silencio seguido de resultado: el sistema puede haber funcionado, pero el usuario no percibe agencia.

§ 5

commit: el rango del cierre

commit necesita una señal de cierre: ni tan breve que pase desapercibida, ni tan larga que teatralice el resultado. La duración depende del intent. Un commit.save + affirm pide confirmación suave, breve y discreta —«guardado, puedes seguir»—. Un commit.complete + fulfill admite más duración, motion o presencia —«objetivo cumplido»—. Un commit.delete + loss pide cierre breve pero huella suficiente —«esto ya no está disponible»—. La clave es que commit no siempre necesita más intensidad: necesita claridad de consecuencia.

§ 6

signal: el rango de la atención

signal trabaja con atención, y la atención tiene coste. Una señal demasiado débil se pierde; demasiado fuerte, fatiga; demasiado breve, no se procesa; demasiado persistente, se convierte en ruido. Un risk necesita tiempo para que el usuario lo procese y corrija; un threat necesita entrar rápido y permanecer mientras la acción sea necesaria —pero una amenaza no debería gritar indefinidamente—.

TABLA 32.2 Persistencia e intensidad orientativas de signal

SIGNAL	PERSISTENCIA	INTENSIDAD
signal.announce + neutral	breve o contextual	baja
signal.notify + neutral	breve-media	baja-media
signal.warn + risk	hasta corrección/cierre	media
signal.alert + threat	hasta acción	alta, pero no infinita

§ 7

handle: el rango del control

handle tiene requisitos de rango muy específicos, porque es la única familia de control motor continuo. Durante `handle.carry`, la latencia debe ser mínima y constante: el usuario no interpreta un lag como «el sistema procesa», sino como «no estoy controlando esto». En `handle.pick` el rango es breve —un cambio de estado inmediato, elevación, sombra o háptica que diga «lo tienes»—. En `handle.drop` el rango depende del resultado: marca breve si es `commit.reorder + affirm`; huella y gravedad si es `commit.delete + loss`.

La regla de handle: el carry debe ser silencioso y fluido, y la evaluación se concentra en pick y drop. Sus antipatronos son el motion que compite con el control durante todo el carry y el lag variable que hace que el objeto «salte».

§ 8

sustain: el rango de la espera

sustain existe porque hay tiempo sin resultado, y su rango no se mide como un evento breve, sino como una continuidad. Un spinner breve puede funcionar; un spinner largo se vuelve pregunta; un spinner eterno se vuelve acusación. La regla: cuanto más dura una espera, más información debe ofrecer.

TABLA 32.3 Duración percibida de la espera y necesidad de información

DURACIÓN PERCIBIDA	NECESIDAD
muy breve	quizá no mostrar nada o una respuesta mínima
breve	indicador simple
media	texto de estado
larga	progreso, explicación o control

§ 9

emerge y shift: rangos de entrada y orientación

En emerge, una aparición local debe ser lo bastante clara para que el usuario entienda qué apareció, de dónde viene y a qué pertenece. Pero sin el peso de un cambio de contexto: un tooltip, un dropdown o un popover necesitan una entrada breve y ligera. Si la aparición dura demasiado o gana demasiada profundidad, empieza a parecer shift.

En shift, un cambio de contexto necesita más orientación: abrir un modal, entrar en edición, cambiar de vista o activar una command palette implica otra relación operativa. La transición debe ser suficiente para orientar, pero no tan larga que interrumpa.

§ 10

Sonido y háptica: rangos de intrusión

Sonido y háptica entran en el cuerpo o en el entorno del usuario, y por eso sus rangos deben ser más conservadores. En eventos frecuentes, mínimos o ausentes; en eventos raros y relevantes, posibles; en eventos críticos, opcionales pero nunca únicos. Un sustain no debería sonar ni vibrar de forma continua; un contact frecuente no debería llevar refuerzo fuerte; un threat puede tener entrada saliente, pero no un bucle agresivo.

FIGURA 32.2

Rangos de intrusión: sonido y háptica

entran en el cuerpo o el entorno del usuario; sus rangos deben ser más conservadores

AUSENTE / MÍNIMO	POSIBLE / SALIENTE
eventos frecuentes	mínimos o ausentes
contact frecuente	evitar refuerzo fuerte
sustain (proceso)	evitar sonido/háptica continuos
eventos raros y relevantes	posibles
eventos críticos · threat	opcionales, nunca únicos · entrada, no bucle

Lo frecuente debe callar; lo raro y grave puede sonar, pero nunca como único canal ni en bucle.

§ 11

Saturación y frecuencia

La intensidad aceptable depende de la frecuencia. Un evento que ocurre una vez al día puede permitirse más presencia; un evento que ocurre cien veces por hora debe ser casi invisible.

La frecuencia tiene además un caso límite: el evento continuo. Un arrastre emite muestras al ritmo del sensor —muy por encima de lo que la percepción resuelve—, y expresar cada muestra como señal completa satura cualquier canal. La regla es muestrear al ritmo de la percepción: agrupar las señales del movimiento al refresco perceptivo, con un suelo mínimo por debajo del cual repetir no añade nada. Y no muestrear las entradas discretas a ritmo humano, donde cada repetición es un valor real.

TABLA 32.4 Frecuencia probable e intensidad recomendable por evento

EVENTO	FRECUENCIA PROBABLE	INTENSIDAD
--------	---------------------	------------

		RECOMENDABLE
contact.press	muy alta	mínima
commit.save + affirm	alta	discreta
sustain.progress	media	baja saliencia
signal.notify + neutral	media	controlada
signal.warn + risk	media-baja	clara
commit.complete + fulfill	baja-media	resolutiva
signal.alert + threat	baja	alta, pero breve
commit.delete + loss	baja	grave, con huella

PRINCIPIO

La gramática debe tener memoria de frecuencia. No todo evento puede pedir atención como si fuera único.

§ 12

Rangos y accesibilidad

Los rangos también deben adaptarse a la accesibilidad. Un usuario puede necesitar más tiempo para leer una señal; otro, menos motion; otro puede tener el sonido desactivado; otro puede fatigarse con la vibración. Por eso los rangos no pueden ser fijos. Deben poder ajustarse mediante reduced motion, controles de sonido, duración suficiente de los mensajes, persistencia de los estados importantes, opción de cierre manual, ausencia de timeouts injustificados y alternativas a las señales efímeras.

Una señal importante no debería desaparecer antes de que el usuario haya tenido una oportunidad razonable de procesarla.

§ 13

Tabla síntesis de rangos

Los rangos del capítulo, condensados por familia e intent:

TABLA 32.5 Rangos orientativos por familia e intent

FAMILIA / INTENT	RANGO ORIENTATIVO	EVITAR
contact	inmediato, muy breve	retraso o teatralización
commit + affirm	breve y discreto	celebración
commit + fulfill	breve-media, resolutivo	trivialidad o espectáculo
signal + risk	persistente hasta corrección	desaparición rápida
signal + threat	entrada rápida + persistencia sobria	alarma sostenida
commit + loss	breve + huella	desaparición sin registro
handle	latencia mínima continua	lag o evaluación durante carry
emerge	entrada/salida ligera	peso de modal
shift	transición orientadora	cambio invisible
sustain	mientras dure proceso	spinner eterno sin información

§ 14

Antipatronos de rango

Microfeedback lento. Un contact que empieza tarde: se pierde agencia.

Confirmación excesiva. Un affirm que se celebra como fulfill: todo parece logro y nada lo parece.

Warning fugaz. Un risk que desaparece antes de poder corregir: el usuario no puede actuar sobre lo que ya no ve.

Threat infinito. Una alerta de sesión que suena en bucle, vibra el dispositivo y parpadea sin control. El usuario no puede silenciarla sin atender; si está en una reunión, en transporte público o con auriculares, la señal se convierte en agresión. El problema no es la amenaza —la sesión sí puede expirar—, sino que la intensidad no tiene techo ni control. La entrada puede ser saliente, pero la persistencia debe ser sobria: presencia visual firme, texto claro, sonido breve si procede, y después silencio mientras la alerta siga visible.

Loss sin huella. Algo se elimina y desaparece sin rastro: la pérdida se confunde con ocultación.

Spinner eterno. Proceso sin explicación ni salida: se rompe la confianza.

Handle con lag variable. Un objeto arrastrado que a veces sigue el gesto y a veces salta: el usuario pierde la sensación de control directo.

§ 15

Síntesis

Los rangos psicoperceptivos no son valores universales: son regiones de diseño. Sirven para mantener la función de cada evento: contact → agencia, commit → cierre, signal → atención, handle → control, emerge → presencia, shift → orientación, sustain → continuidad.

PRINCIPIO

Un evento no solo necesita el canal correcto. Necesita la cantidad correcta de canal.

PRINCIPIO

La intensidad debe corresponder a consecuencia, urgencia, frecuencia, reversibilidad y accesibilidad.

33

Accesibilidad aplicada

Migración semántica en flujos reales

La accesibilidad ya apareció varias veces en este libro: en la Parte I, como prueba de que una semántica no debe depender de un único canal; en la Parte II, como reducción de canales; en las familias, como exigencia concreta. Pero hasta ahora ha sido tratada sobre todo por partes. Este capítulo la aplica a flujos completos.

Un usuario con reduced motion no pierde «una animación»: puede perder causalidad, continuidad, cambio de contexto o cierre. Un usuario que no distingue color no pierde «rojo o verde»: puede perder riesgo, amenaza, confirmación o pérdida. Un usuario sin sonido no pierde «un aviso»: puede perder una señal fuera de foco. Un usuario que navega con teclado no pierde «drag»: puede perder la posibilidad de manipular. Un usuario con lector de pantalla no pierde «layout»: puede perder presencia, marco, orden y consecuencia.

PREGUNTA DEL CAPÍTULO

¿La semántica del evento sobrevive en condiciones perceptivas distintas?

FIGURA 33.1

Lo que se pierde no es el estímulo, sino el significado

CONDICIÓN	NO PIERDE EL ESTÍMULO...	...SINO EL SIGNIFICADO
reduced motion	«una animación» →	causalidad · continuidad cambio de contexto · cierre
sin color	«rojo o verde» →	riesgo · amenaza confirmación · pérdida
sin sonido	«un aviso» →	una señal fuera de foco
teclado	«drag» →	la posibilidad de manipular
lector de pantalla	«layout» →	presencia · marco · orden · consecuencia

La pregunta del capítulo: ¿la semántica del evento sobrevive en condiciones perceptivas distintas?

§ 1

Accesibilidad aplicada no es traducción literal

Cuando un canal se reduce, la solución no consiste en copiar el mismo estímulo por otro canal. La pregunta es anterior: ¿qué significado transportaba ese canal? Si el motion transportaba contacto, el reemplazo debe conservar contacto; si transportaba continuidad, debe conservar continuidad. Si el color transportaba riesgo, el reemplazo debe conservar riesgo corregible; si el sonido transportaba amenaza, debe conservar urgencia. A esto lo llamaremos migración semántica.

PRINCIPIO

La accesibilidad aplicada no conserva necesariamente los mismos estímulos. Conserva la misma lectura semántica.

§ 2

Caso 1: error de formulario

Evento base: signal.warn + risk. El usuario intenta enviar un formulario y un campo obligatorio está vacío. La lectura semántica esperada es «hay un problema corregible aquí» —no «todo ha fallado», no «hay una amenaza crítica»—.

Versión frágil

Un borde rojo en el input, y nada más. Los problemas se acumulan: depende solo del color, no explica qué ocurre, no comunica corrección, puede no ser detectado por usuarios con baja visión o daltonismo, puede no estar vinculado al campo y puede no anunciarse tras el envío.

Versión accesible semánticamente

La versión robusta reparte el significado entre canales, de modo que ninguno sea imprescindible:

TABLA 33.1 Reparto de funciones por canal en el error de formulario

CANAL	FUNCIÓN
Color	refuerzo de advertencia moderada
Forma	borde, callout o contorno local
Icono	reconocimiento rápido
Texto	explicación del problema
Presencia	persistencia hasta corrección
Foco	llevar al primer error si bloquea envío
Relación accesible	mensaje vinculado al campo
Tiempo	no desaparecer automáticamente

Migraciones

Sin color, el campo sigue comunicando error mediante icono, mensaje, borde, posición y relación con el campo. Sin color y con visión parcial, el mensaje debe estar vinculado al campo mediante aria-describedby: el usuario debe poder oír «el campo correo electrónico es obligatorio». Sin

motion, no pasa nada si el error no tiembla: el mensaje persistente sostiene la semántica. Y con varios errores conviene añadir un resumen —«hay 3 errores en el formulario»— y permitir saltar a cada campo.

§ 3

Caso 2: operación larga con reduced motion

Secuencia base: contact.press → sustain.progress → commit.complete + fulfill. El usuario sube un archivo grande.

Qué se perdería si solo quitamos motion

Si eliminamos todas las animaciones sin más, el contacto puede sentirse menos inmediato, el progreso puede parecer estático, el cierre puede perder resolución y el usuario puede no distinguir entre proceso activo y resultado final. Reducir motion no debe amputar la estructura temporal.

Versión accesible semánticamente

En contact: estado pressed claro, cambio inmediato de forma, foco visible, quizá sonido o háptica opcional. En sustain: barra de progreso sin motion excesivo, texto «Subiendo archivo...», porcentaje si existe, estado persistente, actualización discreta. En commit: mensaje visible, marca de completado, estado persistente, sonido opcional, color positivo claro pero no único. La lectura se conserva —«me ha sentido», «sigue trabajando», «objetivo cumplido»— aunque el movimiento se reduzca.

DISTINCIÓN

Reduced motion no significa interfaz sin tiempo. Significa tiempo expresado por canales menos problemáticos.

§ 4

Caso 3: eliminación con undo

Secuencia base: signal.warn + threat → commit.delete + loss → emerge.present undo.

Versión frágil

Botón rojo, la tarjeta desaparece, un toast breve dice «Eliminado», el undo dura dos segundos, no hay anuncio accesible y el foco queda en un lugar ambiguo. Los problemas: amenaza y pérdida se comunican con el mismo color, la desaparición puede parecer ocultación, el undo puede no percibirse, y los usuarios de teclado o lector pierden el contexto.

Versión accesible semánticamente

Cada momento del flujo tiene su propio riesgo y necesita su propio soporte:

TABLA 33.2 Migración semántica por momentos del flujo

MOMENTO	EVENTO	RIESGO	SOPORTE NECESARIO
Antes	signal.warn + threat	no percibir gravedad	texto, foco, forma, color
Después	commit.delete + loss	confundir con ocultación	mensaje, huella, actualización
Reparación	emerge.present undo	no llegar a deshacer	persistencia, botón claro, foco/anuncio

Antes, el signal.warn + threat pide texto claro, forma de advertencia, presencia suficiente, foco en la decisión y color solo como refuerzo: «puedes evitar una consecuencia». Después, el commit.delete + loss pide retirada de la tarea, mensaje de consecuencia, actualización de la lista y huella temporal, sin alarma sostenida: «esto ya se eliminó». Y la reparación —emerge.present undo— pide una barra o mensaje persistente, un botón de deshacer claro, tiempo suficiente, foco no perdido y anuncio si procede: «puedes reparar temporalmente».

§ 5

Caso 4: drag-and-drop accesible

Secuencia base: `handle.pick` → `handle.carry` → `handle.drop` → `commit.reorder` + `affirm`.

Versión frágil

Solo funciona con ratón o touch: no hay alternativa de teclado, no se anuncia la posición, no hay forma clara de cancelar. La semántica de manipulación depende por completo de un gesto físico y visual.

Qué debe conservarse

La accesibilidad no exige que todos hagan el mismo gesto; exige que todos puedan realizar y comprender el mismo evento operativo. El evento no es «arrastrar con ratón»: el evento es mover un objeto de una posición a otra.

Versión accesible semánticamente

Las opciones son conocidas: activar un modo «mover» con teclado, seleccionar el elemento, botones de subir y bajar, elegir destino, anunciar la posición actual y el resultado, permitir cancelar, marcar el destino válido y conservar el foco. Lo importante es que la semántica se conserva: «he elegido este objeto», «lo estoy moviendo», «quedó en otra posición».

PRINCIPIO

La accesibilidad no exige el mismo gesto. Exige el mismo evento operativo comprensible.

§ 6

Caso 5: señal crítica sin sonido ni color

Evento base: `signal.alert` + `threat`. La sesión expirará en treinta segundos si el usuario no actúa.

Sin sonido

La amenaza debe migrar a presencia dominante, texto claro, cuenta atrás si procede, color de alta saliencia, forma crítica, foco si bloquea la acción y persistencia hasta la respuesta.

Sin color

Debe sobrevivir mediante forma, texto, icono, posición, persistencia y, quizá, un motion contenido.

Sin sonido y sin color

Este es el caso extremo donde la migración se pone a prueba: la amenaza debe sobrevivir solo con forma, texto, posición, tamaño y persistencia. Un bloque de texto grande, bien posicionado, con icono reconocible, foco capturado y cuenta atrás textual puede comunicar urgencia sin depender de color ni de sonido. Si la semántica sobrevive aquí, la arquitectura multicanal funciona.

Con lector de pantalla

Debe anunciarse con la prioridad adecuada, pero sin crear ruido continuo. La semántica no es «hacer ruido»: es permitir la acción antes de la consecuencia.

§ 7

Principios de accesibilidad aplicada

Conservar significado, no estímulo. Si un canal se reduce, la primera pregunta es qué significado transportaba.

Diferenciar evento y gesto. El gesto puede cambiar; el evento debe mantenerse. Un drag físico puede convertirse en mover con teclado.

No depender del color para el intent. risk, threat, loss, affirm y fulfill deben sobrevivir sin color.

No depender del motion para la causalidad. Si el movimiento se reduce, la agencia migra a estado, foco, texto, sonido o háptica opcional.

No depender del sonido para la urgencia. Una amenaza debe tener presencia visual y textual suficiente por sí misma.

No hacer efímero lo importante. Las señales críticas y los resultados relevantes deben dejar huella.

Adaptar la intensidad. Accesibilidad también significa reducir fatiga sensorial.

§ 8

Ficha de migración semántica

Para revisar un flujo conviene rellenar una ficha. Sobre el ejemplo de eliminar una tarea con undo:

- **Flujo: eliminar tarea con undo. Evento principal: commit.delete + loss.**
- **Familias implicadas: signal, commit, emerge. Intents: threat, loss, neutral.**
- **Canales dominantes: presencia, forma, texto, color.**
- **Sin color: se mantiene con texto, forma e icono. Reduced motion: retirada menos animada, huella persistente. Sin sonido: no afecta.**
- **Teclado: foco en la confirmación y luego en el undo. Lector: anuncio de la eliminación y de la acción disponible.**
- **Significado en riesgo: pérdida + reparación temporal. Migración: mensaje persistente + undo + actualización de la lista.**

FIGURA 33.2

Ficha de migración semántica

Flujo: eliminar tarea con undo		commit.delete + loss
Familias	signal · commit · emerge	
Intents	threat · loss · neutral	
Canales dominantes	presencia · forma · texto · color	
Sin color	se mantiene con texto, forma, icono	
Reduced motion	retirada menos animada, huella persistente	
Sin sonido	no afecta	
Teclado	foco en confirmación y luego en undo	
Lector	anuncio de eliminación y acción disponible	
Significado en riesgo	pérdida + reparación temporal	
Migración	mensaje persistente + undo + actualización de lista	

§ 9

Antipatrones de accesibilidad aplicada

Adaptar el componente, no el evento. El componente cumple WCAG, pero el flujo completo no se entiende: el usuario puede activar cada control sin comprender la secuencia semántica.

Quitar motion sin sustituir significado. Se elimina el mareo, pero también la causalidad y la continuidad: el usuario deja de percibir qué causó qué.

Silenciar sin respaldo. Se apaga el sonido, pero la alerta fuera de foco se pierde: no queda otra forma de saber que algo reclama atención.

Alto contraste sin estructura. Se mejora la visibilidad, pero la jerarquía de eventos sigue confusa: se ve mejor, pero no se entiende mejor.

Teclado sin semántica. Se puede llegar al control, pero no entender el estado, el resultado ni la consecuencia del evento.

Lector de pantalla sin secuencia. Se anuncian piezas aisladas, pero no el flujo: el usuario oye elementos, no comprende la operación.

§ 10

Síntesis

La accesibilidad aplicada es donde la gramática multicanal demuestra si funciona. No basta con que un canal tenga alternativa, ni con que un componente cumpla una regla, ni con que haya etiquetas: lo que debe sobrevivir es la lectura.

PRINCIPIO

Una adaptación accesible no conserva necesariamente los mismos estímulos. Conserva la misma lectura semántica.

PRINCIPIO

Cuando un canal falla, migra el significado, no el adorno.

FIGURA 33.3

Siete principios de accesibilidad aplicada

- 1 Conservar significado, no estímulo**
si un canal se reduce, pregunta qué significado transportaba
- 2 Diferenciar evento y gesto**
el gesto puede cambiar; el evento debe mantenerse (drag → teclado)
- 3 No depender de color para intent**
risk, threat, loss, affirm y fulfill deben sobrevivir sin color
- 4 No depender de motion para causalidad**
la agencia debe migrar a estado, foco, texto, sonido o háptica
- 5 No depender de sonido para urgencia**
una amenaza debe tener presencia visual o textual suficiente
- 6 No hacer efímero lo importante**
señales críticas y resultados relevantes deben dejar huella
- 7 Adaptar intensidad**
accesibilidad también significa reducir fatiga sensorial

34

Antipatrones globales

Cuando la gramática se rompe

Una gramática no se entiende del todo hasta que muestra sus errores.

Hasta ahora hemos definido eventos, familias, intents, canales, composición, matrices, rangos y accesibilidad aplicada. Pero en una interfaz real, los problemas no siempre aparecen como fallos aislados. Muchas veces aparecen como rupturas de gramática.

No es solo que una animación sea demasiado larga. No es solo que un color tenga poco contraste. No es solo que un sonido moleste. El problema más profundo suele ser otro: el sistema comunica una clase de evento distinta de la que realmente está ocurriendo.

PREGUNTA DEL CAPÍTULO

¿Qué ocurre cuando la interfaz dice una cosa y el evento significa otra?

§ 1

Todo es success

Qué se hace mal: toda operación positiva se comunica igual. Un autoguardado, un toggle, una preferencia cambiada, un formulario enviado, una subida terminada — todo recibe el mismo verde, el mismo check, la misma animación. `commit.save` + `affirm` y `commit.complete` + `fulfill` quedan aplastados bajo una sola categoría: `success`.

Por qué parece correcto: porque ambos casos son positivos. El sistema hizo algo bien. La convención heredada del semáforo refuerza esta simplificación.

Qué fundamento viola: la distinción entre valencia y activación. `Affirm` confirma normalidad. `Fulfill` resuelve una tensión. El autoguardado no debería sentirse como culminación.

Corrección: separar confirmación suave (marca discreta, baja saliencia, sin sonido) de resolución positiva (motion resolutivo, cierre visible, color positivo más claro).

PRINCIPIO

Si todo es success, nada se siente realmente logrado.

§ 2

Todo es danger

Qué se hace mal: todo evento negativo fuerte se comunica igual. Una advertencia, un recurso eliminado, un fallo de validación, una conexión perdida, una acción irreversible — todo se comunica con rojo, alerta y tono grave.

Por qué parece correcto: porque todos tienen valencia negativa. La convención permite agruparlo todo bajo `danger`.

Qué fundamento viola: la diferencia temporal y afectiva entre `risk`, `threat` y `loss` que argumenta el Capítulo 10. Si una pérdida se comunica como amenaza, el sistema genera urgencia donde ya no hay prevención posible.

Corrección: separar los tres casos. `signal.warn` + `risk` para «revisa esto.» `signal.alert` + `threat` para «actúa ahora.» `commit.delete` + `loss` para «esto ya se perdió.»

PRINCIPIO

No todo lo negativo debe gritar. A veces debe corregirse. A veces debe evitarse. A veces solo debe registrarse.

§ 3

Info como cajón de sastre

Qué se hace mal: todo lo que no es `success`, `warning` o `danger` se llama `info`. Un dato contextual, un mensaje neutro, una notificación, un consejo, una ayuda, una confirmación leve — todo cae en `info`.

Por qué parece correcto: porque `info` parece una categoría segura. No es positivo ni negativo.

Qué fundamento viola: la separación entre saliencia atencional y evaluación afectiva. `Info` no responde a «¿cómo se evalúa esto?» sino a «¿debo notarlo?» Eso pertenece mejor a `signal`.

Corrección: sustituir `info` como `intent` por composiciones de señal neutra. `signal.announce` + `neutral` para información contextual. `signal.notify` + `neutral` para notificación sin carga evaluativa.

DISTINCIÓN

Info no es una emoción del evento. Es una función de atención.

§ 4

Todo lo que aparece es alerta

Qué se hace mal: cualquier cosa que aparece se diseña como si reclamara atención fuerte. Un tooltip entra con demasiada saliencia. Un dropdown parece una alerta. Una ayuda contextual usa color de `warning`.

Por qué parece correcto: porque si algo aparece, el equipo quiere que el usuario lo vea. Entonces se aumenta contraste, motion, sombra o color.

Qué fundamento viola: la diferencia entre emerge y signal. emerge introduce presencia. signal reclama atención. No toda aparición es una señal.

Corrección: emerge.open para dropdown. emerge.present para tooltip. signal.warn + risk solo si lo que aparece comunica problema corregible. signal.alert + threat solo si exige atención inmediata.

DISTINCIÓN

Aparecer no es advertir. Entrar en el campo no es reclamarlo entero.

§ 5

Modal rojo

Qué se hace mal: un modal contiene una advertencia y todo el modal se vuelve peligroso. Fondo rojo, borde rojo, botones rojos, toda la superficie absorbe el intent. Esto no es la excepción aparición-que-advierte del Capítulo 9: aquella concierne al tono del evento de aparición; este antipatrón, a una superficie que absorbe la expresión del mensaje que contiene.

Por qué parece correcto: porque el contenido del modal es grave. Parece razonable que el modal completo se diseñe como danger.

Qué fundamento viola: la distinción entre marco y mensaje. El modal es shift.enter-mode — cambia el contexto. La advertencia dentro del modal es signal.warn + threat. El modal no es la amenaza. El mensaje porta la amenaza.

Corrección: componer shift.enter-mode + signal.warn + threat. El modal comunica cambio de contexto mediante profundidad y foco. La advertencia comunica amenaza mediante forma, texto, color e icono.

PRINCIPIO

El intent debe vivir en la figura evaluable, no en el contenedor estructural.

§ 6

Spinner de error

Qué se hace mal: un proceso en curso se comunica con color, motion o tono de error. Un spinner rojo. Un loader tenso. Una espera con alarma.

Por qué parece correcto: porque esperar genera ansiedad. El equipo quiere avisar de que algo importante está ocurriendo.

Qué fundamento viola: la diferencia entre sustain y signal/commit. sustain.progress comunica «esto sigue ocurriendo,» no «esto falló» ni «esto es peligroso.»

Corrección: mantener sustain neutro. Si aparece un problema durante el proceso: sustain.progress + signal.warn + risk. Si falla: commit.fail + risk.

DISTINCIÓN

Una espera no es un error por durar. Es un proceso que necesita continuidad.

§ 7

commit anticipado

Qué se hace mal: la interfaz comunica resultado antes de que exista. Un botón muestra success al pulsar. Un formulario dice «enviado» antes de recibir respuesta.

Por qué parece correcto: porque el sistema quiere dar una respuesta rápida y sentirse optimista.

Qué fundamento viola: la diferencia entre contact, sustain y commit. contact dice «he recibido tu acción.» Sustain dice «sigo trabajando.»

Commit dice «ya terminó.» Si se usa commit en el momento de contact, la interfaz promete una consecuencia que aún no existe.

Corrección: `contact.press` → `sustain.progress` → `commit.complete` + `fulfill`.

PRINCIPIO

No celebres lo que todavía no ha ocurrido.

§ 8

Pérdida sin huella

Qué se hace mal: algo desaparece y no queda rastro. Una tarjeta se elimina con fade out. Un archivo desaparece de una lista. No hay mensaje, undo, registro ni estado.

Por qué parece correcto: porque visualmente la desaparición parece suficiente. El objeto ya no está. La interfaz se limpió.

Qué fundamento viola: la diferencia entre desaparición y pérdida. `emerge.hide` no es `commit.delete` + `loss`. Una pérdida no solo retira presencia — cambia disponibilidad operativa.

Corrección: `commit.delete` + `loss` con huella: retirada, mensaje, undo si existe, actualización de estado.

DISTINCIÓN

Una pérdida importante no debe desaparecer como si solo se hubiera ocultado.

§ 9

Drag sin alternativa

Qué se hace mal: la única forma de manipular un objeto es arrastrar con ratón o touch.

Por qué parece correcto: porque visualmente el drag-and-drop es intuitivo para muchos usuarios.

Qué fundamento viola: la accesibilidad operativa de handle. handle no significa «arrastrar con ratón.» Significa «controlar directamente un objeto.»

Corrección: ofrecer alternativas: botones subir/bajar, seleccionar y elegir destino, mover con teclado, anuncio de posición, cancelación clara.

PRINCIPIO

La interfaz puede cambiar el gesto. No debe perder el evento.

§ 10

Sonido decorativo

Qué se hace mal: la interfaz suena sin necesidad semántica. Cada click suena. Cada notificación suena igual. Cada éxito tiene chime. Los procesos tienen loop sonoro.

Por qué parece correcto: porque el sonido parece añadir riqueza o personalidad.

Qué fundamento viola: la relación entre canal y evento. El sonido no debe existir porque «queda bien.» Debe reforzar una dimensión que el evento necesita.

Corrección: sonido solo si responde a función clara: contacto, cierre, urgencia, pérdida, fuera de foco.

PRINCIPIO

Si un sonido no ayuda a reconocer qué ocurrió, cuándo ocurrió o cómo debe evaluarse, probablemente sobra.

§ 11

Todo vibra

Qué se hace mal: la interfaz usa háptica para todo. Cada toque, selección, error, scroll y notificación vibra.

Por qué parece correcto: porque la háptica parece reforzar contacto.

Qué fundamento viola: la proporcionalidad corporal. La háptica ocurre en el cuerpo del usuario. Puede ayudar pero también invadir y fatigar.

Corrección: reservar háptica para contacto relevante, pick/drop, encaje, alerta importante, pérdida significativa. Debe ser opcional, proporcional y nunca única.

DISTINCIÓN

La háptica no es más feedback. Es contacto corporal con el evento.

§ 12

Color como semántica

Qué se hace mal: el color define el significado completo del evento. Rojo = error. Verde = success. Amarillo = warning. Azul = info.

Por qué parece correcto: porque es la convención heredada. Fácil, familiar, rápida.

Qué fundamento viola: el principio central del sistema de intents. El intent no nace del color. Nace de la evaluación del evento en el espacio valencia/activación.

Corrección: partir de evento → evaluación → intent → canales. No de color → significado.

PRINCIPIO

El color expresa la semántica. No la funda.

§ 13

Señales efímeras

Qué se hace mal: mensajes importantes aparecen y desaparecen demasiado rápido. Un error, una pérdida, una advertencia, un undo — el usuario no alcanza a leer, actuar o recuperar.

Por qué parece correcto: porque el diseño busca no interrumpir. Los mensajes efímeros parecen ligeros y poco invasivos.

Qué fundamento viola: accesibilidad temporal y persistencia semántica. Si un evento exige acción, corrección o reconocimiento, debe persistir lo suficiente.

Corrección: affirm breve. Risk hasta corrección. Threat hasta acción. Loss con huella. Undo con duración suficiente y accesible.

DISTINCIÓN

Lo importante no debe existir solo durante un instante.

§ 14

shift invisible

Qué se hace mal: el contexto cambia pero el usuario no recibe orientación. Una nueva vista aparece sin foco. Un modal abre pero el lector sigue en el fondo. Un modo de edición se activa sin señal.

Por qué parece correcto: porque visualmente el cambio parece claro para quien lo diseña.

Qué fundamento viola: la función de shift. El usuario necesita saber dónde está, qué puede hacer, cómo vuelve, qué cambió.

Corrección: todo shift debe cuidar foco, título, profundidad, landmarks, navegación de retorno, estado de modo.

PRINCIPIO

Un cambio de contexto que el usuario no puede ubicar es un shift fallido, por elegante que sea la transición.

15. Tabla resumen

§ 15

Síntesis

Los antipatrones globales muestran que la gramática puede romperse de muchas formas. No basta con que un componente funcione. No basta con que una animación sea fluida. No basta con que un color sea reconocible. La pregunta es: ¿el sistema comunica el evento correcto, con la intensidad correcta, por los canales correctos, en el momento correcto?

DISTINCIÓN

Un antipatrón semántico no es solo una mala práctica visual. Es una contradicción entre el evento que ocurre y el evento que la interfaz hace percibir.

PRINCIPIO

Cuando una interfaz se siente rara, muchas veces no falla el efecto. Falla la gramática.

FIGURA 34.1

Antipatrones del tono: el intent equivocado

Todo es success	todo se celebra; ninguna confirmación se distingue de otra
Todo es danger	todo alarma; la alarma deja de significar amenaza
Info como cajón de sastre	saliencia usada como si fuera valencia
Modal rojo	un cambio de marco (shift) vestido de amenaza (threat)
Color como semántica	el intent depende solo del color y no sobrevive sin él

FIGURA 34.2

Antipatrones del evento: la familia equivocada

la interfaz hace percibir un evento distinto del que ocurre

Todo lo que aparece es alerta	un emerge tratado como signal
Spinner de error	un sustain (proceso) con color de error
Commit anticipado	se da por hecho antes de que el evento ocurra
Pérdida sin huella	un loss que no deja registro ni undo
Shift invisible	un cambio de marco que no se percibe
Drag sin alternativa	un handle sin otra vía para manipular

FIGURA 34.3

Antipatrones del canal y el momento

Sonido decorativo	sonido sin significado: ruido que se aprende a ignorar
Todo vibra	háptica que pierde su valor por exceso
Señales efímeras	un signal que desaparece antes de poder leerse

**Cuando una interfaz se siente rara, muchas veces no falla el efecto.
Falla la gramática.**

Un antipatrón semántico es una contradicción entre el evento que ocurre y el que la interfaz hace percibir.

TABLA 34.1 Antipatrones globales: qué confunden y cómo se corrigen

ANTIPATRÓN	QUÉ CONFUNDE	CORRECCIÓN
Todo es success	affirm con fulfill	separar confirmación y culminación

Todo es danger	risk, threat, loss	separar riesgo, amenaza y pérdida
Info como cajón	saliencia con intent	signal + neutral
Todo aparece como alert	emerge con signal	separar presencia y atención
Modal rojo	marco con mensaje	shift + signal.threat
Spinner de error	proceso con fallo	sustain + señal si procede
Commit anticipado	contacto con resultado	contact → sustain → commit
Pérdida sin huella	ocultación con pérdida	commit.loss + huella
Drag sin alternativa	gesto con evento	alternativa operativa
Sonido decorativo	canal con adorno	sonido solo si comunica evento
Todo vibra	respuesta invasiva	háptica proporcional
Color como semántica	canal con intent	intent antes que color
Señales efímeras	ligereza con pérdida	persistencia proporcional
Shift invisible	cambio sin orientación	foco y estructura

35

Validación de una semántica perceptiva

Cómo saber si el sistema comunica lo que cree comunicar

Una gramática de la interfaz no debería sostenerse solo porque suene coherente. Si este libro propone una gramática perceptiva del evento, debe poder hacerse una pregunta incómoda:

PREGUNTA DEL CAPÍTULO

¿Cómo sabemos si el usuario percibe el evento que creemos estar comunicando?

Porque una interfaz puede estar muy bien construida desde el punto de vista técnico y aun así fracasar semánticamente. El botón funciona, la animación corre a sesenta cuadros por segundo, el color cumple el contraste exigido. Y sin embargo el usuario puede no entender qué ocurrió: si el sistema recibió su acción, si algo sigue procesando, si algo ya terminó, si debe corregir, si algo se perdió.

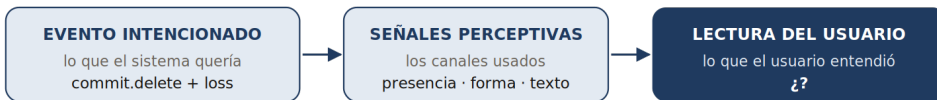
Conviene dejar claro desde el principio qué es este capítulo: una guía de uso, no un informe. El libro no aporta aquí datos ni resultados —los

resultados los producirá cada equipo sobre su propio producto—. Lo que aporta son seis protocolos para obtenerlos, una ficha para registrarlos y una tabla para interpretarlos. Las respuestas de usuario que aparecen como ejemplo no son hallazgos del libro: son ilustraciones de qué conviene escuchar cuando el lector ejecute cada prueba.

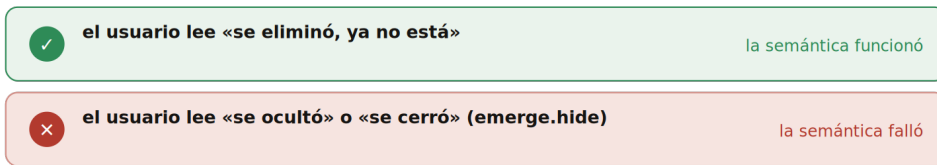
FIGURA 35.1

La triangulación de la validación

validar es comprobar si estas tres cosas coinciden



EL RESULTADO DEPENDE DE LA LECTURA



Una semántica no se valida preguntando si el efecto gusta. Se valida preguntando qué evento ha entendido el usuario.

§ 2

Qué no es validación semántica

Conviene despejar cuatro confusiones. La validación semántica no es usabilidad general: un usuario puede completar una tarea sin entender los eventos intermedios. No es un test de preferencia: preguntar si gusta una animación no valida si comunica contact o commit. No es solo accesibilidad técnica: los roles correctos no garantizan una lectura correcta. Y no es QA funcional: que el estado cambie no significa que el usuario lo perciba.

§ 3

Qué se valida

La validación recorre los niveles de la propia gramática: qué familia percibió el usuario, qué verbo o fase, qué evaluación y qué composición. Antes de ejecutar ningún protocolo conviene decidir qué nivel se pone a prueba, porque la pregunta que se hará al usuario cambia con él.

TABLA 35.1 Niveles de validación y su pregunta

NIVEL	PREGUNTA
Familia	¿qué tipo de evento percibió?
Verbo	¿qué cree que ocurrió concretamente?
Intent	¿cómo evalúa el evento?
Canal	¿qué señales usó para entenderlo?
Secuencia	¿entendió el orden de los eventos?
Accesibilidad	¿se conserva la lectura si se reduce un canal?
Fatiga	¿la señal sigue siendo proporcional en repetición?

Protocolo 1 — Test de lectura inmediata

Cómo hacerlo. Se muestra al usuario una interacción —en producto o en prototipo— y, nada más terminar, se le pregunta con lenguaje natural: «¿qué acaba de ocurrir?». Sin explicar nada antes, sin sugerir opciones, sin dar tiempo a racionalizar: interesa la primera lectura, no la reconstrucción.

Qué escuchar. La respuesta se compara con el evento diseñado. Si el evento era un `contact.press`, respuestas como «he pulsado» o «el botón respondió» indican que la semántica llegó; «se ha guardado» o «no sé si ha pasado algo» indican que no. Si era un `sustain.progress`, «está cargando» confirma la lectura; «ha fallado» o «se bloqueó» delatan una espera que parece avería. Si era un `commit.delete + loss`, «se eliminó» confirma; «se ocultó», «se cerró» o «no sé dónde fue» delatan una pérdida que parece ocultación.

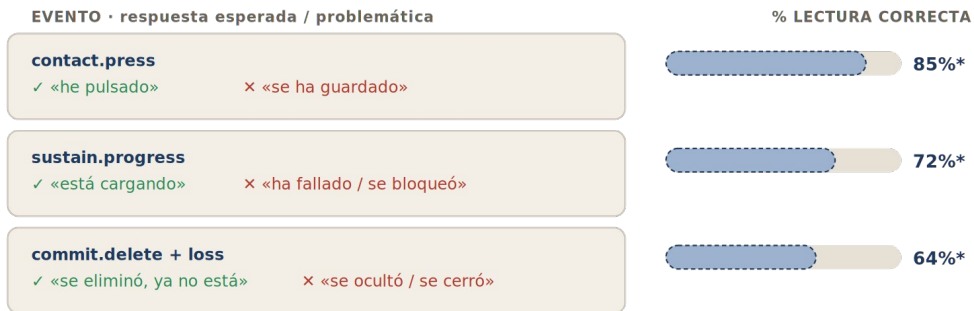
PRINCIPIO

Si el usuario no puede decir qué ocurrió, la interfaz no comunicó un evento. Comunicó un cambio.

FIGURA 35.2

Protocolo 1: test de lectura inmediata

mostrar la interacción y preguntar enseguida: «¿qué acaba de ocurrir?»



* cifras de ejemplo, pendientes de sustituir por los resultados reales de validación.

Si el usuario no puede decir qué ocurrió, la interfaz no comunicó un evento: comunicó un cambio.

§ 5

Protocolo 2 — Test de diferenciabilidad

Cómo hacerlo. Se muestran dos eventos cercanos, uno después de otro, y se pregunta qué diferencia percibe el usuario entre ambos. Es la prueba de que el sistema no solo comunica, sino que distingue.

Qué escuchar. Entre un contact.press y un commit.save + affirm, la lectura sana es «en el primero solo responde; en el segundo se guardó»; «parecen lo mismo» revela que contacto y resultado comparten expresión. Entre un signal.warn + risk y un signal.alert + threat conviene preguntar cuál requiere actuar antes: si ambas «parecen igual de graves», el espacio evaluativo no está llegando.

FIGURA 35.3

Protocolo 2: test de diferenciabilidad

comparar dos eventos y preguntar: «¿qué diferencia percibes?»

DATOS PROVISIONALES



* cifras de ejemplo, pendientes de sustituir por los resultados reales de validación.

Si dos eventos «parecen lo mismo», la gramática no está diferenciando lo que debería.

§ 6

Protocolo 3 — Test de intent

Cómo hacerlo. Se muestra un evento evaluable y se pregunta por su tono, siempre con lenguaje natural, nunca con los términos del sistema: no «¿esto es risk?», sino «¿esto parece corregible o urgente?», «¿dirías que algo se perdió o que algo terminó bien?». Cada intent tiene su pregunta característica, y la tabla siguiente las reúne para que el lector componga su guion.

TABLA 35.2 Preguntas de validación por intent

INTENT	PREGUNTA DE VALIDACIÓN
neutral	¿esto parece informativo sin carga fuerte?
affirm	¿esto confirma que todo va bien sin interrumpir?
fulfill	¿esto parece objetivo cumplido?
risk	¿esto parece corregible?
threat	¿esto exige actuar ahora?
loss	¿esto ya ocurrió y dejó consecuencia?

§ 7

Protocolo 4 — Test de reducción de canal

Cómo hacerlo. Se prueba el mismo evento retirando un canal cada vez: sin color, con reduced motion activado, sin sonido, solo con teclado, con lector de pantalla. En cada variante se repite el protocolo 1.

Qué escuchar. La pregunta correcta no es «¿se sigue viendo parecido?», sino «¿se sigue entendiendo el mismo evento?». Un evento robusto conserva su familia y su intent en todas las variantes; uno frágil cambia de significado cuando pierde su canal favorito.

§ 8

Protocolo 5 — Test de secuencia

Cómo hacerlo. Se muestra una operación completa —no un evento aislado— y se pide al usuario que cuente qué pasó primero, qué después y qué al final.

Qué escuchar. Si el relato conserva los momentos —«pulsé, estuvo trabajando, avisó de algo, al final quedó guardado»—, la composición funciona. Si el usuario solo percibe «hubo un error y luego un success», la composición perdió matiz: el contacto, la espera y la advertencia intermedia no llegaron a existir para él.

§ 9

Protocolo 6 — Test de fatiga

Cómo hacerlo. Se somete el evento a repetición —veinte veces seguidas, o su frecuencia real de uso durante una sesión— y se pregunta al final si sigue siendo tolerable, si se ha vuelto molesto, si ha perdido significado.

Qué escuchar. Un microclick puede ser útil una vez y fatigar en cada tecla; un guardado con animación puede ser agradable una vez y absurdo

en cada autoguardado. La repetición es la variable que los demás protocolos ignoran, y la que más antipatrones de intensidad destapa.

§ 10

La ficha de validación semántica

Los seis protocolos se registran en una ficha. Es una plantilla para rellenar —una por flujo evaluado—, y su valor está en obligar a explicitar la lectura esperada antes de escuchar la real:

- **Nombre del flujo y evento o secuencia evaluada.**
- **Familia esperada, intent esperado, canales usados y canal dominante.**
- **Lectura esperada, lectura real y diferencia detectada.**
- **¿Se distingue de los eventos cercanos?**
- **¿Sobrevive sin color, con reduced motion, sin sonido, solo con teclado?**
- **¿Produce fatiga por repetición?**
- **Corrección propuesta.**

§ 11

Cómo interpretar los resultados

La validación busca patrones, no anécdotas. Si un usuario se confunde, puede ser ruido; si varios confunden lo mismo, hay un problema semántico. La tabla siguiente ayuda a traducir las respuestas típicas en diagnóstico y corrección: es la guía de lectura de la ficha, no un catálogo de resultados.

TABLA 35.3 Respuestas típicas, diagnóstico y corrección

RESPUESTA DEL USUARIO	DIAGNÓSTICO PROBABLE	CORRECCIÓN
-----------------------	----------------------	------------

«No sé qué pasó»	evento poco legible	reforzar presencia / texto / huella
«Se guardó» ante contact	commit anticipado	separar contact de commit
«Parece peligroso» ante risk	exceso de activación	reducir saliencia
«Parece oculto» ante loss	falta huella	mensaje / undo / registro
«Parece modal» ante dropdown	profundidad excesiva	reducir shift, reforzar emerge
«Se bloqueó» ante sustain	continuidad insuficiente	texto / progreso
«Molesta tras repetir»	fatiga sensorial	reducir intensidad / frecuencia

§ 12

Validación sin laboratorio

Nada de esto exige un laboratorio formal. Un equipo puede ejecutar los seis protocolos con un prototipo interactivo, entre cinco y ocho usuarios, grabación de pantalla y preguntas inmediatas; comparando variantes, probando con reduced motion, revisando con teclado y sin color. Lo que distingue la validación semántica no es el aparato, sino la pregunta.

PRINCIPIO

La validación semántica no pregunta si el usuario puede usar la interfaz. Pregunta si puede leer lo que la interfaz está diciendo.

§ 13

Antipatrones de validación

Preguntar si gusta. El gusto no equivale a lectura: una animación puede encantar y no comunicar nada.

Validar solo el resultado final. El usuario puede completar la tarea sin haber entendido los eventos intermedios; la tarea sale bien y la semántica está rota.

Probar solo la versión completa. Sin probar la reducción de canales no se sabe si la semántica es robusta o si dependía de un solo canal.

Confundir rapidez con claridad. Una interacción rápida puede ser ambigua; el tiempo de tarea no mide la lectura.

Preguntar con términos del sistema. No se pregunta «¿percibiste risk?», sino «¿esto parecía corregible o urgente?».

Ignorar la fatiga. Una microinteracción agradable puede volverse insoportable en repetición, y el test de una sola exposición no lo verá.

§ 14

Síntesis

Este capítulo deja en manos del lector un instrumento, no un resultado: seis protocolos, una ficha y una tabla de interpretación. La validación semántica cierra el circuito entre lo que el sistema quiso decir, lo que mostró y lo que el usuario entendió — y ese circuito solo puede cerrarlo quien tiene delante a sus usuarios.

DISTINCIÓN

Una semántica perceptiva no debe defenderse solo porque suene coherente. Debe poder ponerse a prueba.

PRINCIPIO

Si el usuario interpreta otro evento distinto del que el sistema quería comunicar, la semántica falló aunque la interfaz funcione.

36

De la gramática al sistema

Nombres, documentación, tokens y diseño operativo

Una gramática de interfaz no se vuelve útil solo porque esté bien formulada. Se vuelve útil cuando puede entrar en el trabajo diario de un equipo. Cuando ayuda a nombrar eventos. Cuando permite documentar decisiones. Cuando ordena tokens. Cuando mejora revisiones de diseño. Cuando evita discusiones vagas.

PREGUNTA DEL CAPÍTULO

¿Cómo puede una gramática así entrar en un sistema de diseño sin convertirse en un framework rígido?

§ 1

La gramática no es un framework

Este libro no propone que todos los equipos adopten los mismos nombres internos, los mismos tokens ni la misma arquitectura. La gramática cumple otra función: obliga a distinguir funciones que la implementación no debería mezclar.

Un equipo puede llamar a algo danger. Otro puede llamarlo destructive. Otro puede usar critical. Lo importante no es el nombre exacto. Lo importante es que el sistema distinga: risk $\neq$ threat $\neq$ loss, affirm $\neq$ fulfill, emerge $\neq$ shift, contact $\neq$ commit.

PRINCIPIO

La gramática no obliga a una implementación concreta. Obliga a no confundir funciones perceptivas distintas.

§ 2

Por qué llevar la gramática al sistema

Si la gramática se queda solo en el libro, sirve para pensar. Pero si entra en el sistema, también sirve para coordinar. En muchos equipos, las decisiones sobre interfaz se reparten: diseño visual define color, frontend implementa estado, UX define copy, accesibilidad revisa foco, producto define prioridad, motion se añade al final, sonido ni se considera.

La gramática permite alinear esas decisiones alrededor de un evento. Por ejemplo, signal.warn + risk implica: copy que explica un problema corregible, color que refuerza advertencia sin amenaza, forma que localiza el problema, motion contenido, accesibilidad que persiste y vincula al campo.

FIGURA 36.1

Por qué llevar la gramática al sistema



Si la gramática se queda en el libro, sirve para pensar; si entra en el sistema, también para coordinar.

§ 3

Tres niveles de adopción

3.1. NIVEL CONCEPTUAL

El equipo usa la gramática como lenguaje de revisión. Preguntas típicas: ¿esto es contact o commit? ¿esto es risk o threat? ¿esta desaparición es hide o loss? No requiere cambiar código ni tokens. Solo requiere cambiar cómo se conversa.

3.2. NIVEL DOCUMENTAL

El sistema de diseño documenta eventos semánticos. Cada evento tiene pregunta perceptiva, canales recomendados, errores comunes, accesibilidad y ejemplos. Aquí la gramática se vuelve documentación.

3.3. NIVEL OPERATIVO

La gramática entra en tokens, componentes, eventos declarativos o herramientas. Este nivel es más potente pero también más peligroso. Si se implementa sin criterio, puede convertirse en rigidez.

§ 4

Naming: nombrar eventos, no efectos

Muchos sistemas nombran por apariencia (red-alert, green-success, slide-in, shake-error) o por componente (ButtonDanger, SuccessToast, ErrorModal). Ese naming mezcla planos.

Una gramática semántica intenta nombrar por evento: contact.press, commit.save + affirm, signal.warn + risk, commit.delete + loss.

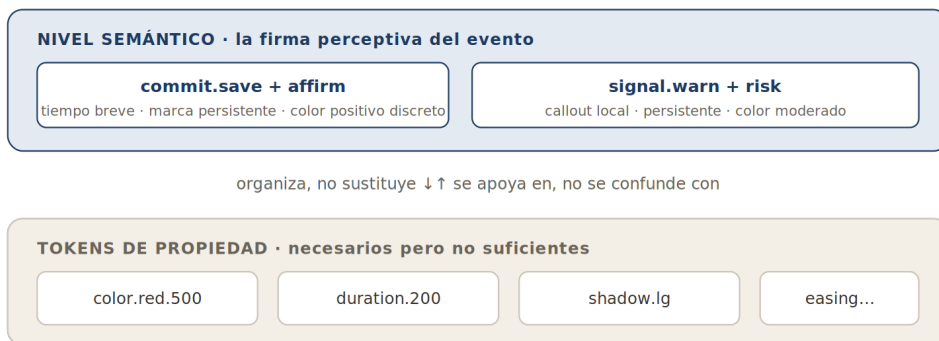
§ 5

Tokens semánticos

Los tokens suelen organizarse por propiedades: color.red.500, duration.200, shadow.lg. Eso es necesario pero no suficiente. Una gramática semántica no sustituye esos tokens. Los organiza en un nivel superior. Pero hay que pensar en firmas perceptivas, no en tokens de intent que parezcan tokens de color.

FIGURA 36.2

Tokens semánticos: un nivel superior, no un disfraz



Hay que pensar en firmas perceptivas, no en tokens de intent que parezcan tokens de color.

§ 6

Firma perceptiva

Una firma perceptiva es el conjunto de decisiones de canal asociadas a un evento. Por ejemplo, `commit.save` + `affirm`: tiempo breve, motion asentamiento mínimo, forma marca persistente, color positivo discreto, sonido no por defecto. La firma perceptiva evita que cada canal se decida por separado.

§ 7

Fichas de evento semántico

Una documentación de sistema podría incluir fichas como esta:

Ficha: `commit.delete` + `loss`. Pregunta: ¿qué dejó de estar disponible? Canales dominantes: presencia, forma, texto. Canales de apoyo: color, motion, sonido opcional. Firma perceptiva: retirada del objeto, mensaje de consecuencia, huella o registro, undo si existe. Accesibilidad: no depender solo de desaparición visual, anunciar pérdida si procede, mantener undo el tiempo suficiente. No usar: alarma sostenida de threat, solo color rojo, fade out sin huella. Antipatrones cercanos: todo es danger, pérdida sin huella, threat después de loss.

Ficha: `signal.warn` + `risk`. Pregunta: ¿qué requiere revisión? Canales dominantes: forma, texto, presencia. Firma: callout local, mensaje claro, persistencia hasta corrección, color moderado. Accesibilidad: mensaje vinculado al campo, no depender solo de color, no desaparecer automáticamente. No usar: alarma de threat, modal si no cambia el marco, shake agresivo.

Ficha: `commit.complete` + `fulfill`. Pregunta: ¿qué objetivo quedó cumplido? Canales dominantes: motion, presencia, forma. Firma: cierre resolutivo, marca de completado, color positivo visible, sonido breve opcional. No usar: success genérico para todo, celebración excesiva, desaparición sin estado final.

FIGURA 36.3

Ficha de evento semántico

commit.delete + loss		¿qué dejó de estar disponible?
Canales dominantes	presencia · forma · texto	
Canales de apoyo	color · motion · sonido opcional	
Firma perceptiva	retirada del objeto · mensaje de consecuencia · huella · undo	
Accesibilidad	no depender solo de la desaparición visual anunciar la pérdida · mantener undo el tiempo suficiente	
No usar	alarma sostenida de threat · solo color rojo · fade sin huella	
Antipatronos cercanos	todo es danger · pérdida sin huella · threat después de loss	

La ficha documenta la decisión, no el componente: qué evento es, cómo suena y qué evitar.

§ 8

Documentar decisiones, no solo componentes

Un design system suele documentar componentes: Button, Modal, Toast, Tooltip, Spinner. Pero un botón puede participar en contact.press, commit.delete + loss, signal.warn + threat, shift.navigate. Un modal puede participar en shift.enter-mode, signal.alert + threat, commit.cancel + neutral. La documentación debería incluir una capa de eventos, no solo de componentes.

§ 9

Tokens de canal y tokens de evento

Tokens de canal definen propiedades perceptivas básicas: color.red.600, motion.duration.short. Tokens de evento agrupan decisiones por semántica: event.signal.warn.risk podría mapear a color intent.risk.color,

shape callout.warning, duration persistent.untilResolved. El token de evento no reemplaza los tokens de canal. Los coordina.

§ 10

Cuidado: tokens no son verdad

Si el sistema convierte la gramática en tokens demasiado rígidos, puede matar la adaptación contextual. Un signal.warn + risk en un campo de formulario no es igual que en una operación financiera. La gramática orienta. No sustituye al juicio.

DISTINCIÓN

Un token semántico no debe congelar la experiencia. Debe impedir que se confundan funciones distintas.

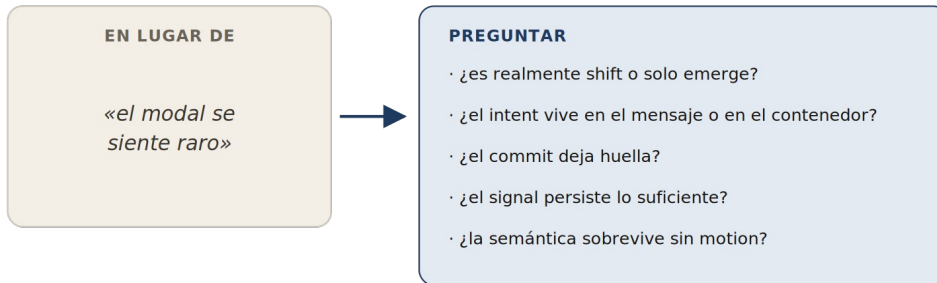
§ 11

Cómo usarlo en revisión de diseño

En lugar de decir «el modal se siente raro» podemos preguntar: ¿es realmente shift o solo emerge? ¿el intent vive en el mensaje o en el contenedor? ¿el commit deja huella? ¿el signal persiste lo suficiente? ¿la semántica sobrevive sin motion?

FIGURA 36.4

Cómo usarlo en revisión de diseño



La gramática convierte una sensación difusa en preguntas concretas que el equipo puede responder.

§ 12

De la gramática al sistema sin dogmatismo

El peligro de toda gramática es convertirse en policía. Este libro no propone eso. La pregunta no es «¿hemos aplicado la taxonomía de forma perfecta?» sino «¿hemos evitado confundir eventos perceptivamente distintos?»

§ 13

Síntesis

Convertir la gramática en sistema no significa construir un framework universal. Significa introducir una capa de lenguaje donde antes solo había componentes, estilos y efectos.

DISTINCIÓN

La gramática no obliga a una implementación concreta. Obliga a distinguir funciones que la implementación no debería mezclar.

PRINCIPIO

Nombra eventos antes de nombrar efectos.

TABLA 36.1 Del naming por efecto al naming semántico

NAMING HABITUAL	PROBLEMA	NAMING SEMÁNTICO
redAlert	color como significado	signal.alert + threat
successToast	componente + color	commit.save + affirm
dangerButton	intent en componente	commit.delete + loss
loadingSpinner	canal como evento	sustain.progress
shakeError	efecto como semántica	signal.warn + risk
modalDanger	contenedor como amenaza	shift.enter-mode + signal.warn + threat

TABLA 36.2 Preguntas de revisión y qué detectan

PREGUNTA	DETECTA
¿Qué evento ocurre?	ambigüedad de familia
¿Hay intent?	evaluación ausente o excesiva
¿Dónde vive el intent?	contenedor coloreado
¿Qué canal domina?	sobrecarga o fragilidad
¿Qué queda como huella?	commit débil
¿Qué pasa sin color?	dependencia cromática
¿Qué pasa con reduced motion?	pérdida de causalidad
¿El evento es frecuente?	fatiga sensorial
¿Hay cierre?	sustain infinito

TABLA 36.3 Niveles de adopción del sistema y su uso

NIVEL	USO
Conceptual	conversaciones y revisión
Documental	guías, fichas, ejemplos
Operativo	tokens, eventos, componentes

Cierre de la Parte IV

Con este capítulo se cierra la Parte IV, y con ella el recorrido operativo del libro: la gramática convertida en composiciones, matrices, rangos, protocolos de validación y decisiones de sistema. Lo que queda no es un apéndice técnico, sino la pregunta con la que todo empezó. A ella vuelve el Cierre.

CIERRE

Cierre

Hacia una interfaz con gramática perceptiva propia

Este libro empezó con una incomodidad, no con una certeza. La incomodidad de trabajar durante años construyendo interfaces y sentir que muchas decisiones importantes quedaban repartidas en capas separadas: componentes, estados, estilos, animaciones, sonidos, accesibilidad, validaciones, loaders, modales, notificaciones, tokens, eventos técnicos.

El usuario no percibe «un componente». No percibe «un token». No percibe «un transition-duration». Percibe otra cosa: algo respondió, algo empezó, algo sigue ocurriendo, algo se guardó, algo reclama atención, algo puede perderse, algo ya se perdió, algo apareció, algo cambió de marco, algo quedó completado. La pregunta inicial era sencilla, pero difícil de formular:

LA PREGUNTA DEL LIBRO

¿Por qué no tenemos un lenguaje más claro para hablar de lo que ocurre cuando una interfaz responde?

§ 1

Lo que este libro ha intentado hacer

Este libro no ha intentado fundar una teoría definitiva de la interfaz. Ha intentado algo más concreto: nombrar la capa de eventos perceptivos que ya existe entre el sistema y el usuario. Para nombrarla, ha propuesto una gramática: evento → familia semántica → verbo/fase → evaluación → intent → canales de expresión → lectura unificada.

§ 2

La interfaz como contexto operativo perceptible

La primera decisión fue dejar de entender la interfaz solo como superficie. Una interfaz es la dimensión perceptible de un contexto operativo. El componente sigue importando, pero deja de ser la unidad central del significado: lo que importa no es solo qué objeto hay en la interfaz, sino qué evento hace perceptible.

§ 3

El evento como unidad básica de cambio

La segunda decisión fue tomar el evento como unidad básica. No el evento técnico del navegador, ni el callback, ni el estado interno: el cambio perceptible que modifica la relación del usuario con el sistema.

§ 4

Las familias semánticas

Para ordenar esos eventos, el libro propuso ocho familias como ocho preguntas perceptivas recurrentes:

- **contact** — ¿me has sentido?
- **commit** — ¿qué resultado quedó?
- **signal** — ¿debo atender ahora?
- **handle** — ¿lo tengo yo?
- **emerge** — ¿qué hay disponible?
- **shift** — ¿dónde estoy ahora?
- **sustain** — ¿sigue vivo el proceso?
- **delegate** — ¿quién actúa ahora?

§ 5

El intent más allá del color

Después vino la pregunta evaluativa. Durante años hemos tratado success, warning, danger e info como si fueran una taxonomía suficiente, pero gran parte de esa estructura procede de una herencia cromática. El libro propuso separar el intent del color: un intent es una región evaluativa del evento —neutral, affirm, fulfill, risk, threat, loss—. Un autoguardado no es una culminación; un riesgo corregible no es una amenaza activa; una amenaza no es una pérdida consumada. El color puede reforzar todo eso, pero no debe definirlo.

§ 6

Los canales como materia sensible

Un evento no existe en abstracto: debe materializarse por canales —tiempo, motion, presencia, profundidad, forma, color, sonido, háptica—. Pero ningún canal debería cargar solo con todo el significado. Una semántica robusta puede migrar. La accesibilidad no es posterior a la semántica: es su prueba.

§ 7

Composición, matrices y validación

La última parte intentó convertir la gramática en herramienta: composiciones, matrices, rangos, fichas de migración, antipatronos y protocolos de validación. No como burocracia, sino como formas de hacer una pregunta simple: ¿la interfaz comunica el evento que cree comunicar?

§ 8

Lo que este libro viene a completar

Este libro no nace contra la teoría de interfaces existente. No intenta corregir desde fuera lo que el diseño de interacción, la usabilidad, la accesibilidad, la arquitectura de información, los sistemas de diseño, la ergonomía, la psicología de la percepción o la investigación HCI ya han construido. Existen teorías, métodos y guías muy desarrolladas para pensar la interfaz: cómo organizar información, diseñar flujos, construir componentes, hacer sistemas consistentes, asegurar accesibilidad, evaluar usabilidad.

Este libro no pretende sustituir nada de eso; su intención es más limitada y más concreta. Durante años de trabajo en front-end, algunas preguntas seguían apareciendo sin encontrar un lenguaje suficientemente preciso: ¿qué diferencia hay entre que el sistema reciba una acción y que confirme un resultado? ¿Por qué una espera necesita sentirse viva? ¿Cuándo una aparición es solo presencia y cuándo se convierte en señal? ¿Por qué una amenaza no debería expresarse igual que una pérdida? ¿Cómo migra el significado si el usuario reduce motion, no oye sonido o no distingue color?

Estas preguntas no niegan la teoría existente. Nacen en una zona que muchas veces queda entre varias disciplinas: entre componente y experiencia, entre estado técnico y percepción, entre efecto visual y significado, entre accesibilidad y semántica, entre motion y evento, entre color e intent. Este libro intenta trabajar precisamente ahí. No redefine la interfaz desde cero: propone una capa complementaria, una gramática perceptiva del evento.

DISTINCIÓN

Este libro no viene a reemplazar la teoría de interfaces. Viene a completarla desde la pregunta por el significado perceptivo del evento.

§ 9

Lo que sí propone

Propone que la interfaz tiene una capa semántica que merece ser nombrada. Propone que entre el sistema técnico y la experiencia del usuario hay eventos perceptivos que pueden organizarse. Propone que el intent no debería vivir en el color, sino en la evaluación del evento. Propone que los canales deben coordinarse alrededor de una lectura única. Propone que la accesibilidad es una prueba de si el significado estaba bien distribuido. Y propone que el trabajo de interfaz necesita un lenguaje más preciso para hablar de lo que ocurre.

§ 10

Volver al programador front-end

Al final, este libro vuelve al lugar del que salió: a un programador front-end enfrentado a decisiones aparentemente pequeñas. ¿Este botón solo responde o confirma? ¿Este loader tranquiliza o parece bloqueo? ¿Esta alerta es riesgo o amenaza? ¿Esta eliminación avisa o ya registra pérdida? ¿Este modal es marco o mensaje? ¿Qué cree el usuario que acaba de ocurrir?

Ninguna de esas preguntas es menor. Este libro ha intentado ofrecer un vocabulario para pensarlas con más claridad: no para eliminar la intuición, sino para darle estructura.

§ 11

Una gramática provisional

Toda gramática de interfaz debe ser provisional. Las plataformas cambiarán, los dispositivos cambiarán, la IA modificará muchas interacciones, la háptica será más rica, las interfaces espaciales traerán otros problemas, la accesibilidad seguirá ampliando el campo. Por eso esta gramática no debe cerrarse sobre sí misma: quizá algunas familias cambien, quizá aparezcan nuevas, quizá algunos intents se dividan, quizá

otros equipos encuentren mejores nombres. Eso no debilita el proyecto; lo mantiene vivo.

Una gramática no existe para impedir que el lenguaje evolucione. Existe para hacer posible una conversación más precisa.

§ 12

Cierre

Si este libro tiene algún valor, no está en haber resuelto definitivamente la interfaz. Está en haber intentado formular con claridad una pregunta que aparece una y otra vez en la práctica: ¿cuándo un cambio en la interfaz deja de ser efecto y se convierte en significado?

Este libro empezó con fragmentación —componentes por un lado, estados por otro, motion por otro, color por otro, sonido por otro, accesibilidad por otro— y termina con una propuesta de unión: evento, familia, intent, canales, accesibilidad, validación. No como sistema cerrado, sino como lenguaje compartido.

PRINCIPIO

La interfaz no solo debe funcionar. Debe hacer perceptible lo que ocurre.

PRINCIPIO

Este libro no intenta cerrar la teoría de la interfaz. Intenta ofrecer un lenguaje para una capa que ya estaba operando sin nombre.

Las ocho preguntas de las familias, con la síntesis completa de la gramática, quedan recogidas en la referencia canónica del Apéndice B.

APÉNDICE A

Capa semántica declarativa

Esta gramática no debe implementarse como un sistema de variantes visuales. No se trata de reemplazar `<Button variant="danger">` por otra variante con otro nombre.

La gramática de eventos propone una capa declarativa por encima de componentes, tokens y patrones. Esa capa declara qué evento ocurre, a qué familia pertenece, si porta intent y qué canales de expresión deberían participar.

La secuencia conceptual es:

evento

→ familia

→ verbo / fase

→ intent si procede

→ firma de expresión

→ componente / patrón / tokens

En una implementación real, esta capa puede representarse como un DSL, un JSON, un mapa de morfos, una convención de documentación o una herramienta interna de sistema de diseño.

Ejemplo conceptual:

{

«event»: «deleteConfirmationAlert»,

```

«family»: «signal»,
«verb»: «alert»,
«intent»: «threat»,
«expression»: {
«channels»: [«presence», «depth», «shape», «color»],
«persistence»: «until-decision»,
«intensity»: «high»,
«ally»: {
«focus»: «contained»,
«announcement»: «assertive»
}
}
}

```

La regla técnica más importante es que el intent no se hereda de forma implícita. Se declara en el evento evaluable que lo porta. Si un diálogo comunica amenaza, el morfo debe decidir qué evento declara ese intent: signal.alert o commit.delete. La apertura del diálogo (emerge.open) no lo porta: es marco, no mensaje.

El runtime no debe adivinar. La semántica debe estar declarada.

Formalización: el Semantic Event Spec

Lo anterior puede formalizarse, y conviene hacerlo. Una capa declarada permite algo que una convención tácita no permite: comprobar por máquina si una interfaz declara lo que cree declarar.

El modelo es deliberadamente pequeño. Un documento declara un mapa de nodos. Un nodo puede ser dos cosas. La primera: un evento atómico, con su familia, su verbo y, si procede, intent, fase, actor y firma de expresión. La segunda: una composición de otros nodos, ya sea una secuencia temporal (sequence, lo que este libro nota con $\rightarrow$) o una co-ocurrencia (simultaneous: eventos que suceden a la vez).

Esa estructura se lee en prosa con la notación que ya hemos usado:

family.verb

family.verb + intent

$a \rightarrow b \rightarrow c$

$a + b$ (simultáneos)

La forma declarativa —un JSON validable contra un esquema— es la representación canónica; la notación abreviada es su lectura humana.

Lo importante no es el formato. Lo importante es que el esquema hace cumplir las reglas de la gramática, en lugar de confiar en la disciplina de quien escribe. Las ocho familias y los seis intents son conjuntos cerrados: un success o un danger heredados del semáforo se rechazan. La regla marco $\neq$ mensaje queda inscrita: emerge, shift y contact no pueden portar un intent distinto de neutral. Por eso emerge.open + threat —la combinación que este libro corrige una y otra vez— es, literalmente, un documento inválido. Hay una sola salida: declararlo como la excepción aparición-que-advierte, con su justificación. El esquema no prohíbe la excepción; la hace cara y auditable. Y la autoría pertenece a delegate: el campo actor, que declara quién lleva la iniciativa, solo es válido en esa familia.

PRINCIPIO

Una capa declarada convierte un principio en una comprobación. La regla que el libro defiende en prosa, el esquema la rechaza en la práctica.

Los verbos, en cambio, no se cierran. El vocabulario canónico de cada familia es una recomendación, no una cárcel: dominios especializados —trading, medicina, CAD, sistemas críticos— pueden necesitar los suyos. El esquema lleva una versión (`specVersion`); esta es la 0.1. Las familias y los intents son el núcleo estable; la vía de extensión es lo que las versiones siguientes desarrollarán.

Esta capa no vive solo en el papel. Su forma operativa se publica como especificación abierta acompañante: el esquema, el vocabulario de verbos, ejemplos validables y el mapeo a tokens semánticos, variables de Figma y ARIA. El objetivo es que un equipo pueda decir, con sentido literal: «mapeamos nuestros eventos al spec».

La implementación de referencia que materializa este esquema —con el guardarraíl de la excepción y los mecanismos que la gramática exige— se examina en el Apéndice D.

APÉNDICE B

Firmas de expresión y matriz mínima

Una firma de expresión es la configuración concreta de canales, intensidad, duración, persistencia y huella con la que un evento se hace reconocible en un sistema concreto.

La familia y el intent dicen qué evento ocurre y cómo debe valorarse. La firma de expresión dice cómo se materializa esa lectura.

TABLA B.1 Matriz mínima de firmas de expresión

Evento	Canales principales	Canales secundarios	Intensidad	Huella	Accesibilidad
contact.press	tiempo, forma	motion, háptica	mínima	no	foco visible
commit.save + affirm	presencia, tiempo	color suave	baja	breve	texto si procede
commit.complete + fulfill	presencia, motion, forma	sonido opcional	media	sí	estado persistente
signal.warn + risk	forma, texto, presencia	color	media	hasta corrección	relación con campo
signal.alert + threat	presencia, profundidad, forma	sonido opcional	alta	hasta decisión	anuncio urgente
commit.delete + loss	presencia, forma, tiempo	color, motion	alta	sí	deshacer / mensaje persistente
delegate.act	presencia, texto, sustain	motion	media	hasta retorno	cancelar / revisar

Referencia canónica: familias e intents

Las dos tablas transversales del libro, reunidas aquí para consulta rápida. Los capítulos remiten a esta página en lugar de repetirlas.

TABLA B.2 Síntesis de las ocho familias

Familia	Pregunta perceptiva	¿Acepta intent?	Fase típica	Canales dominantes	Antipatrón capital
contact	¿me has sentido?	No	inmediata (<100 ms)	tiempo, motion, forma, háptica	celebrar el contacto como si fuera resultado
commit	¿qué resultado quedó?	Sí	al resolverse la acción	presencia, forma, tiempo, color	success prematuro: celebrar lo que aún no ocurrió
signal	¿debo atender ahora?	Sí	cuando el sistema inicia	presencia, color, forma, sonido	alarma para todo: si todo es urgente, nada lo es
handle	¿lo tengo yo?	Parcial (el desenlace)	durante la manipulación	motion, profundidad, forma, háptica	arrastre ciego: no se ve qué está cogido
emerge	¿qué hay disponible?	No	al aparecer o retirarse	presencia, profundidad, motion	cargar el intent en la apertura
shift	¿dónde estoy ahora?	No	al cambiar de marco	motion, presencia, profundidad	teletransporte: cambiar sin orientar
sustain	¿sigue vivo el proceso?	Parcial (el progreso, no el desenlace)	mientras dura	tiempo, motion, forma	spinner eterno sin estado ni huella
delegate	¿quién actúa ahora?	No por defecto (lo portan sus consecuencias)	al cambiar la iniciativa	presencia, texto, tiempo	autoría invisible: actuar sin que se note quién actuó

TABLA B.3 Los seis intents

Intent	(ES)	Qué declara	Valencia / Activación	Expresión típica	Riesgo de mal uso
neutral	neutro	Algo debe notarse, sin tono	neutra / baja	presencia suave, texto	Usarlo como «no lo hemos pensado»
affirm	confirmatorio	Se aplicó; todo en orden	positiva / baja	color suave, presencia breve	Celebrar de más; confundirlo con fulfill
fulfill	resolutivo	Se cerró una tensión u objetivo	positiva / media	motion, forma, presencia	Gastarlo en micro-acciones cotidianas
risk	corregible	Hay problema y puedes corregirlo	negativa / media	forma, texto persistente junto al campo	Vestirlo de threat: alarmismo
threat	amenazante	Aún puedes evitarlo: atiende	negativa / alta	presencia, profundidad, sonido opcional	Cargarlo en el marco y no en la señal
loss	perdido	Ya ocurrió: registra y repara	negativa / baja-media	tiempo, forma seca, huella y deshacer	Sonar como alarma cuando ya no hay nada que evitar

Paletas iniciales: sonido y háptica

Los Capítulos 17 y 18 reclaman un vocabulario que la web no tiene, y reclamarlo sin ofrecer un punto de partida dejaría la deuda a medias. Estas dos paletas lo son: firmas orientativas por intent, con la filosofía de rangos del Capítulo 32 y números tomados de una implementación real —

la del Apéndice D—. No son recetas: son el primer borrador de una materia.

TABLA B.4 Paleta sonora inicial por intent

INTENT	FIRMA SONORA ORIENTATIVA
neutral	Silencio o transitorio mínimo: limpio, ganancia muy baja ($\leq 0,05$), sin contorno marcado.
affirm	Breve (≈ 80 – 140 ms), limpio, registro medio-agudo, ganancia baja, contorno plano o leve arco.
fulfill	Ascendente y resolutivo: algo más largo (≈ 150 – 200 ms), admite doble nota, ganancia moderada.
risk	Seco y contenido: ataque claro, duración corta, aspereza leve, sin insistencia ni repetición.
threat	Ataque casi inmediato (≈ 4 ms), aspereza alta — dos órdenes de magnitud sobre el resto de la paleta—, ganancia máxima, saliente mientras dure la situación.
loss	Grave (≈ 500 Hz), descendente, seco: registra sin alarmar, con ganancia contenida. El silencio posterior es parte de la firma.

TABLA B.5 Paleta háptica inicial por intent

INTENT	FIRMA HÁPTICA ORIENTATIVA
neutral	Ausencia, o un pulso mínimo.
affirm	Un pulso suave, único.
fulfill	Un pulso más definido, o un doble pulso suave.
risk	Un pulso contenido, sin repetición.
threat	Vibración fuerte o doble, con patrón reconocible, duración limitada y posibilidad de silenciar.

loss	Un pulso seco y pesado, único; nunca vibración sostenida.
------	---

Dos reglas de la práctica completan las tablas. La primera es la del *gesto continuo*, para cuando el sonido acompaña una manipulación — un arrastre, un deslizador—. La posición gobierna la altura tonal; la velocidad, el brillo y la ganancia; la dirección, el contorno. Y la masa perceptiva del objeto gobierna el timbre: mover un panel entero suena más grave, menos brillante y más leñoso que mover el pulgar de un deslizador. Todo ello en susurro — ganancias de 0,01 a 0,03, muestreadas al ritmo de la percepción (Capítulo 25)—.

La segunda es la del *silencio canónico*. Los eventos de frecuencia altísima —el interruptor cotidiano, el tooltip que aparece al pasar— declaran el silencio como su firma por defecto: una ganancia base negativa que anula la voz de la familia. Pero la anulación es perforable: los deltas evaluativos la atraviesan, y el mismo interruptor que calla a diario suena cuando el cambio es destructivo o culminante. El silencio no es ausencia de diseño: es la firma que la frecuencia exige (Capítulo 32), con la evaluación como única excepción.

Y una nota de identidad. La firma paramétrica es el contrato semántico del sonido; la voz concreta puede ponerla una síntesis o una muestra de audio con carácter de producto. La disciplina es que toda muestra declare su firma como respaldo: así la identidad sonora de una marca convive con la gramática en lugar de sustituirla.

APÉNDICE C

Tres casos de estudio

Caso 1 — Formulario con error corregible

Un usuario envía un formulario y un campo obligatorio está vacío. La interfaz no debería tratar este caso como amenaza. Todavía hay margen de acción.

Secuencia:

contact.press

→ signal.warn + risk

→ focus.field

→ commit.block + risk

La firma de expresión es contenida: el mensaje aparece junto al campo, con forma clara y color moderado, y persiste hasta que el error se corrige. Si el envío queda bloqueado, el foco se orienta hacia el campo que lo bloquea.

Caso 2 — Eliminación con confirmación y deshacer

Un usuario intenta eliminar un elemento. Antes de confirmar, la interfaz introduce la amenaza. Después de confirmar, comunica pérdida y mantiene una huella si existe deshacer.

Secuencia:

contact.press

- emerge.open
- signal.alert + threat
- commit.delete + loss
- emerge.present-undo

La firma de expresión sigue ese arco. Antes: un diálogo claro, con el foco contenido y las acciones bien diferenciadas. Después: un mensaje que persiste tras la eliminación y una opción de recuperación mientras siga siendo válida.

Caso 3 — Sistema que propone y aplica cambios

Un sistema propone modificar varios campos, espera revisión y aplica cambios tras autorización. El problema principal no es solo el resultado, sino quién actúa y cuándo.

Secuencia:

- delegate.plan
- signal.warn + risk
- delegate.review
- delegate.authorize
- delegate.act
- sustain.processing
- commit.apply + affirm

La firma de expresión responde a esa pregunta: el sistema explica el alcance de lo que propone, mantiene visible su estado y deja cancelar en todo momento. La revisión es explícita, y el resultado aplicado se confirma.

APÉNDICE D

De la gramática al código: el caso Sema

A lo largo del libro, la gramática se ha mantenido deliberadamente agnóstica: familias, intents y canales se han descrito como categorías perceptivas, no como piezas de un producto. Este apéndice hace, una sola vez, el movimiento contrario: muestra la gramática soportando el peso de un sistema real. El sistema es *UIX*, una librería de interfaz cuyo módulo semántico —Sema— implementa las familias, los intents, los verbos y los canales de este libro. Se presenta aquí como lo que es: *una* implementación posible, no *la* implementación.

Conviene leerlo con doble propósito. El primero, ver que la gramática aguanta: que sus categorías no solo ordenan ejemplos de libro, sino declaraciones compilables, resolución en tiempo real y decisiones incómodas. El segundo, aprender a separar lo esencial de lo accidental: qué decisiones viajarían a cualquier otro sistema —del más artesanal al más industrial— y cuáles son rasgos de esta implementación concreta. La última sección hace explícita esa separación.

Qué es este apéndice (y qué no)

Este apéndice no es documentación de *UIX*: las versiones cambian, los nombres se mueven, y ninguna página impresa debe competir con la documentación viva de un proyecto. Tampoco es un requisito: todo lo anterior del libro se sostiene sin él, y la gramática no cambia si esta implementación desaparece mañana.

Lo que sí es: la prueba de carga de la teoría. Cada mecanismo citado aquí existe, compila y tiene pruebas automatizadas en el momento de escribir esto. Y es también la fuente de una historia que el lector ya conoce: el recuadro del Capítulo 9 —«lo que la práctica corrigió»— salió de este sistema. Aquí se cuenta con el detalle que aquel recuadro no podía permitirse.

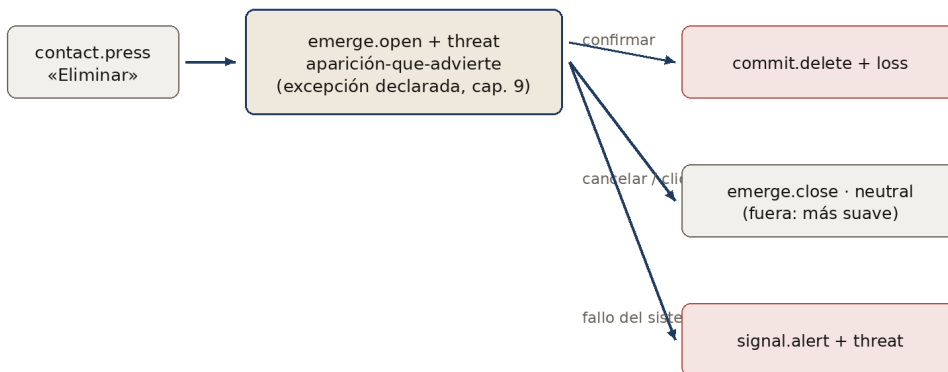
El caso: eliminar un proyecto

El hilo conductor será un flujo completo, elegido porque concentra las decisiones más difíciles de la gramática: el usuario elimina un proyecto con contenido dentro.

El usuario pulsa «Eliminar». El sistema registra el gesto —*contact.press*, respuesta inmediata, dentro de la ventana de agencia del Capítulo 12—. Y entonces se abre un diálogo de confirmación destructiva. Aquí aparece la primera decisión fuerte: esa apertura no es neutral. El diálogo se abre *solo* porque hay una acción destructiva pendiente. Apliquémosle el test sustractivo del Capítulo 9: ¿quitando esta superficie, la advertencia sobrevive en otro sitio? La respuesta es no: la aparición es la única advertencia. La implementación declara por tanto *emerge.open* con tono de advertencia, como excepción nombrada y justificada. El primer momento perceptivo —la interrupción misma— ya porta la evaluación.

FIGURA D.1

El caso: eliminar un proyecto



Un solo evento de cierre; la familia la decide la causa. Después: huella y deshacer viven fuera del diálogo.

El cierre polimórfico: la ocurrencia adopta la familia de su miembro más evaluable (cap. 30).

Dentro del diálogo, el texto hace su trabajo de canal: título con la consecuencia, descripción con el alcance, foco atrapado en la decisión. Y el flujo tiene tres salidas, que la figura anticipa. Si el usuario confirma, el cierre porta la consecuencia: un *commit* —en este flujo, *commit.delete + loss*—. Si cancela, el cierre es una retirada sin carga: *emerge.close*, neutral; y si esa cancelación fue un clic fuera de la superficie —un gesto de baja intencionalidad—, la misma salida suena más suave. Si la operación falla al ejecutarse, el cierre porta la alerta: *signal.alert + threat*.

Después del cierre, la superficie ya no existe; pero la pérdida sí. La huella y la reparación —el mensaje de resultado, el deshacer temporal— viven fuera del diálogo: en una superficie persistente y en el canal de anuncio accesible. Es exactamente lo que pedía el caso 3 del Capítulo 33. La ocurrencia se disipa; el estado queda.

La declaración: el contrato semántico del componente

En esta implementación, cada componente declara sus eventos en un contrato —familia, verbo, objetivo, orden respecto de la mutación y, si procede, intent—. Ese contrato es la fuente única de la semántica del componente, y es compilable: renombrar un evento o una parte rompe las reglas que dependen de él en tiempo de compilación, no en producción. La gramática deja de ser una convención de equipo y pasa a ser un tipo.

Dos detalles del contrato del diálogo merecen atención. El primero: el intent de la apertura no acepta cualquier valor. Está restringido, por tipo, a *neutral*, *risk* y *threat* — sin celebraciones y sin *loss*. La restricción codifica dos doctrinas a la vez. La primera es el test sustractivo: esta aparición solo puede portar tonos de advertencia, porque advertir es su única razón de existir. La segunda es una regla que el libro insinuaba y la implementación escribió con todas sus letras: la pérdida irreversible no se confirma en un diálogo. Se advierte antes y se registra después. No hay paleta de pérdida para una superficie de confirmación.

El segundo: la excepción es cara. El contrato exige, junto al intent, una justificación escrita —y un validador la comprueba: un evento estructural con tono y sin justificación no compila—. Es el guardarraíl del Capítulo 9 convertido en herramienta: la excepción existe, pero nunca en silencio.

El contrato declara también el orden señal↔mutación del Capítulo 30. La apertura es *posterior*: la superficie primero existe y luego se siente — anticipar la señal retrasaría la llegada del propio diálogo—. El cierre es *anticipado*: la señal suena antes del desmontaje, porque después ya no habría dónde ocurrir.

Una salida, tres familias: el cierre polimórfico

La decisión más elegante del contrato es el cierre. El libro, ante «cerrar tras guardar», habría compuesto una secuencia: primero el resultado (*commit.save* + *fulfill*), después la retirada (*emerge.close*). La implementación observó algo que la secuencia oculta: en un diálogo, resultado y retirada son perceptivamente simultáneos — el cierre es la forma en que el usuario percibe el resultado. Separarlos produce dos señales donde la experiencia tiene una.

La solución: un único evento de cierre cuya familia se resuelve según la causa. Cancelar produce *emerge.close*; confirmar produce el *commit* correspondiente; fallar produce *signal.alert* + *threat*. La ocurrencia adopta la familia del miembro más evaluable de la pareja — que es, exactamente, el criterio de dominancia del Capítulo 30 aplicado en tiempo de diseño en lugar de en tiempo de ejecución.

¿Y cómo sabe la salida por qué ocurre? La causa se escribe como estado —«guardado», «cancelado», «descartado desde fuera», «fallido»— un instante *antes* de que el estado principal cambie a cerrado. Es la doctrina de las dos superficies del Capítulo 12, con un matiz que el libro no tenía. El estado no solo persiste el resultado: puede portar la *causa* de la transición, para que la señal de salida sea evaluable según por qué se va.

La resolución: las matrices hechas ejecutables

Declarado el evento, falta decidir cómo suena, vibra y cuánto vive. La implementación resuelve cada señal atravesando capas con precedencia. Y esas capas son, casi línea a línea, las matrices del Capítulo 31 con un

orden de resolución. La base de la familia aporta la firma estructural. La modulación del intent aporta la firma evaluativa. El evento concreto puede ajustar; el sistema puede corregir globalmente. Y el contexto —las reglas que miran dónde ocurre el evento— decide lo último. Con una convención que evita ambigüedades: la modulación del intent *suma* (es un modificador sobre la base), las capas superiores *asignan* (deciden el valor final).

Sobre esa cascada rige una regla de reparto que merece nombre propio: el contexto da carácter; la firma evaluativa no se toca. En un evento evaluable, la energía, la altura y el contorno pertenecen al intent. Si una regla contextual los reescribiera, *risk*, *threat* y *loss* volverían a sonar iguales, y se perdería la mercancía del Capítulo 10. En un evento estructural sin intent, en cambio, el contexto sí puede dar forma a la materia. El propio diálogo lo ilustra. Su cierre estructural invierte el contorno de la apertura: ascendente al entrar, descendente y más grave al salir; aire entrando, aire saliendo. Es la direccionalidad del Capítulo 13 traducida al canal sonoro.

El pack del diálogo añade un modulador que el Capítulo 32 no llegó a nombrar: la intencionalidad del gesto. El cierre por clic fuera —un gesto ambiguo, de baja deliberación— suena más suave que el cierre por botón. Junto a la consecuencia y la frecuencia, la intencionalidad de la causa gradúa la intensidad: cuanto menos deliberado el gesto, menos presencia merece su eco.

La materialización: una superficie, dueños separados

Queda el último tramo: convertir la señal resuelta en percepción. Aquí la implementación tomó una decisión arquitectónica que es doctrina del

Capítulo 20 hecha estructura. Durante la ventana perceptiva del evento, la ocurrencia se proyecta como marcas transitorias sobre su objetivo: qué evento, de qué familia, con qué intent. Y *todos* los consumidores leen esa misma superficie. La hoja de estilos tiñe y anima leyéndola. El canal sonoro y el háptico se suscriben leyéndola. El canal de anuncio accesible deriva de ella su prioridad. Un evento, varios canales, una lectura — como arquitectura, no como aspiración.

Las marcas transitorias conviven con el estado persistente sin pisarse. La ocurrencia dice «esto acaba de pasar» y se disipa al agotar su ventana, del orden de un cuarto de segundo por defecto y graduada por familia. El estado dice «esto es ahora verdad» y permanece. Y la ventana se respeta entera: la señal completa su vida antes de que el cambio estructural la atropelle, con las dos excepciones ya declaradas por el orden del evento.

Cierra la materialización una política de errores con doctrina dentro: la capa perceptiva es ornamental — si el audio falla o la vibración no existe, la operación del usuario jamás se aborta. Con una excepción que el Capítulo 33 obliga a nombrar: la señal crítica accesible no es ornamento, y su fallo no puede absorberse en silencio.

Lo que la práctica corrigió

El recuadro del Capítulo 9 contaba que esta excepción —la aparición-que-advierte— no nació en la teoría: la impuso la práctica. La versión larga es esta. La doctrina original del sistema, heredada del libro, asumía que solo las familias evaluables podían declarar intent; las estructurales eran neutrales por definición. Al construir el diálogo de confirmación destructiva, mantener neutral la apertura dejaba mudo el primer momento perceptivo. El tono llegaba tarde, con el mensaje interior. Y esa mudez se leía como infra-reacción ante un contexto grave.

La corrección no fue abrir la mano: fue nombrar la excepción, darle una prueba —el test sustractivo— y hacerla cara —justificación obligatoria, validada por herramienta—. El defecto sigue siendo firme: la evaluación viaja en la figura, no en el marco. Pero la gramática demostró que sabe hacer lo que su Cierre promete: evolucionar con nombre, prueba y coste, en lugar de romperse o ignorar el caso.

La práctica dejó una segunda corrección, más incómoda, y merece la misma franqueza. El libro doctrina el diálogo modal como *shift.enter-mode* — cruzar a otro régimen; el ejemplo del Capítulo 8 lo usa así—. El contrato real del componente declara *emerge.open*, y su cierre polimórfico admite *emerge*, *commit* y *signal*: *shift* no está en la lista. La desviación está registrada en el cuaderno de desviaciones del proyecto, con su razón. En el componente genérico pesó más la aparición que el cruce de marco. Y el cruce quedó reservado a los usos que de verdad bloquean el fondo y capturan el foco — el mismo componente puede ser una cosa u otra según cómo se use.

La doctrina del libro no cambia: el modal pesado sigue siendo *shift*. Lo que la práctica enseñó es que la frontera *emerge/shift* no pasa entre componentes, sino por dentro de ellos. Y algo más general: una gramática con cuaderno de desviaciones es más creíble que una sin él — cada desviación registrada es una regla que se tomó en serio.

Lo que el libro exigió al código

La corrección también fluyó en sentido contrario. Al auditar la implementación con la vara del propio libro, aparecieron seis mecanismos que la teoría exige y el código no tenía. Hoy los seis están implementados y probados:

- **La memoria de frecuencia (Capítulo 32): las repeticiones del mismo evento se atenúan progresivamente hasta un suelo — el guardado**

número cincuenta no suena como el primero. La amenaza está exenta: su función no admite fatiga.

- **El árbitro de dominancia (Capítulo 30): cuando dos señales se solapan, lo evaluable domina sobre lo estructural, la mayor activación sobre la menor y lo reciente sobre lo previo. La señal dominada conserva su ventana visual, pero calla sus canales sensoriales.**
- **El canal de anuncio como pieza de fábrica (Capítulos 19 y 33): las tecnologías de asistencia reciben la ocurrencia con prioridad derivada del intent — la amenaza interrumpe, el resto espera turno.**
- **La migración semántica como dato (Capítulo 19): la tabla canal-caído → destino-del-significado existe como estructura consultable, no como doctrina delegada a cada aplicación.**
- **Las reducciones simétricas (Capítulo 33): cada canal sensorial acepta los niveles pleno, reducido y apagado — el sonido con la misma dignidad que la háptica, no como caso especial.**
- **El coste de la excepción (Capítulo 9): el guardarraíl que hace imposible declarar tono en un evento estructural sin justificación escrita.**

La lección de esta lista importa más que la lista: la relación entre una teoría y su implementación no es de sentido único. El código falsifica a la teoría donde la teoría no miró; la teoría audita al código donde el código se conformó. Este apéndice existe porque las dos direcciones ocurrieron.

Lo que sigue abierto

La honestidad del libro obliga a cerrar con lo no resuelto. Tres casos, de mayor a menor calado.

El primero es una frontera del propio diálogo. El libro lee el modal bloqueante como cruce de marco —*shift.enter-mode*: cambia el régimen operativo, atrapa el foco, exige orientación—. Esta implementación lo

declara aparición —*emerge.open*— con foco atrapado. Ambas lecturas son defendibles: la superficie, en efecto, aparece; el régimen, en efecto, cambia. La gramática no exige que todas las implementaciones lean igual el caso frontera; exige que cada una elija, declare y sea consecuente. El desacuerdo, mientras esté declarado, es información: señala exactamente dónde la frontera *emerge/shift* necesita más trabajo teórico.

El segundo es un uso de la excepción que probablemente no pasa su propia prueba. Otro componente del sistema —una línea temporal que revela hitos— declara su evento de revelado con los seis intents posibles. Aplicado el test sustractivo, la evaluación casi seguro sobrevive sin el revelado: vive en el hito revelado, no en el hecho de revelarse. El caso está señalado dentro del propio código como tensión declarada. Y eso es, exactamente, lo que el guardarraíl existe para producir: excepciones visibles que fuerzan revisión, en lugar de excepciones silenciosas que erosionan la regla.

El tercero es de mantenimiento: el contrato entre un libro y un código que se citan mutuamente. Las citas por número de sección se pudren con cada edición; la solución —un mapa de anclas estables entre la doctrina y su implementación— es trabajo poco glorioso del que dependen las otras dos direcciones.

Anclas estables: cómo cita el código a este libro

Ese trabajo empieza aquí. Cada identificador de la tabla fija un lugar del libro — capítulo y sección, nunca páginas — para que la documentación y el código de cualquier implementación puedan citar la doctrina sin ambigüedad. Y sin pudrirse con las reimpresiones: mientras los §§ no cambien, las anclas aguantan. El mapa completo acompaña al libro como

documento aparte; esta tabla recoge las que un código tiene más motivos para citar.

TABLA D.1 Anclas estables: del código al libro

ANCLA	DESTINO
BK-FAMILIAS	Capítulo 8 · las ocho familias
BK-EVALUABILIDAD	Capítulo 9 · evaluable frente a estructural
BK-A1-EXCEPCION	Capítulo 9, §9 · la aparición-que-advierte y el test sustractivo
BK-INTENTS	Capítulo 10 · los seis intents
BK-THREAT-LOSS	Capítulo 10 · amenaza frente a pérdida
BK-DOS-SUPERFICIES	Capítulo 12 · ocurrencia y estado
BK-SONIDO-MATERIA	Capítulo 17 · nombrar el sonido por su materia
BK-HAPTICA-MATERIA	Capítulo 18 · primitivas hápticas por sensación
BK-MIGRACION	Capítulo 19 · migración semántica
BK-COORDINACION	Capítulo 20 · varios canales, un solo evento
BK-MUESTREO	Capítulo 25, §5 · muestrear al ritmo de la percepción
BK-SUSTAIN-ESTADO	Capítulo 28, §7 · sustain como estado con transiciones
BK-DELEGATE	Capítulo 29 · quién actúa
BK-INTERPRETE	Capítulo 29, §11 · la frontera del intérprete
BK-COMPOSICION	Capítulo 30 · orden señal-mutación y dominancia
BK-MATRICES	Capítulo 31 · familias × canales, intents × canales
BK-RANGOS	Capítulo 32 · rangos orientativos
BK-REFERENCIA	Apéndice B · tablas canónicas B.1-B.3
BK-D11	Apéndice D · la desviación emerge/shift del diálogo
BK-STREAMING	Apéndice E, caso 1 · el texto que fluye
BK-OPTIMISTA	Apéndice E, caso 2 · la interfaz optimista

Cómo leer este apéndice desde otro sistema

Quien construya sobre otra base —un framework distinto, otro lenguaje, o ninguna librería en absoluto— debería llevarse de aquí lo esencial y descartar lo accidental.

Es esencial declarar los eventos como contrato —familia, verbo, intent y orden señal↔mutación— en un lugar único y verificable. Es esencial respetar la ventana perceptiva del evento antes de pisarla con el cambio estructural. Es esencial mantener separadas la ocurrencia transitoria y el estado persistente. Es esencial resolver la expresión por capas con precedencia, repartiendo carácter y firma evaluativa entre contexto e intent. Es esencial someter toda excepción al test sustractivo y cobrarle justificación. Y es esencial dotar al motor, sea cual sea, de los seis mecanismos de la sección anterior. Porque no son rasgos de esta implementación: son requisitos que la gramática impone a cualquiera.

Es accidental: los nombres de los módulos y de los atributos; el reparto concreto de lo visual en hojas de estilo; los valores por defecto de las ventanas; el vehículo de tipos que hace compilables los contratos. Nada de eso viaja, y nada de eso importa: otra implementación con otros nombres que responda a las mismas preguntas es igual de fiel a la gramática.

PRINCIPIO

Una implementación no demuestra que la gramática sea verdadera. Demuestra que es construible — y que, al construirla, corrige y es corregida.

APÉNDICE E

Casos difíciles: donde la gramática suda

Los capítulos de este libro razonan sobre interfaces que ya sabemos mirar: formularios, diálogos, listas, procesos con final. Este apéndice mira las otras: las que el presente pone delante del lector y la gramática todavía no digiere del todo.

Cada caso sigue la misma disciplina. Primero, qué nombra hoy la gramática — casi siempre más de lo que parece—. Después, dónde se tensa: la costura exacta en la que las categorías dejan de bastar. Y al final, qué queda abierto, dicho sin disimulo. Donde existe práctica real, se cita: varios de estos casos dejaron cicatriz en la implementación de referencia del Apéndice D, y la cicatriz enseña más que la doctrina.

PRINCIPIO

Un caso donde la gramática suda enseña más que veinte donde funciona.

Caso 1 — El texto que fluye: el streaming de un modelo generativo

El usuario envía una pregunta y la respuesta no llega: va llegando. Durante segundos, el texto crece palabra a palabra, se detiene, sigue, a veces se corrige. Es hoy la interacción definitoria de la era que el Capítulo 29 anuncia, y el cuerpo del libro la despacha en una línea: `contact.submit · sustain.streaming · commit.complete`.

Esa línea nombra bien el esqueleto. Hay un contacto que registra la petición. Hay un proceso vivo — la implementación de referencia añadió *streaming* como verbo de sustain precisamente por este caso—. Y hay una resolución. La gramática no está muda: distingue la espera del resultado, exige que el flujo se perciba vivo y prohíbe celebrarlo antes de tiempo.

La primera costura es la superficie que crece. Cada palabra que aparece es, en rigor, una aparición — un micro *emerge.present*—. Pero puntuar cientos de apariciones por respuesta viola el muestreo perceptivo del Capítulo 25: el canal se inunda y el gesto se aplana. La lectura práctica es tratar el flujo como *un* sustain cuya superficie crece, no como mil emerge encadenados. Funciona, y sin embargo pierde algo: la llegada de cada fragmento *sí* se percibe, y la gramática no tiene nombre para esa granularidad intermedia.

La segunda costura es el commit. ¿Cuándo queda fijada una respuesta que puede corregirse mientras fluye? El usuario lee, y a veces actúa, sobre texto que aún no está comprometido — copia un párrafo que el modelo reescribe un segundo después—. Hay un commit percibido antes del commit real: el pariente directo de la interfaz optimista del caso siguiente. La disciplina mínima es de huella: distinguir perceptivamente el texto en tránsito del texto fijado. Y puntuar el cierre — el *commit.complete* — como evento propio, no como mera detención del flujo.

La tercera costura es la autoría. Mientras el texto fluye, ¿quién actúa? La pregunta de delegate tiene respuesta: actúa el agente, por cuenta del usuario. Lo que no la tiene es su corolario: ¿qué parte de lo mostrado es ya un acto y qué parte es borrador de un acto? Queda abierto si el flujo generativo merece una fase propia en la gramática — entre el sustain y el commit — o si basta la disciplina de huella. Este libro no lo zanja.

Caso 2 — La interfaz optimista

Enviar un mensaje y verlo aparecer al instante en la conversación. Marcar un favorito y ver el contador subir. La red aún no ha confirmado nada, pero la interfaz ya muestra el resultado. Es la interfaz optimista: la que muestra el commit antes de que exista su consecuencia.

La gramática tiene doctrina exacta contra esto. El orden del Capítulo 30: la señal de resultado va después del resultado. La implementación de referencia lo escribió como regla del contrato — la confirmación de guardado se declara posterior a la mutación: no se celebra lo que aún no existe—. La interfaz optimista viola esa regla a sabiendas, y a menudo con buenas razones: la latencia percibida se desploma y la conversación fluye.

El coste no es la violación: es el evento nuevo que aparece cuando el optimismo falla. La reconciliación. Un *affirm* ya entregado que hay que revocar: el mensaje que no salió, el favorito que no contó. En la gramática es un *commit.fail* + *risk* — pero llega sobre algo que el usuario dio por fijado, con el orden temporal invertido que el Capítulo 12 enseña a temer. La mayoría de las interfaces optimistas lo resuelven mal: un parpadeo, un mensaje que desaparece sin explicación, un contador que baja solo.

Lo que la gramática sí sabe decir: si se anticipa, que se declare. Un commit provisional no debería vestir la firma completa del commit — es, en rigor, un sustain con forma de commit, y su huella puede decirlo: el mensaje en tránsito ligeramente distinto del entregado—. Y la revocación merece diseño de evento de primera clase: huella, explicación y reparación, no un deshacerse en silencio.

Queda abierto el nombre. ¿Es el commit provisional un verbo propio — un *commit.pending* que las familias no tienen—, o una mentira útil que sustain ya cubre? La implementación de referencia eligió la disciplina del orden y no nombró el patrón. El patrón, mientras tanto, gobierna media web.

Caso 3 — Skeletons: la espera con forma de promesa

Los bloques grises con la silueta del contenido futuro se han vuelto la espera por defecto de la web. La gramática los lee sin esfuerzo aparente: son *sustain* — un proceso vivo, sin evaluación—. La implementación de referencia lo confirma por contrato: sus indicadores de espera no aceptan intent. La espera no es evaluable; lo evaluable llegará con el resultado.

La costura está en la forma. Un spinner sostiene; un skeleton, además, promete. Al tomar la silueta del contenido — tantas líneas, una imagen aquí, un título allá—, el skeleton hace algo que ningún *sustain* del libro hacía: es la aparición anticipada de otra cosa. Un *emerge.present* prestado, de algo que aún no existe.

Y las promesas se rompen. Si lo que llega no coincide con la silueta — el diseño salta, la imagen no estaba, las tres líneas eran seis—, el usuario percibe un fallo que ningún evento declaró. No hay *commit.fail*: nada falló en el sistema. Falló la forma prestada. La gramática no tiene antipatrón para la promesa incumplida de un skeleton, y debería: es de los fallos semánticos más frecuentes del presente.

La regla practicable: la forma prestada es deuda, y solo debe prestarse la que el contenido pueda pagar. Y la transición merece puntuación: el paso de silueta a contenido es la verdadera aparición — el *emerge* real—, y fundirlo en un reemplazo instantáneo regala el momento perceptivo más informativo de la espera. Queda abierto qué es el skeleton: ¿un verbo de *sustain*, una composición de *sustain* y *emerge*? ¿O el primer miembro de una clase que la gramática aún no tiene — los eventos que anticipan la forma de otros?

Caso 4 — El campo sin fondo: scroll infinito

La lista que nunca termina. El usuario desplaza — *handle.scroll*, el verbo existe—, y el sistema añade contenido antes de que el borde llegue. No hay petición percibida ni final posible.

La gramática nombra las piezas: un handle continuo, sustains breves de carga, un emerge por cada lote. Lo que el scroll infinito elimina es otra cosa: el commit de término. Nunca ocurre «esto ha acabado». La familia que registra consecuencias se queda sin trabajo en la interfaz donde el usuario más horas pasa. Y esa ausencia no es neutra: es el diseño el que decide que no haya cierre que percibir.

Hay más costuras. La carga se dispara por un gesto que el usuario no hizo como petición: desplazarse no es pedir más, y sin embargo trae más. Contenido que llega sin contacto que lo justifique; respuesta sin pregunta. Y el campo sin fondo roba la huella espacial: el «dónde estaba yo» con el que shift devuelve al usuario a su sitio no existe donde el territorio crece bajo los pies.

Lo decible hoy: puntuar los umbrales — cada lote nuevo merece su emerge, no una continuidad fingida—, y devolver el término cuando exista: «has llegado al final» es un commit, y merece firma, no silencio. Queda abierta la pregunta grande. ¿Es el campo sin fondo un antipatrón en sí — una interfaz diseñada para que nada se perciba como consecuencia—? ¿O un régimen legítimo, con reglas propias que este libro no ha escrito?

Caso 5 — El otro que edita: colaboración en vivo

Un documento compartido. Otro cursor se mueve, un párrafo cambia solo, una selección ajena ilumina un titular. La gramática de este libro reparte

todos sus eventos entre dos actores — el usuario y el sistema—, y aquí actúa un tercero que no es ninguno de los dos.

No es delegate, y conviene decirlo pronto. El colaborador no actúa *por cuenta de* nadie: actúa por cuenta propia sobre un espacio común. La pregunta de delegate — ¿quién actúa? — sirve; su estructura — la dirección de la delegación — no. Sus acciones llegan además con una rareza perceptiva: son emerge y commit sin contact previo del receptor. Eventos con autor pero sin gesto local: el párrafo cambia y nadie de este lado lo tocó.

Con las piezas existentes se llega lejos. La presencia ajena es un *emerge.present* persistente — está aquí—. Su actividad, un signal de baja intensidad — hace algo—. Sus consecuencias, commits con autoría atribuida — lo hizo ella—. La atribución de agencia del Capítulo 29 hace el trabajo duro: mientras cada cambio tenga autor perceptible, el espacio compartido se mantiene legible.

Lo que queda abierto es la pregunta de fondo: ¿merece la presencia de otro agente humano una familia propia? Los criterios del Capítulo 7 dan señales a favor — una pregunta perceptiva recurrente («¿quién más está aquí y qué hace?»), diferenciabilidad, recurrencia creciente—. Este libro la deja formulada y sin conceder: añadir una familia es caro, y la novena tendría que pagar el precio que las ocho pagaron.

Caso 6 — La notificación que emigra: eventos a nivel de sistema

La aplicación quiere avisar y el usuario no está en ella. El evento emigra: lo mostrará el sistema operativo, con su banner, su sonido y sus reglas. Es el único caso del libro en que la firma perceptiva la ejecuta otro.

La gramática viaja bien hasta la frontera. El evento sigue siendo lo que era — un `signal.notify`, un `commit.complete + fulfill`—. Lo que no cruza la frontera es la firma. El sistema aplanar los intents — el mismo banner para la celebración y para la amenaza—. Decide el sonido por su cuenta. Y deja la persistencia en manos de los ajustes del usuario. Es la migración semántica del Capítulo 19, pero forzada y hacia un canal que la aplicación no gobierna: la firma ya no se declara; se negocia.

Dos disciplinas sobreviven al aplanado. La primera: el texto es el único canal que cruza intacto — el intent debe poder leerse en las palabras, porque nada más lo llevará—. La segunda: la vuelta se re-puntúa. Cuando el usuario entra desde la notificación, la aplicación debe reconstruir la huella: qué ocurrió, dónde, con qué consecuencia. La ocurrencia se gastó en una superficie ajena; el estado es lo único que queda.

Abierto queda el problema entero: la coordinación *entre* gramáticas. El Capítulo 20 enseña a coordinar canales dentro de una superficie; nadie ha escrito cómo se coordinan dos superficies con dueños distintos. Mientras tanto, la regla honesta es de humildad: diseñar la notificación como si solo existiera su texto, y tratar todo lo demás como regalo revocable.

Caso 7 — Voz y XR, brevemente

Dos fronteras más lejanas merecen párrafo, no capítulo. La interfaz de voz no tiene superficie persistente: la doctrina de las dos superficies — ocurrencia y estado — pierde la mitad, porque todo es ocurrencia. La huella no tiene dónde vivir. Los commits piden eco explícito — «borrado» dicho, no supuesto—, y cada `sustain` carga sobre la memoria del usuario lo que una barra de progreso llevaría por él. Una gramática sin estado consultable es posible, pero es otra gramática: más ceremoniosa, más redundante, y con razón.

En XR ocurre lo contrario: los canales se vuelven literales. La presencia y la profundidad dejan de ser metáfora sobre un plano y pasan a ser el espacio mismo; un objeto que se acerca deprisa es un threat corporal antes que uno semántico. La carga paradigmática del Capítulo 10 opera aquí con materia prima real — el sobresalto no es convención: es reflejo—. La gramática gana potencia expresiva y pierde inocencia: en XR, equivocar un intent no confunde; asusta. Ambos casos quedan anotados como fronteras del medio, no como deudas del libro: cruzarlas pediría los capítulos de canal reescritos para otro cuerpo.

Lo que los siete casos enseñan juntos

Vistos en fila, los casos comparten un patrón que vale más que cada uno. La gramática no falla por falta de nombres: casi todo lo nombrable estaba nombrado. Falla — suda — en las costuras: el flujo que no cierra, la promesa con forma, el autor sin gesto, la superficie ajena, el campo sin fondo. Son problemas de junta, no de pieza.

Y eso es una buena noticia disfrazada. Las piezas aguantan; lo que falta es doctrina de composición para regímenes nuevos — exactamente el tipo de trabajo que una gramática provisional promete hacer—. Cada caso queda aquí con su parte resuelta, su costura señalada y su pregunta abierta. Cuando alguna se cierre, este apéndice habrá cumplido: era el mapa de la frontera, no la frontera.

PRINCIPIO

La gramática no falla por falta de nombres: suda en las costuras. Y una costura señalada es trabajo por hacer, no vergüenza que esconder.

Glosario

Contexto operativo: Conjunto de información del sistema, estados visibles, restricciones y operaciones disponibles que determinan qué puede hacer el usuario, qué significa cada acción y qué resultado puede producir.

Interfaz: Medio por el que una persona percibe, interpreta y actúa sobre un contexto operativo.

Evento: Cambio perceptible del contexto operativo que afecta a lo que el usuario puede hacer, debe atender o necesita interpretar.

Evento técnico: Ocurrencia registrada por el sistema, como click, submit, change o dragstart.

Unidad básica de cambio: Función del evento dentro de la gramática: no el fragmento más pequeño, sino la unidad desde la que se analiza qué cambió para el usuario.

Gramática de eventos: Sistema de diferencias, relaciones y reglas de composición que permite hacer reconocibles clases significativas de cambio.

Familia semántica: Clase recurrente de evento que responde a una pregunta estable del usuario.

Interacción semántica: Forma recurrente con la que una interfaz hace reconocible una familia de evento.

Firma de expresión: Configuración concreta de canales, intensidad, duración, persistencia y huella con la que se expresa un evento.

Intent: Valoración que declara un evento sobre cómo debe percibirse: neutral, affirm, fulfill, risk, threat o loss.

Canales de expresión: Vías por las que un evento se hace perceptible: tiempo, motion, presencia, profundidad, forma, color, sonido y háptica.

Contact: Familia que comunica que el sistema ha sentido una acción.

Commit: Familia que comunica que un resultado se ha aplicado, fallado, completado o perdido.

Signal: Familia que reclama atención.

Handle: Familia de manipulación directa.

Emerge: Familia de aparición, retirada o disponibilidad.

Shift: Familia de cambio de contexto.

Sustain: Familia de continuidad o proceso en curso.

Delegate: Familia donde cambia quién actúa, quién tiene la iniciativa o quién ejecuta parte de la acción.

Neutral: Intent de baja carga: algo debe notarse sin tono fuerte.

Affirm: Confirmación positiva suave.

Fulfill: Cierre positivo de una tensión u objetivo cumplido.

Risk: Problema corregible.

Threat: Amenaza activa que todavía puede evitarse.

Loss: Pérdida ya producida o disponibilidad perdida.

Huella: Modo en que un evento sigue importando después de su expresión visible.

Migración de significado: Traslado de una lectura a otros canales cuando un canal se reduce, falla o no está disponible.

Índice analítico

acción delegada: delegate; capítulos 8, 29, 36; apéndice C

accesibilidad: capítulos 3, 19, 33; apéndice B

canales de expresión: parte II; capítulos 11-20

commit: capítulos 8, 23

contact: capítulos 8, 22

contexto operativo: capítulos 1, 4

delegate: capítulos 8, 29, 36; apéndice C

evento: capítulos 1, 4, 5

firma de expresión: apéndice B; capítulo 36

handle: capítulos 8, 25

intent: capítulos 9, 10

loss: capítulo 10

migración de significado: capítulos 19, 33

risk: capítulo 10

shift: capítulos 8, 27

signal: capítulos 8, 24

sustain: capítulos 8, 28

threat: capítulo 10

unidad básica de cambio: capítulo 4

Referencias bibliográficas

Amershi, S., Weld, D., Vorvoreanu, M., et al. (2019). Guidelines for Human-AI Interaction. Proceedings of CHI 2019.

Arthur, W. B. (1994). Increasing Returns and Path Dependence in the Economy. University of Michigan Press.

Bilgic, M. & Mooney, R. J. (2005). Explaining Recommendations: Satisfaction vs. Promotion. Beyond Personalization Workshop, IUI 2005.

Chalmers, M. & MacColl, I. (2003). Seamful and Seamless Design in Ubiquitous Computing. Workshop «At the Crossroads», UbiComp 2003.

Cunningham, W. (1992). The WyCash Portfolio Management System. OOPSLA '92.

David, P. A. (1985). Clio and the Economics of QWERTY. American Economic Review, 75(2).

de Souza, C. S. (2005). The Semiotic Engineering of Human-Computer Interaction. MIT Press.

Ehsan, U., Liao, Q. V., Passi, S., Riedl, M. O. & Daumé, H. (2024). Seamful XAI: Operationalizing Seamful Design in Explainable AI. Proceedings of the ACM on Human-Computer Interaction (CSCW).

Eslami, M., Rickman, A., Vaccaro, K., et al. (2015). «I always assumed that I wasn't really that close to [her]»: Reasoning about Invisible Algorithms in News Feeds. Proceedings of CHI 2015.

Gibson, J. J. (1979). The Ecological Approach to Visual Perception. Houghton Mifflin.

Heider, F. & Simmel, M. (1944). An Experimental Study of Apparent Behavior. *The American Journal of Psychology*.

Horvitz, E. (1999). Principles of Mixed-Initiative User Interfaces. *Proceedings of CHI 1999*.

Hutchins, E., Hollan, J. & Norman, D. (1985). Direct Manipulation Interfaces. *Human-Computer Interaction*.

Kuhn, T. S. (1962). *The Structure of Scientific Revolutions*. University of Chicago Press.

LeDoux, J. (1996). *The Emotional Brain*. Simon & Schuster.

Liebowitz, S. J. & Margolis, S. E. (1990). The Fable of the Keys. *Journal of Law and Economics*, 33(1).

Maes, P. (1994). Agents that Reduce Work and Information Overload. *Communications of the ACM*, 37(7).

Michotte, A. (1946). *The Perception of Causality*.

Miller, R. B. (1968). Response time in man-computer conversational transactions. *AFIPS Fall Joint Computer Conference*.

Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann.

Norman, D. A. (2013). *The Design of Everyday Things*. Basic Books.

Parasuraman, R. & Manzey, D. H. (2010). Complacency and Bias in Human Use of Automation: An Attentional Integration. *Human Factors*.

Pariser, E. (2011). *The Filter Bubble: What the Internet Is Hiding from You*. Penguin Press.

Posner, M. I. (1980). Orienting of Attention. *Quarterly Journal of Experimental Psychology*.

Russell, J. A. (1980). A Circumplex Model of Affect. *Journal of Personality and Social Psychology*.

Scherer, K. R. (2005). What are emotions? And how can they be measured? Social Science Information.

Shneiderman, B. (1983). Direct Manipulation: A Step Beyond Programming Languages. IEEE Computer.

Spence, C. (2011). Crossmodal correspondences: A tutorial review. Attention, Perception, & Psychophysics.

Synofzik, M., Vosgerau, G. & Newen, A. (2008). Beyond the comparator model: A multifactorial two-step account of agency. Consciousness and Cognition.

Tintarev, N. & Masthoff, J. (2007). A Survey of Explanations in Recommender Systems. IEEE 23rd International Conference on Data Engineering Workshop (ICDE).

Wallace, M. T. & Stevenson, R. A. (2014). The construct of the multisensory temporal binding window and its dysregulation in developmental disabilities. Neuropsychologia.

W3C. Web Content Accessibility Guidelines (WCAG) 2.2.

W3C. WAI-ARIA Authoring Practices Guide.

W3C Design Tokens Community Group. Design Tokens Format Module.

Wagemans, J., Elder, J. H., Kubovy, M., et al. (2012). A century of Gestalt psychology in visual perception. Psychological Bulletin.

Weiser, M. (1991). The Computer for the 21st Century. Scientific American, 265(3).

Lecturas complementarias

Obras consultadas que informan el marco del libro sin ser citadas directamente en el cuerpo.

Brewster, S. & Brown, L. M. (2004). Tactons: Structured Tactile Messages for Non-Visual Information Display.

Kaptelinin, V. & Nardi, B. (2006). Acting with Technology: Activity Theory and Interaction Design. MIT Press.

Merleau-Ponty, M. (1945). Phenomenology of Perception.

Rasmussen, J. & Vicente, K. J. (1989). Coping with human errors through system design: implications for ecological interface design.

Suchman, L. (1987). Plans and Situated Actions. Cambridge University Press.

Vicente, K. J. (2002). Ecological Interface Design: Progress and Challenges. Human Factors.

Sobre el autor

Juan V. Navarro Leal escribió su primera línea de código hace cuarenta años — este año se cumplen cuatro décadas de su primer ordenador — y no ha dejado de escribirlas desde entonces. Su carrera, autodidacta y guiada por la curiosidad, ha recorrido buena parte de la historia práctica de la programación. Los lenguajes de inferencia, como Lisp y Prolog; los académicos, como Pascal, Modula-2 y Ada; y los que tocan la máquina, el ensamblador y C.

Ese recorrido desembocó en el front-end, donde nacieron las preguntas de este libro. «Diseñando lo que ocurre» es su destilación: la sospecha, convertida en gramática, de que a las interfaces les faltaba un lenguaje para lo que ocurre.