Quuipu

SERIE QUUIPU RESEARCH & ARCHITECTURE

WHITEPAPER 08

Context Model

Il context come infrastruttura — Un modello formale per esperienze digitali contestuali



Whitepaper fondativo · Italiano

Quuipu Edizione 2 · v1

Identità della pubblicazione

Pubblicazione
Context Model

Serie
Serie Quuipu Research & Architecture

Numero di pubblicazione
Whitepaper 08

Edizione
Quuipu Edizione 2 · v1

Lingua
Italiano

Edizione storica
Cognisphere Edizione 1

Questa pubblicazione succede a Cognisphere Edizione 1. I PDF storici restano immutabili.

Indice

01. Context as infrastructure

02. Formal vocabulary

03. The Context Vector

04. Declared context

05. Inferred context

06. Intent model

07. Goal model

08. Session model

09. State model

10. Temporal context

11. Context transitions

12. Context events

13. Context resolution

14. Context conflicts

15. Context expiration

16. Context minimization

17. Context continuity

18. Context portability

19. Context and memory

20. Context and semantics

21. Context and AI

22. Context privacy

23. Multi-tenant context

24. Context APIs

25. Context invariants

26. Reference implementation pattern

27. Failure modes

28. Metrics and evaluation

29. The Context Contract

30. The infrastructure principle

A. Context Item reference schema

B. Context transition matrix

C. Portable Context Package

D. Context Model invariants

Preamble

Il context come infrastruttura — Un modello formale per esperienze digitali contestuali

> Questa pubblicazione succede a Cognisphere Edizione 1. Quuipu Edizione 2 preserva la tesi storica aggiornando brand e identità di pubblicazione attivi.

Preamble

Quuipu Context Model Context as Infrastructure A Formal Model for Context-Aware Digital Experiences

Foundational Model

Context is not a prompt Il contesto diventa infrastruttura quando possiede schema, lifecycle, transizioni, policy, scadenza e portabilità indipendenti dalla singola interfaccia.

C(t) = <I, G, S, T, D, E, R, P> Intent · Goal · Session/State · Temporal · Declared · Inferred Evidence · Resources/Sources · Policy

Context Sources ↓

Typed Context Items ↓

Resolver ↓

Task Projection pi(C, task) ↓

Capability / Experience ↓

Transition + Event

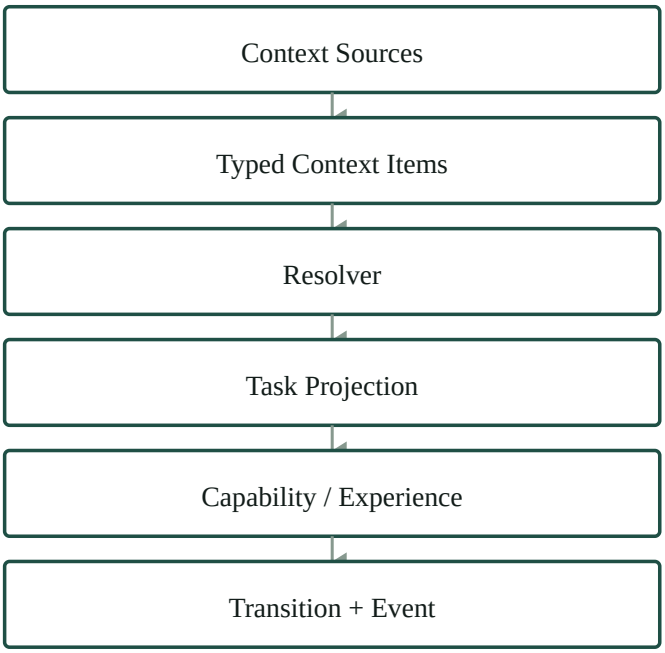


Figure 1 — Loop infrastrutturale del context.

Formal Core

State and transition system C(t+1) = tau(C(t), e, P) X(t+1) = delta(X(t), event, policy)

Object Formal role Key property

C(t) Available context at time t Partial, typed, governed

C* Resolved context Conflict-aware operational view

pi(C,task) Task projection Minimum necessary context

X(t) Runtime state Operational continuity

tau Context transition Add/update/expire/transfer

delta State transition Deterministic structural evolution

CAPITOLO 01

Context as infrastructure

Il contesto non è metadata decorativo né una cronologia da riversare in un modello. In Quuipu è una struttura operativa governata che determina quali informazioni, relazioni, trasformazioni e percorsi sono pertinenti in un dato momento.

Tesi Una digital experience diventa context-aware quando il sistema può rappresentare, aggiornare, limitare, scadere e trasferire il contesto senza confonderlo con identità, profilazione o memoria illimitata.

Infrastructure property Il Context Model deve essere indipendente dalla singola UI, dal provider AI e dal CMS: più experience possono consumare lo stesso context contract.

Boundedness Ogni elemento contestuale possiede origine, scope, confidence, lifetime e policy; ciò che non può essere giustificato non dovrebbe entrare nello state operativo.

CAPITOLO 02

Formal vocabulary

Un modello utile richiede termini non intercambiabili. Context, intent, goal, session, state e memory descrivono oggetti differenti e devono restare distinguibili nel runtime.

Context Insieme strutturato di fatti, dichiarazioni, segnali e riferimenti temporalmente validi rilevanti per una decisione o esperienza.

Intent Orientamento corrente dell'utente verso un'azione o una comprensione; può essere dichiarato, selezionato o inferito entro limiti espliciti.

Goal Condizione-obiettivo più stabile rispetto al singolo intent: ciò che la sessione o il percorso cerca di raggiungere.

Session Contenitore temporale e operativo che delimita una sequenza di interazioni correlate.

State Rappresentazione corrente del sistema necessaria a continuare coerentemente l'esperienza.

Memory Persistenza selettiva oltre il ciclo immediato; non coincide con context e richiede regole proprie di retention e revoca.

Term Lifetime tendency Authority

Intent Short / task-bound Declared or bounded inferred

Goal Medium / journey-bound User/org defined

Session Ephemeral Runtime boundary

State Immediate Runtime-derived

Context Mixed Multi-source, resolved

Memory Persistent/selective Governed persistence

CAPITOLO 03

The Context Vector

Per ragionare tecnicamente, Quuipu rappresenta il contesto come composizione di dimensioni tipizzate invece che come un unico blob testuale.

Definition Al tempo t: C(t) = , dove I=intent, G=goal, S=session/state, T=temporal context, D=declared context, E=inferred evidence, R=resource/source context, P=policy/privacy constraints.

Partiality Non tutte le dimensioni devono essere valorizzate. Un contesto valido può essere parziale purché dichiari ciò che manca.

Typed dimensions Ogni dimensione ha schema, source, timestamp, scope e validity; questo consente merge e expiration senza reinterpretare testo libero.

Projection Una capability non riceve C(t) intero, ma una proiezione minima pi(C(t), task) che contiene solo le dimensioni necessarie.

Dimension Symbol Example

Intent I understand eligibility

Goal G complete application

Session/State S current step 3

Temporal T deadline / freshness

Declared D selected language/mode

Inferred evidence E bounded session signal

Resources R authoritative source set

Policy P privacy/materiality constraints

CAPITOLO 04

Declared context

Il contesto dichiarato è quello espresso direttamente dalla persona o dall'organizzazione e costituisce la forma preferita di adattamento quando disponibile.

Examples 'Sto cercando di capire i requisiti per...', lingua scelta, modalità Simple, ruolo dichiarato per il task, vincolo temporale o destinazione selezionata.

Authority Un dato dichiarato non è automaticamente vero in senso assoluto; è però autorevole rispetto alla dichiarazione dell'utente e non dovrebbe essere sovrascritto silenziosamente da inferenze.

Revocability Il declared context deve poter essere modificato, azzerato o sostituito con effetto osservabile.

No identity requirement Un utente può dichiarare un bisogno senza rivelare una categoria sensibile o una diagnosi.

CAPITOLO 05

Inferred context

L'inferenza può colmare lacune operative, ma deve avere un rango inferiore al contesto dichiarato e non trasformarsi in profilazione invisibile.

Evidence-based inference L'inferenza dovrebbe derivare da segnali limitati e pertinenti alla sessione, con reason code e confidence.

Precedence Declared > explicitly selected > bounded inferred > unknown è una gerarchia utile per la risoluzione dei conflitti.

Materiality Più una decisione è materiale, meno è accettabile affidarsi a inferenze deboli.

Correction Quando l'utente corregge un'inferenza, lo state deve aggiornarsi e la precedente ipotesi non deve continuare a guidare l'esperienza.

Rank Context class Conflict behavior

1 Declared Preserve unless user changes it

2 Explicitly selected Use within selection scope

3 Bounded inferred May fill gaps; never silently override 1/2

4 Unknown Remain unknown or ask

CAPITOLO 06

Intent model

L'intent non è una query string. È un oggetto operativo che collega ciò che l'utente sta cercando di fare alle capability e alle fonti che possono soddisfarlo.

Intent object Intent = {id, type, subject, constraints, declaredBy, confidence, createdAt, expiresAt}.

Intent hierarchy Intenti generali possono essere raffinati in sub-intent senza perdere la relazione con l'obiettivo.

Competing intents Una sessione può contenere intenti concorrenti; il runtime deve identificare activeIntent invece di fonderli indiscriminatamente.

Intent completion Un intent può essere fulfilled, abandoned, superseded o expired, producendo una transition esplicita.

CAPITOLO 07

Goal model

Il goal fornisce continuità sopra una sequenza di intenti e impedisce che ogni interazione venga interpretata come un nuovo problema isolato.

Goal object Goal = {id, desiredState, successCriteria, scope, priority, status}.

Goal vs intent 'Capire se posso presentare domanda' può essere goal; 'verificare i documenti richiesti' è un intent subordinato.

Success criteria Un goal utile possiede criteri osservabili, non soltanto una descrizione linguistica.

Goal portability Se autorizzato, un goal può sopravvivere a una sessione e riaprire il contesto in un'altra experience.

CAPITOLO 08

Session model

La sessione delimita il contesto effimero e offre un confine naturale per minimizzazione, cleanup e isolamento.

Session identity Session ID non deve equivalere a user identity; può essere anonimo e scoped al tenant o all'experience.

Lifecycle created → active → suspended → resumed → closed/expired.

Session state Contiene active intent, current node, selected mode, bounded history references e capability status.

Isolation Sessioni concorrenti non condividono state implicitamente; ogni trasferimento deve essere esplicito.

CAPITOLO 09

State model

Lo state è ciò che il runtime deve conoscere adesso per continuare in modo coerente. È più stretto del contesto totale e più operativo della memoria.

State tuple X(t) = .

State transition X(t+1) = delta(X(t), event, policy), dove delta è una funzione governata e osservabile.

Determinism A parità di state, event e policy, le transizioni strutturali dovrebbero essere deterministiche anche quando una capability AI produce contenuto probabilistico.

Snapshot Snapshot versionati consentono rollback, debugging e decision trace senza conservare transcript illimitati.

Event ↓

Validate ↓

delta(X,e,P) ↓

New Snapshot ↓

Emit Context Event

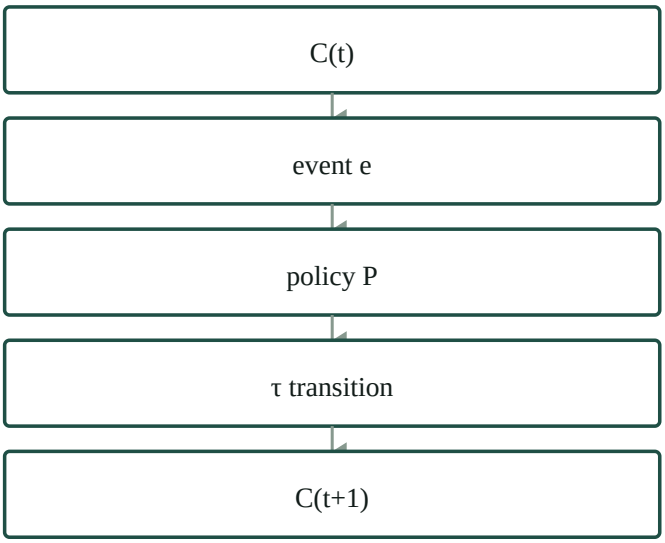


Figure 2 — Percorso di transizione di stato.

CAPITOLO 10

Temporal context

Il significato di un fatto contestuale dipende spesso dal tempo: quando è stato osservato, per quanto è valido e rispetto a quale evento.

Time fields observedAt, effectiveFrom, effectiveUntil, ttl e lastValidatedAt sono distinti.

Event time vs processing time Il momento in cui un evento è avvenuto può differire dal momento in cui il runtime lo riceve.

Freshness La rilevanza temporale può essere una funzione f(age, domain, sourceAuthority), non un TTL universale.

Temporal contradictions Due valori possono essere entrambi corretti in intervalli differenti; il resolver deve preservare valid-time invece di dichiarare conflitto automaticamente.

Temporal field Meaning

observedAt When the value was observed

effectiveFrom / Until Business-valid interval

ttl Operational reuse window

lastValidatedAt Last authority/freshness check

CAPITOLO 11

Context transitions

Il Context Model è dinamico. Ogni nuova dichiarazione, selezione, evento o scadenza può produrre una transizione controllata.

Transition function C(t+1) = tau(C(t), e, P), con e=event e P=policy.

Transition types add, update, confirm, invalidate, expire, supersede, merge, fork e transfer.

Preconditions Una transizione può richiedere source authority, user confirmation o capability readiness.

Postconditions Il runtime deve poter verificare invarianti dopo ogni transition, ad esempio activeIntent unico e nessun item scaduto nella projection.

Transition Meaning Typical trigger

add New context item Declaration/event

confirm Raise explicitness User confirmation

invalidate No longer usable Conflict/policy

expire Lifetime ended Clock/TTL

supersede New value replaces old Updated declaration

fork Create alternative branch Competing goal

transfer Move projection Device/experience handoff

CAPITOLO 12

Context events

Gli eventi rendono il context lifecycle osservabile e disaccoppiano producer e consumer.

Core events context.declared, intent.activated, context.updated, context.expired, goal.completed, session.suspended, context.transferred.

Envelope Event = {eventId, type, tenantId, sessionId, timestamp, actor, payloadRef, reasonCode, schemaVersion}.

Idempotency Eventi con eventId già applicato non devono duplicare transizioni.

Ordering Sequence number o causal reference possono essere necessari quando più producer aggiornano lo stesso contesto.

CAPITOLO 13

Context resolution

Resolution trasforma un insieme potenzialmente incompleto o contraddittorio in una vista operativa adatta al task.

Resolver R(C, task, P) → C*, dove C* è il resolved context consumabile dalla capability.

Priority La risoluzione combina precedence, freshness, authority, explicitness e task relevance.

Unknown as a value L'assenza di informazione deve restare unknown, non essere riempita automaticamente con un default implicito.

Explainability Il resolver può restituire reason codes: DECLARED_OVERRIDE, EXPIRED_SIGNAL, SOURCE_CONFLICT, POLICY_REDACTED, TASK_IRRELEVANT.

score(v) = w1·explicitness + w2·authority + w3·freshness + w4·taskRelevance - w5·policyRisk

CAPITOLO 14

Context conflicts

Un sistema context-aware deve saper rappresentare conflitti senza forzarli prematuramente in un singolo valore.

Conflict classes declared-vs-inferred, source-vs-source, temporal, policy, multi-goal e cross-device.

Resolution strategy Prefer, qualify, ask, branch o block sono strategie diverse; non tutto deve essere auto-risolto.

Materiality threshold Per task a basso rischio può essere sufficiente una preferenza; per task regolati può essere richiesta conferma.

Trace Il conflitto e la sua risoluzione devono restare ricostruibili.

CAPITOLO 15

Context expiration

Un contesto che non scade diventa progressivamente un profilo. Expiration è quindi una proprietà architetturale, non un'operazione di housekeeping.

TTL by class Navigation state può vivere minuti; intent ore; explicit goal giorni o mesi se autorizzato; inferred signals spesso meno.

Hard vs soft expiration Hard expiry rimuove l'item dall'uso; soft expiry richiede revalidation prima di riutilizzarlo.

Expiry propagation Quando un item base scade, le derived assumptions dipendenti devono essere invalidate o ricalcolate.

Deletion Expiration operativa e cancellazione persistente sono concetti distinti e devono avere policy esplicite.

Class Typical lifetime Expiry action

Navigation state Minutes Hard expire

Active intent Hours/session Expire or supersede

Explicit goal Journey-defined Soft expire / revalidate

Inferred signal Short Hard expire

Source fact Domain-defined Revalidate by freshness

Portable package Short Reject after expiresAt

CAPITOLO 16

Context minimization

La qualità del context model dipende anche da ciò che decide di non conservare o non propagare.

Task projection pi(C, task) produce una vista minima per capability.

Purpose limitation Un elemento raccolto per una funzione non viene automaticamente riutilizzato per un'altra.

No transcript default Il transcript completo non è il formato canonico del context state.

Data classes Policy può escludere categorie da persistence, AI gateway, cross-device transfer o analytics.

CAPITOLO 17

Context continuity

Continuity preserva il filo del percorso senza richiedere che tutto il passato resti attivo.

Continuity token Un riferimento può puntare a goal, last valid state e source set senza contenere l'intera cronologia.

Resume Il runtime ricostruisce un nuovo C(t) verificando freshness e policy al momento della ripresa.

Selective memory Solo elementi esplicitamente promossi a memory sopravvivono oltre i confini definiti.

Discontinuity Quando il contesto non è più affidabile, il sistema deve poter ripartire dal canonical experience.

CAPITOLO 18

Context portability

Portabilità significa trasferire un contesto comprensibile e governato fra experience, device o host senza esportare implicitamente identità o dati non necessari.

Portable package CP = {schemaVersion, tenantScope, goalRef, intentRef, stateProjection, sourceRefs, issuedAt, expiresAt, consent/policy}.

Same-tenant transfer Può preservare più state quando source authority e policy sono condivise.

Cross-tenant transfer Richiede una projection molto più restrittiva e mapping esplicito; per default nessun dato privato attraversa il boundary.

Verification Il receiver valida firma/integrità, schema, expiry, audience e policy prima dell'import.

Source Context ↓

Projection ↓

Portable Package CP ↓

Integrity + Audience + Expiry Check ↓

Candidate Import ↓

Local Resolution

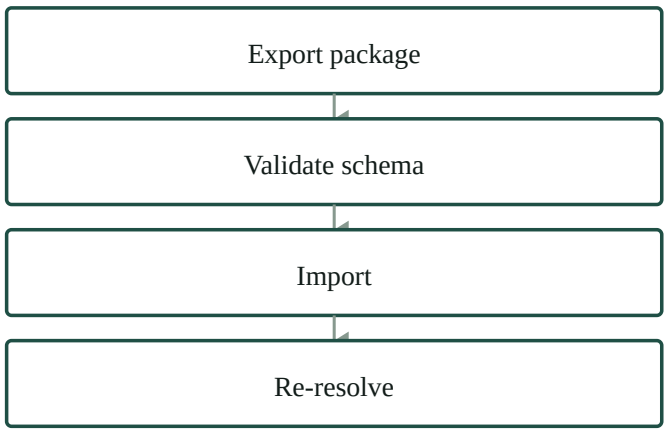


Figure 3 — Portabilità del context.

CAPITOLO 19

Context and memory

Context e memory interagiscono ma non sono sinonimi. Memory è una fonte possibile per ricostruire contesto, soggetta a selezione e verifica.

Promotion Un context item può diventare memory solo attraverso una regola di promotion o una scelta esplicita.

Recall Memory retrieval produce candidate context; non aggiorna automaticamente active state.

Staleness Un ricordo persistente può essere obsoleto e deve essere rivalidato.

Revocation La revoca di memory deve propagarsi alle future context reconstructions.

CAPITOLO 20

Context and semantics

Il contesto seleziona quale significato o rappresentazione è pertinente; non autorizza a cambiare il significato canonico.

Semantic identity Concept identity rimane stabile mentre context determina depth, ordering, examples e representation.

Disambiguation Il context può scegliere fra sensi validi di un termine usando dominio e intent.

Transformation boundary Context può influenzare come spiegare, non inventare nuovi fatti.

WP04 link Semantic Integrity vincola ogni context-conditioned transformation.

CAPITOLO 21

Context and AI

Un LLM è un consumer di una projection del contesto, non il proprietario del contesto.

Packaging L'AI Gateway riceve pi(C, task), evidence e policy envelope.

Provider isolation Provider memory o conversation state non diventano automaticamente Quuipu state.

Prompt minimization Typed context evita di reinviare transcript e riduce token, leakage e ambiguity.

Model output Una risposta AI può generare candidate events o candidate context, ma la transition viene validata dal runtime.

C(t) ↓

Task Projection ↓

AI Gateway ↓

LLM ↓

Candidate Output/Event ↓

Runtime Validation ↓

tau / delta

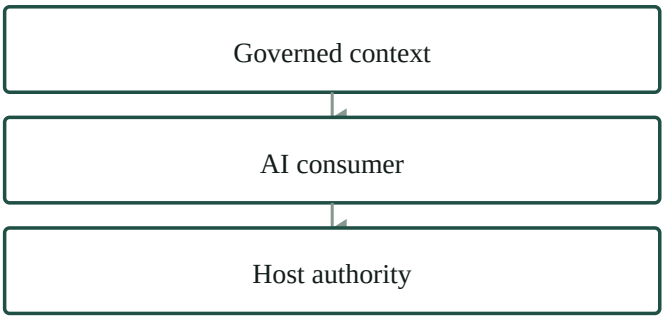


Figure 4 — L'AI consuma il context; non lo possiede.

CAPITOLO 22

Context privacy

Privacy-by-architecture richiede che ogni context item dichiari perché esiste, dove può fluire e quando deve cessare di essere usato.

Metadata origin, purpose, sensitivity, retentionClass, allowedConsumers e transferPolicy accompagnano l'item.

Sensitive inference Il runtime non dovrebbe inferire categorie sensibili solo per migliorare la personalizzazione.

Least context La migliore esperienza non coincide con il massimo contesto disponibile.

Audit Decision trace registra l'uso di context rilevante senza richiedere una copia integrale dei dati.

CAPITOLO 23

Multi-tenant context

In deployment enterprise, il tenant boundary è parte del modello formale e precede ogni resolution.

Namespace Context key e source refs sono namespaced per tenant.

No ambient merge Due tenant non condividono context, memory, embeddings o approval state implicitamente.

Shared infrastructure Servizi runtime possono essere condivisi fisicamente mantenendo logical isolation e policy separate.

Transfer gateway Ogni cross-tenant exchange passa da un gateway esplicito con mapping e minimization.

CAPITOLO 24

Context APIs

Le API devono esporre operazioni semantiche invece di consentire mutazioni arbitrarie di un blob di sessione.

Commands declareContext, activateIntent, setGoal, transitionState, expireContext, exportProjection, importContextPackage.

Queries getResolvedContext, getActiveIntent, getGoalStatus, getContextTrace, getPortableProjection.

Optimistic concurrency Version/etag impedisce lost update in presenza di più producer.

Schema evolution Ogni envelope dichiara schemaVersion e migration rules.

API class Examples Constraint

Command declareContext, activateIntent, setGoal Validated semantic mutation

Transition transitionState, expireContext Policy + version checked

Query getResolvedContext, getActiveIntent Read projection/state

Portability exportProjection, importContextPackage Audience + expiry + tenant policy

Trace getContextTrace Reason codes, not hidden reasoning

CAPITOLO 25

Context invariants

Gli invarianti rendono il modello verificabile e impediscono che l'evoluzione del prodotto trasformi context in un contenitore indefinito.

C-01 Declared context non viene sovrascritto silenziosamente da inferred context.

C-02 Ogni item ha scope e lifetime.

C-03 Expired context non entra nelle task projections senza revalidation.

C-04 Una capability riceve il minimo context necessario.

C-05 Cross-tenant portability è deny-by-default.

C-06 AI output non muta state senza una transition validata.

CAPITOLO 26

Reference implementation pattern

Un'implementazione concreta può combinare event log minimizzato, state store, resolver, expiration scheduler e portability gateway.

Write path Command → validation → transition → state snapshot → event.

Read path Task → resolver → policy filter → projection → capability.

Expiry path Clock/event → expiration rule → invalidation → derived-state recomputation.

Transfer path Projection → package → integrity/policy check → import as candidate context.

Observability Metrics descrivono transition rate, stale-context rejection, conflict rate, projection size e resume success.

Command ↓

Schema + Policy Validation ↓

Transition Engine ↓

State Store ↓

Event Log ↓

Resolver / Projection

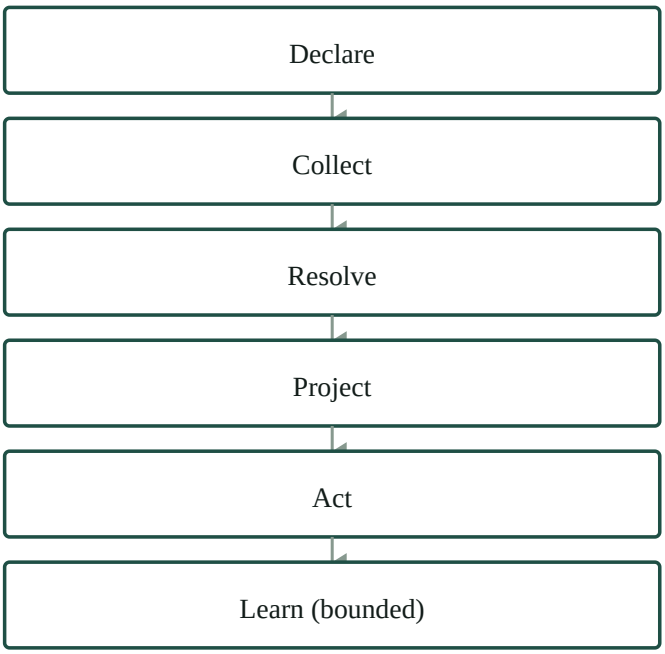


Figure 5 — Pattern di implementazione di riferimento.

CAPITOLO 27

Failure modes

Un Context Model robusto deve progettare esplicitamente il comportamento quando il contesto è incompleto, vecchio, incoerente o non trasferibile.

Stale context Revalidate, ask or fall back; non continuare silenziosamente.

Ambiguous intent Presentare scelte bounded invece di inferire un obiettivo ad alta materialità.

State divergence Version conflict produce retry/merge controllato, non last-write-wins cieco.

Portability failure Il receiver può rifiutare il package e ricostruire dal canonical experience.

Policy conflict La policy più restrittiva prevale fino a risoluzione esplicita.

CAPITOLO 28

Metrics and evaluation

Il Context Model può essere valutato con metriche architetturali e di esperienza, non soltanto con accuratezza di classificazione.

Resolution quality Percentuale di task risolti senza context correction; conflict resolution correctness.

Freshness Stale-context use rate e revalidation success.

Minimization Dimensione media delle projections rispetto al context disponibile.

Continuity Resume success e context-reset rate.

Trust Declared-over-inferred precedence violations, cross-tenant leakage = zero, trace completeness.

Metric Target direction Why

Context correction rate Down Resolution quality

Stale-context use Toward zero Temporal integrity

Projection/context ratio Down, task-dependent Minimization

Resume success Up Continuity

Cross-tenant leakage Zero Isolation

Precedence violations Zero Human agency

CAPITOLO 29

The Context Contract

Il contratto finale separa ciò che il sistema sa, ciò che suppone, ciò che l'utente ha dichiarato e ciò che è autorizzato a usare.

Know Fatti e source state verificati entro uno scope.

Assume Inferenze bounded con confidence e expiry.

Declare Intenti, vincoli e preferenze espresse dalla persona.

Use Solo la projection autorizzata per il task corrente.

Forget Ciò che scade, viene revocato o non ha più purpose.

CAPITOLO 30

The infrastructure principle

Context as Infrastructure significa che la continuità dell'esperienza non dipende da una chat, da un cookie opaco o dalla memoria proprietaria di un modello.

Portable Il contesto può essere serializzato come contratto tipizzato e minimizzato.

Replaceable UI, host e provider AI possono cambiare senza perdere il modello concettuale.

Governable Policy, expiry, provenance e tenant boundaries sono parte dello stesso sistema.

Human-centered L'utente può dichiarare, correggere, revocare e trasferire ciò che è rilevante.

Final thesis The Web becomes context-aware when context itself becomes explicit, bounded, portable infrastructure.

CAPITOLO A

Context Item reference schema

Field Type Purpose

contextId UUID/string Stable item identity

kind enum intent/goal/fact/preference/state/ref

value typed payload Context content

origin declared/inferred/source/runtime Provenance

scope tenant/session/task/experience Where it may apply

confidence 0..1 / qualified Inference only where meaningful

observedAt timestamp Observation time

effectiveFrom/Until timestamp? Valid-time

expiresAt / ttl timestamp/duration Operational lifetime

purpose enum/string Purpose limitation

sensitivity class Privacy routing

allowedConsumers set Projection control

version integer Concurrency/evolution

CAPITOLO B

Context transition matrix

Current Event Guard Next / effect

unknown declare schema valid declared

inferred declare same dimension declared supersedes inferred

declared infer none declared remains active

active ttl elapsed hard expiry expired

active ttl elapsed soft expiry needs-validation

active conflict materiality high ask/block

session active suspend snapshot valid suspended

portable candidate import audience/policy valid locally resolved

CAPITOLO C

Portable Context Package

CP = {schemaVersion, packageId, issuer, audience, tenantScope, issuedAt, expiresAt, goalRef, intentRef, stateProjection, sourceRefs, policyRef, integrity}

Validation gate Failure behavior

Schema unsupported Reject or migrate explicitly

Expired Reject; rebuild context

Audience mismatch Reject

Tenant policy denies Reject/minimize

Integrity invalid Reject

Source refs unavailable Import partial + mark unresolved

Sensitive class not transferable Redact before import

CAPITOLO D

Context Model invariants

CTX-01 · Declared outranks inferred Inference cannot silently overwrite a declaration.

CTX-02 · Unknown remains representable Missing context is not silently converted into certainty.

CTX-03 · Every item expires or is explicitly persistent Lifetime is part of the schema.

CTX-04 · Purpose bounds use Context is projected for a task, not ambiently reused.

CTX-05 · State transitions are governed Mutation occurs through validated tau/delta functions.

CTX-06 · AI is a consumer Provider state is not canonical context.

CTX-07 · Memory requires promotion Persistence is selective, not automatic.

CTX-08 · Portability is minimized Transfer moves a projection, not the entire user history.

CTX-09 · Tenant boundary precedes resolution No cross-tenant ambient context.

CTX-10 · Temporal validity is first-class Freshness and valid-time affect resolution.

CTX-11 · Conflict can remain explicit The system may ask, branch or block instead of guessing.

CTX-12 · Canonical fallback survives Loss of context never removes access to the base experience.

APPENDIX E

Context threat / failure matrix Failure Architectural risk Control

Stale intent Wrong journey TTL + revalidation

Inferred override Loss of agency Precedence invariant

Context accumulation Profiling / cost Purpose + expiration + projection

Cross-session bleed Incorrect continuity Session isolation

Cross-tenant bleed Data exposure Namespace + deny-by-default transfer

Provider memory capture Vendor coupling/privacy Gateway projection only

Out-of-order events State divergence Sequence/causal refs

Blind last-write-wins Lost declaration Optimistic concurrency

Portable package replay Old context reuse packageId + expiry + audience

APPENDIX F

Relationship to the Quuipu series Whitepaper Context Model relationship

01 · The Cognitive Web Context is a foundational property of adaptive experience.

02 · Architecture Host/Core boundaries consume context through contracts.

03 · Quuipu Web Declared context drives website adaptation and continuity.

04 · Semantic Integrity Context selects representation without changing authoritative meaning.

05 · Trust Architecture Context use is governed by provenance, policy, trace and human control.

06 · Quuipu AI LLMs consume minimized context projections.

07 · Applications Domain use cases configure context dimensions and materiality.

08 · Context Model Formalizes context as typed, temporal, bounded and portable infrastructure.

Quuipu Context Model Context as Infrastructure A Formal Model for Context-Aware Digital Experiences

EXPLICIT. BOUNDED. TEMPORAL. PORTABLE.