Intentions as Communication Context Boundary

A Research Note for Perceptive App Development

Pronab Pal

IntentixLab, Melbourne, Australia

Draft version: 0.1 **Date:** 19 June 2026 **DOI:** 10.5281/zenodo.20761558
(https://doi.org/10.5281/zenodo.20761558)

Abstract

This note proposes that **Intention** should be treated as a first-class communication anchor in Perceptive App development. It builds on the Human Intention Space framework previously introduced as a natural-language phrase driven approach for placing social computing interaction in a designed space (Pal, 2024). Human communication rarely depends on payload alone. It depends on a bounded situation in which speaker, listener, object, action, expectation, and uncertainty become mutually interpretable. This bounded situation is referred to here as **Situational Reality**. In ordinary human communication, intention acts as an anchor that constrains what a message is about, why it is being communicated, and how subsequent perception should be interpreted. Traditional software systems usually lack an explicit, standard, runtime construct for such an anchor. They pass payloads, events, routes, requests, topics, or commands, but the contextual boundary of communication often remains hidden in code, documentation, or developer convention.

Intention Space addresses this gap by making **Signals** the backbone of runtime communication. A Signal carries context at two levels: first through an **Intention**, which establishes the semantic boundary of communication, and second through **Pulses**, whose phrases and trivalent values express fine-grained perceptual state. The Response area of a Pulse then carries ordinary payload data. This structure separates communicative context from payload while keeping them transportable together. The result is a computational model that more closely resembles human communication: meaning is not carried by raw data alone, but by data situated inside an intentional frame.

Keywords

Intention Space; Perceptive Apps; Situational Reality; communication context boundary; Signals; Pulses; Trivalence; LLM routing; developer-declared intention boundaries; semantic runtime architecture

1. Central Claim

The central claim of this note is:

> *Intentions are communication anchors of the human mind; in Perceptive Apps they should become explicit context boundaries for runtime communication.*

In real life, a message is rarely interpreted as an isolated payload. The same sentence, gesture, form value, or device signal may mean different things under different intentions. When one person says, "the door is open," the relevant meaning depends on whether the intention is to warn, invite, report a status, request action, diagnose a fault, or close a security incident.

Traditional software systems typically represent the payload but do not always represent this communicative boundary as a first-class runtime object. The boundary may be implicit in:

- a function name

- an API route
- a message topic
- a user-interface screen
- a workflow state
- a developer's mental model

These constructs are useful, but they usually do not standardize intention as the semantic anchor of communication. The application context therefore remains partly hidden. Intention Space makes this boundary explicit.

2. Situational Reality

This approach uses the term **Situational Reality** to mean reality as it becomes available for action through bounded perception and interpretation.

```
Situational Reality = Intention + Perceptual State
```

The world may be physically shared, but the meaning of an event is not automatically shared. Two people can observe the same event and act under different interpretations. A software system can receive the same payload in two different execution contexts and produce very different consequences.

For example, consider the simple condition:

```
door open = Y
```

This condition is not yet a full communicative situation. It becomes actionable only when placed under an intention:

```
Intention: report home comfort state
Pulse: door open = Y
Response: front door
```

or:

```
Intention: evaluate security breach
Pulse: door open = Y
Response: laboratory side entrance
```

The payload may be similar, but the situational reality is different. One may lead to ventilation, the other to escalation. The intention is the context boundary that prevents the payload from floating without meaning.

3. Relation to Human Communication

The proposal extends the earlier Intention Space claim that context and intention should be closely knitted into the computation process (Pal, 2024), and it has several points of contact with existing work in philosophy of language, social cognition, and communication theory.

In Gricean accounts of communication, meaning is strongly connected to speaker intention and the hearer's recognition of that intention (Grice, 1957). Speech-act theory likewise treats utterances not merely as sentences but as actions performed in social contexts (Austin, 1962; Searle, 1969). A request, warning, promise, report, and command may carry similar propositional content while differing in communicative force.

Work on common ground and grounding in communication also shows that successful communication depends on shared assumptions, mutual understanding, and repair when alignment is missing (Clark & Brennan, 1991). Shared agency and shared intentionality research further emphasizes that human cooperation depends on intentions that can coordinate participants, roles, and subplans (Bratman, 1992; Schweikard &

Schmid, 2017; Tomasello et al., 2005).

Intention Space does not simply import these theories into software. Rather, it takes a practical architectural lesson from them:

> *Communication is not only the movement of data; it is the movement of data under an interpretable intention.*

This is why Perceptive Apps should not treat payload as the primary unit of runtime meaning. Payload is necessary, but it is not sufficient. The payload must travel with a declared communicative boundary.

4. Signals as Context-Carrying Units

In Intention Space, a Signal is defined as:

```
Signal := <Intention, PulseSet>
```

The Intention establishes the semantic communication boundary. The PulseSet expresses the perceptual state that is relevant within that boundary.

A Pulse may be represented as:

```
Pulse := <phrase, TV, Response>
```

where:

- `phrase` is the semantic identity of the perception
- `TV` is trivalent status: `Y`, `N`, or `UN`
- `Response` carries ordinary payload data or response values

This gives context two levels:

1. **Intention-level context:** What kind of communication is this?
2. **Pulse-level context:** What perceptions are present, absent, or unknown inside that communication?

The Response field then carries traditional payload values, but the payload is no longer asked to carry hidden application meaning by itself.

5. Why Traditional Payloads Are Not Enough

Traditional software often compresses meaning into payload shape. A JSON body, database row, event message, or API request may contain values, but the interpretation of those values is distributed across many places:

- route handlers
- service methods
- validation code
- UI screen assumptions
- workflow state machines
- documentation
- developer memory

This creates a common problem: the data is visible, but the communication context is hidden.

For example:

```
{
  "door": "front",
  "open": true
}
```

This payload does not tell us whether the system is:

- reporting a home comfort condition
- warning about a security breach
- checking accessibility for a visitor
- validating a delivery instruction
- diagnosing a sensor fault

Software can infer these meanings from route or code context, but that context is rarely carried as a stable semantic object. In Intention Space, the same situation becomes:

```
Signal:
  Intention: evaluate security breach
  Pulses:
    <door open, Y, front door>
    <authorized access recorded, N, last badge read: none>
    <resident present, UN, no recent presence signal>
```

This is more granular than a payload because it tells the receiving Design Node not only what data exists, but under what communication boundary the data should be interpreted.

6. Granularity Through Trivalence

The trivalence of Pulses is central to the approach. Many systems collapse context into binary state or nullable fields. Intention Space instead distinguishes:

- `Y`: the perception is positively present
- `N`: the perception is negatively established
- `UN`: the perception is unknown, unresolved, or still to be determined

This gives Perceptive Apps a practical way to represent incomplete reality without pretending that unknown values are false.

Consider a medical appointment reminder:

```
Intention: prepare patient visit
Pulses:
  <appointment confirmed, Y, 14:30>
  <patient has arrived, N, reception check-in absent>
  <insurance verified, UN, pending verification>
  <urgent symptom reported, UN, no triage answer yet>
```

A traditional payload might say:

```
{
  "appointmentTime": "14:30",
  "arrived": false,
  "insurance": null,
  "urgentSymptom": null
}
```

The payload representation is useful, but it does not explicitly distinguish communication purpose from perceptual status. Intention Space does. The Design Node receives a bounded situation, not merely a record.

7. Relation to Current Technical Work

7. Developer Actions as Natural Intention Boundaries

Normal software development already creates intention boundaries. A developer names a route, writes a function, defines a form, creates a workflow step, declares a message topic, designs a screen, or exposes an API operation. Each of these acts is not merely technical. It is an act of bounded meaning.

For example:

```
POST /appointments/cancel
```

does not merely identify an HTTP path. It expresses a communicative boundary: some participant is trying to cancel an appointment. Likewise:

```
submitInsuranceVerification()
```

already tells us that the relevant communication is not generic data submission, but insurance verification.

Traditional software uses these boundaries operationally, but does not usually preserve them as explicit semantic runtime objects. They remain scattered across names, routes, code structure, and documentation. This is precisely where Intention Space gains leverage.

By recognizing these normal developer actions as **natural intention boundaries**, we can keep an LLM on a bounded trail:

```
developer-declared boundary
  -> allowed Intention
  -> allowed input Signal
  -> required Pulses
  -> deterministic validator
  -> DN execution
```

The LLM is not asked to invent application meaning. It is asked to identify which declared communicative boundary the user's expression belongs to, fill the required Pulses, or report that the situation is outside the declared boundary.

This reframes the role of the LLM:

> *The LLM is not the application actor; it is an Intention Receptor operating inside boundaries already created by developer action.*

This is a major practical advantage. It turns the developer's ordinary design work into a semantic containment structure for AI. Routes, functions, screens, forms, and API operations become more than implementation artifacts. They become declared tracks through which human language can be safely matched into executable Intention Space.

8. Relation to Current Technical Work

Several current technical movements point toward the same general concern: software systems need better ways to carry context.

The OpenAPI Specification standardizes machine-readable descriptions of HTTP APIs so that humans and computers can understand service capabilities without inspecting source code (OpenAPI Initiative, 2025). Model Context Protocol standardizes how AI applications connect to external systems, tools, and data sources (Model Context Protocol, 2026). Intent-based networking uses intents to express desired outcomes or policies

rather than low-level device commands (Mehmood et al., 2023). Recent context engineering work for large language models treats inference-time context as a formal design object rather than an incidental prompt (Mei et al., 2025).

These developments support the broader observation that context is becoming a primary engineering concern. However, Intention Space differs in its emphasis. It does not treat context only as retrieval material, API metadata, or external tool access. It treats **Intention** as the runtime boundary of communication and **Pulses** as trivalent perceptual carriers inside that boundary.

The practical distinction is:

```
Traditional integration: payload + route/tool/API metadata
Context engineering: prompt/context resources + model runtime
Intent-based systems: desired outcome or policy objective
Intention Space: Signal = Intention boundary + trivalent Pulse context + Response payload
```

This makes Intention Space particularly suited to Perceptive Apps, where the application must continuously coordinate human perception, device signals, runtime state, and executable Design Nodes.

9. A Perceptive App Example

Imagine a care-home application that coordinates morning medication.

In a conventional application, a medication action may be represented as:

```
{
  "residentId": "R214",
  "medicationId": "M88",
  "taken": false,
  "time": "08:00"
}
```

This payload contains data, but it does not itself express the communication situation. The same values could belong to a reminder, a compliance report, a risk escalation, a stock update, or a family notification.

In Intention Space, the situation can be carried as a Signal:

```
Signal:
  Intention: establish morning medication readiness
  Pulses:
    <resident identified, Y, R214>
    <medication scheduled, Y, M88 at 08:00>
    <medication physically available, Y, drawer A3>
    <resident consent present, UN, no response yet>
    <dose administered, N, not yet administered>
```

The receiving Design Node does not merely receive fields. It receives an intentional context boundary and a perceptual map of the situation. It can decide that the next appropriate action is not "mark failed," but "ask for resident consent" because the unknown pulse is explicitly represented.

Later, another Signal may carry the same resident and medication payload under a different intention:

```
Signal:
  Intention: escalate missed medication risk
  Pulses:
    <resident identified, Y, R214>
    <medication scheduled, Y, M88 at 08:00>
    <dose administered, N, not administered by 08:45>
    <resident consent present, UN, no recorded response>
    <nurse acknowledged delay, N, no acknowledgement>
```

The payload overlaps, but the communication boundary has changed. This is exactly how human communication works: the same facts can participate in different actions when the intention changes.

10. Design Implications

The approach leads to several design principles for Perceptive Apps:

1. **Make intention explicit.** Every meaningful runtime communication should carry an Intention, not merely a payload.
2. **Separate context from payload.** Response data should not be forced to carry hidden application meaning.
3. **Represent uncertainty directly.** Unknown state should be expressed as `UN`, not smuggled through nulls, missing fields, or false defaults.
4. **Use Signals as semantic boundaries.** Design Nodes should absorb and emit Signals, preserving communication context across execution.
5. **Treat payload as situated.** Payload values become meaningful only inside an Intention and Pulse configuration.
6. **Recognize developer-declared boundaries.** Routes, functions, forms, screens, workflow states, and API operations should be treated as natural intention boundaries.
7. **Prefer bounded interpretation.** An LLM or router should select among declared input Signals and fill required Pulses, not invent unbounded application meaning.

These principles make the application more inspectable because the context that drives execution becomes visible and transportable.

11. Contribution

The contribution of this note is not the claim that intention matters in human communication; that is already supported by a broad intellectual tradition. The contribution is the architectural proposal that Perceptive Apps should make intention an explicit runtime construct.

In this proposal:

- **Intention** is the communication context boundary.
- **Signal** is the transportable unit of intentional communication.
- **Pulse** is the atom of perceptual context.
- **Trivalence** preserves presence, absence, and unresolved state.
- **Response** carries ordinary payload inside an explicit context.
- **Developer-declared boundaries** provide natural tracks for keeping LLM interpretation inside declared application meaning.

This shifts application development from payload-first design to intention-bounded communication design.

12. Conclusion

Human beings communicate through situated intention. We do not merely exchange payloads; we establish a context boundary, recognize relevant perceptions inside it, resolve uncertainty, and act. Traditional software often hides this boundary inside code and convention. Intention Space makes it explicit.

For Perceptive App development, this is a foundational move. By treating Intentions as communication anchors and Signals as context-carrying runtime units, applications can maintain finer granularity of meaning than traditional payload-centered architectures. They can represent not only what value was passed, but why it

was passed, under which situational boundary, and with which perceptual uncertainties still unresolved.

This is the sense in which Intention Space mimics the human mode of communication: it does not reduce communication to data transfer. It treats communication as intentional, situated, and perceptually bounded.

References and Adjacent Literature

Austin, J. L. (1962). *How to Do Things with Words*. Oxford University Press.

Bratman, M. E. (1992). Shared cooperative activity. *The Philosophical Review*, 101(2), 327-341.

Clark, H. H., & Brennan, S. E. (1991). Grounding in communication. In L. B. Resnick, J. M. Levine, & S. D. Teasley (Eds.), *Perspectives on Socially Shared Cognition*. American Psychological Association.

Grice, H. P. (1957). Meaning. *The Philosophical Review*, 66(3), 377-388.

Mehmood, K., Kralevska, K., & Palma, D. (2023). Intent-driven autonomous network and service management in future cellular networks: A structured literature review. *Computer Networks*, 220, 109477. https://doi.org/10.1016/j.comnet.2022.109477

Mei, L., Yao, J., Ge, Y., Wang, Y., Bi, B., Cai, Y., Liu, J., Li, M., Li, Z., Zhang, D., Zhou, C., Mao, J., Xia, T., Guo, J., & Liu, S. (2025). A survey of context engineering for large language models. arXiv:2507.13334. https://arxiv.org/abs/2507.13334

Model Context Protocol. (2026). What is the Model Context Protocol? https://modelcontextprotocol.io/docs/getting-started/intro

OpenAPI Initiative. (2025). *OpenAPI Specification v3.2.0*. https://spec.openapis.org/oas/latest.html

Pal, P. (2024). Human Intention Space - Natural Language Phrase Driven Approach to Place Social Computing Interaction in a Designed Space. *International Journal on Natural Language Computing*, 13(3). https://aircconline.com/abstract/ijnlc/v13n3/13324ijnlc02.html

Searle, J. R. (1969). *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press.

Schweikard, D. P., & Schmid, H. B. (2017). Shared agency. *The Stanford Encyclopedia of Philosophy*. https://plato.stanford.edu/entries/shared-agency/

Tomasello, M., Carpenter, M., Call, J., Behne, T., & Moll, H. (2005). Understanding and sharing intentions: The origins of cultural cognition. *Behavioral and Brain Sciences*, 28(5), 675-691.