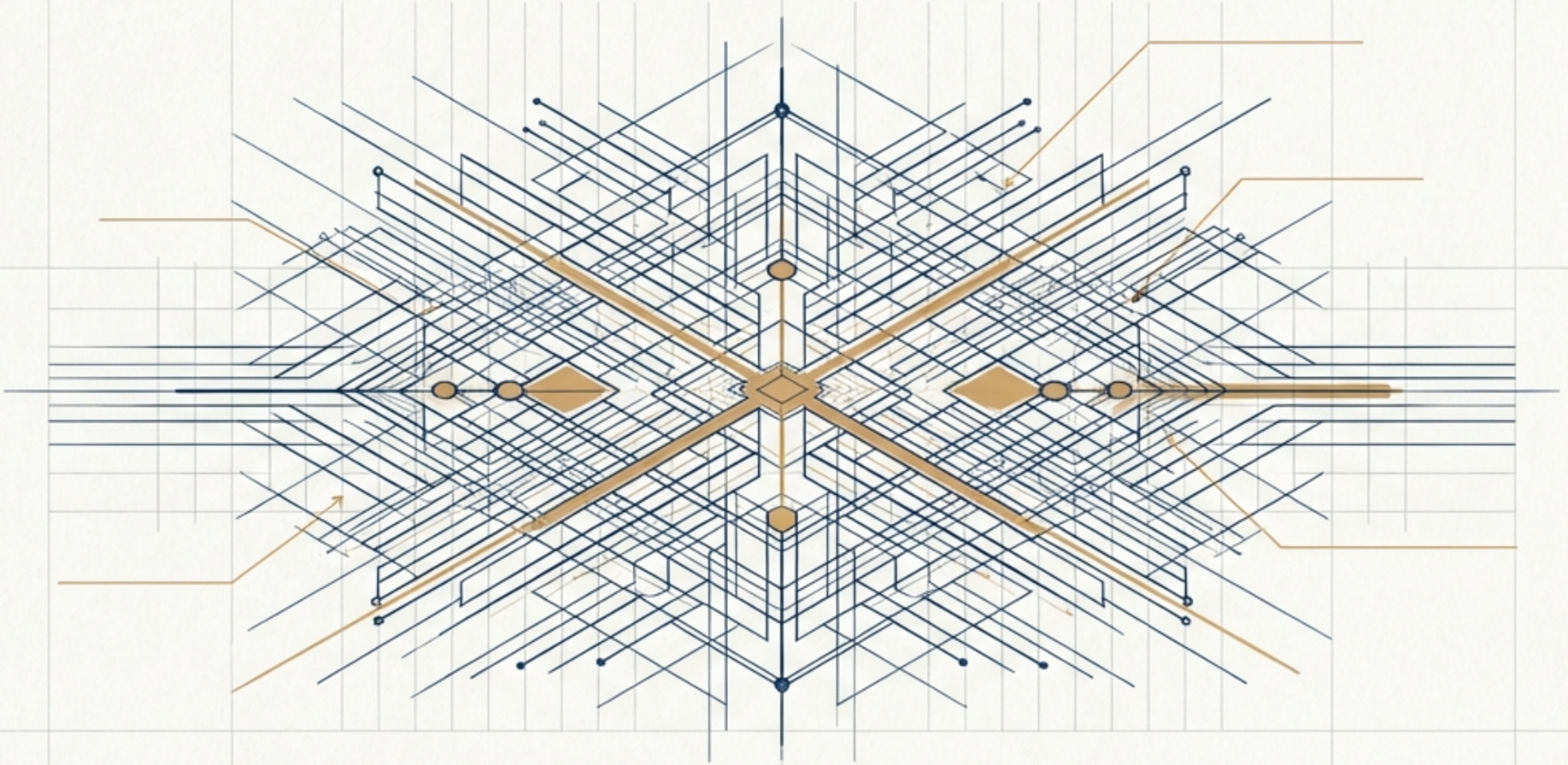
The Blueprint for a Global Understanding Environment

An Architecture for the Next Generation of Ubiquitous Computing



UbiWare Project

The Inevitable Future is an ‘Ultimate Interoperability Nightmare’

The vision of Ubiquitous Computing and the Internet of Things (IoT) promises a world of interconnected physical and digital resources—from machines and sensors to data services and humans. However, this massive interconnectivity of heterogeneous components (different standards, data formats, protocols) creates exponential complexity.

Without a new approach, this leads to a system that is impossible for humans to manage and for architects to design for proactively. The system becomes brittle, unmanageable, and unpredictable.

> *“The nightmare of ubiquitous computing… in which human operators will be unable to manage the complexity of interactions in the system, neither even architects will be able to anticipate that complexity.”*

Our Vision: The Global Understanding Environment (GUE)

A world where heterogeneous resources—physical, digital, and human—are not just connected, but can proactively cooperate and be understood globally.

Web-Accessible:
All resources are accessible through a unified framework.

Proactive:
Resources are not passive; they can initiate actions and self-manage.

Cooperative:
Resources can discover each other and work together to achieve complex goals.

Three Foundational Pillars for a Global Understanding Environment

Interoperability

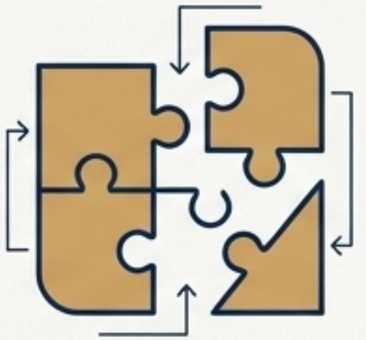
Challenge: How can radically different systems, devices, and data sources understand each other's meaning and capabilities on the fly?

Need: A move beyond rigid, pre-defined standards towards dynamic, meaningful data exchange.

Automation

Challenge: How can a system of billions of components manage itself without constant human intervention?

Need: Components must be proactive and autonomous, capable of 'running themselves' based on high-level policies.



Integration

Challenge: How can individual components be composed into complex, coherent business processes dynamically?

Need: A framework for coordinating the behavior and goals of autonomous components.

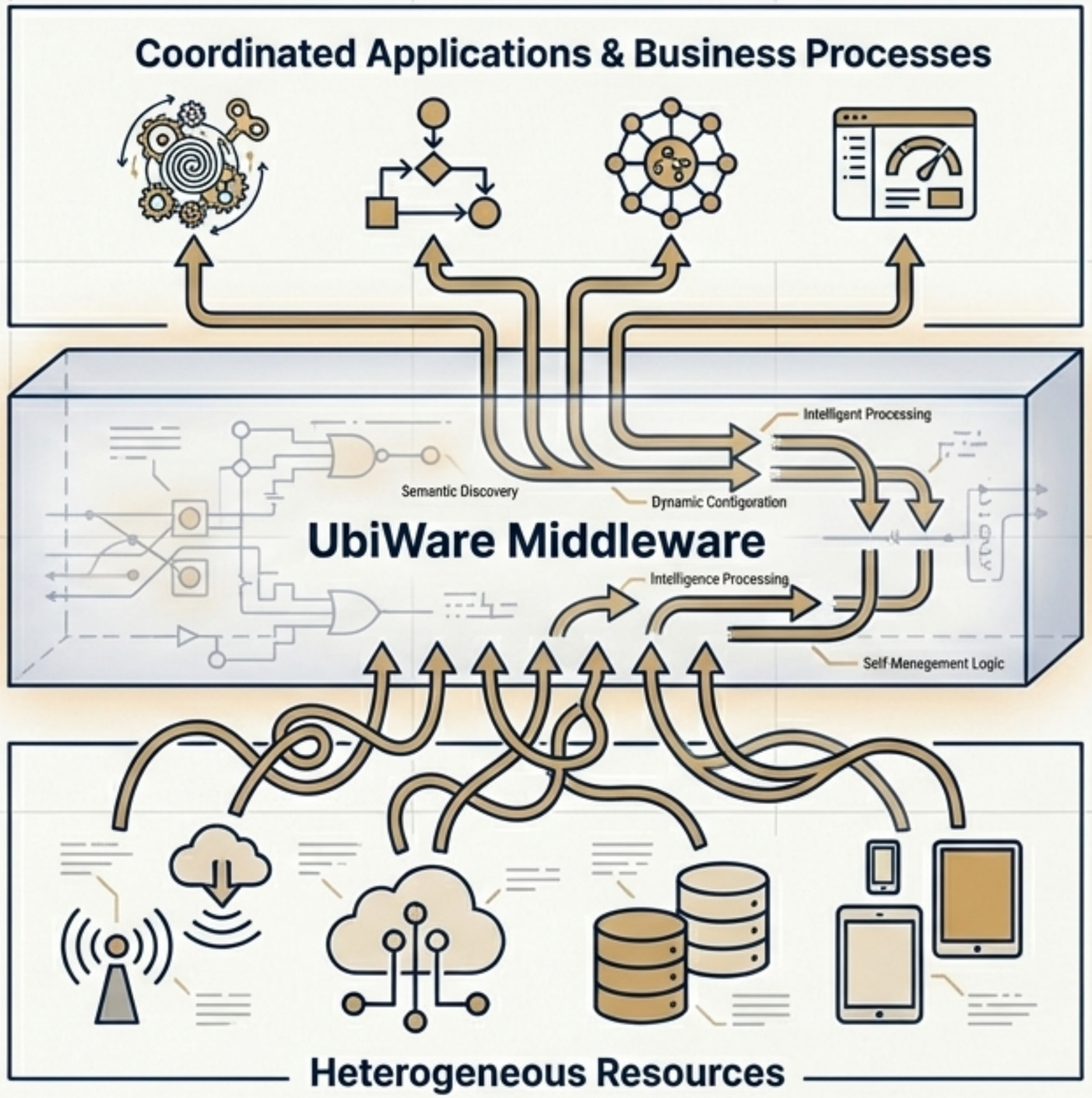
The Architectural Solution: Smart Semantic Middleware as the System's "Glue"

To achieve the GUE, we need an intermediary layer—a middleware—that can bridge heterogeneous components.

Traditional middleware is insufficient. The solution requires a new generation of **Smart Semantic Middleware**.

Our research project, **UbiWare**, is a platform designed to be this middleware. It enables the creation of self-managing, complex systems from distributed, reusable components.

> UbiWare is designed to allow components to automatically discover each other and configure a system with complex functionality based on their atomic capabilities.

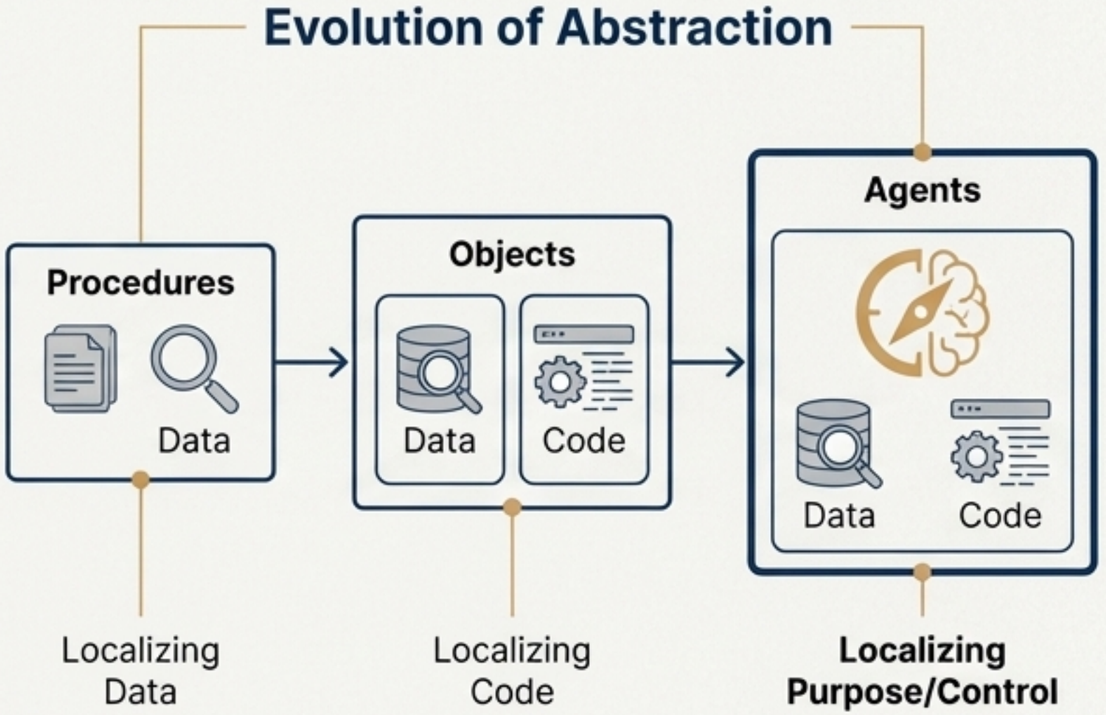


Powering Automation Through an Agent-Oriented Paradigm

> An encapsulated computer system situated in an environment, capable of flexible, autonomous action to meet its design objectives. (Wooldridge, 1999)

Why Agents are the Next Step in Software Engineering

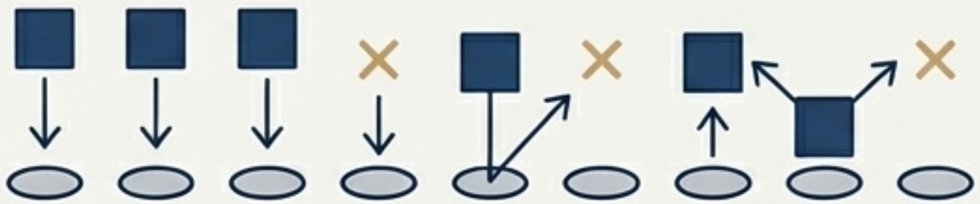
- **Bridging the Gap:** The agent concept minimizes the "semantic distance" between the problem domain (goals, stakeholders) and the solution domain (code), simplifying the design of complex systems.
- **Enabling Autonomy:** This paradigm is fundamental to creating the self-managing, self-configuring, and self-healing systems required by the Autonomic Computing vision.



Achieving True Interoperability with Semantic Technology

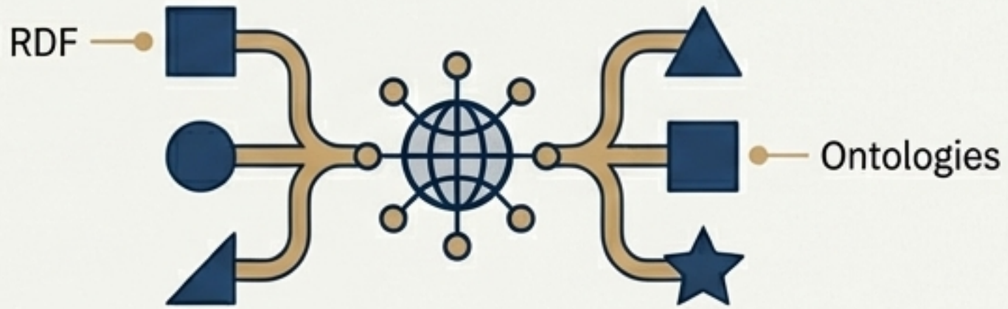
The Problem with Rigid Standards

Imposing a single standard on all components in a ubiquitous environment is not feasible.



The Semantic Solution

We use Semantic Web technologies (RDF, Ontologies) to describe the properties, capabilities, and behaviors of resources. This creates a machine-understandable layer of meaning.

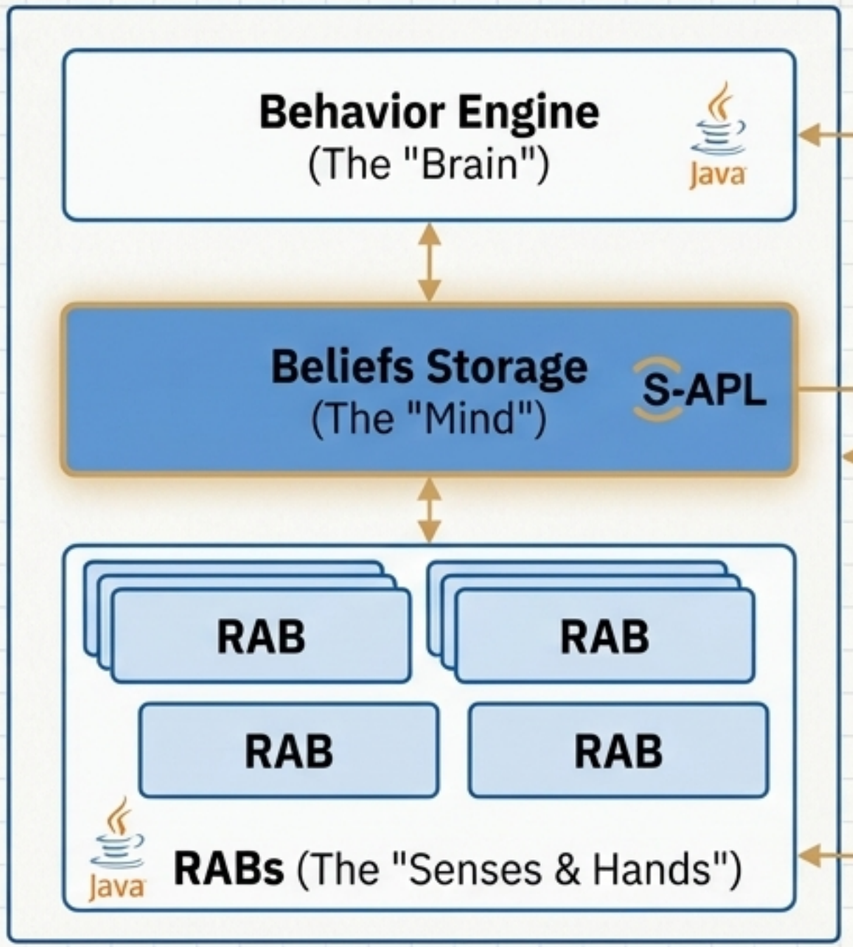


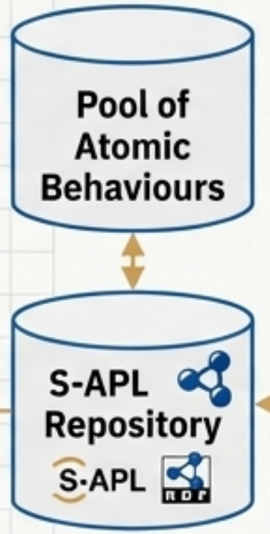
The Ultimate Goal: "Serendipitous Interoperability"

> *"...the ability of software systems to discover and utilize services they have not seen before, and that were not considered when and where the systems were designed."*

This is only possible with a rich, semantic description of services, enabling fully automated discovery and invocation without human intervention or prior agreements.

The UbiWare Agent Architecture: A 3-Layer Semantic Core





The core Java-based execution and decision-making component.

The declarative middle-layer, implemented in **S-APL**. This is the key innovation. It is RDF-based, meaning both data (beliefs) and program code (rules) are treated as semantic triples in a single, unified store. This allows rules to modify other rules, enabling powerful meta-reasoning.

The agent can dynamically query S-APL repositories for new rules/roles and load RABs from a pool of atomic behaviours.

Java components that act as sensors (perceiving the environment) and actuators (acting upon it). These are the agent's interface to the outside world.

S-APL: A Semantic Agent Programming Language

The Need for a New Language

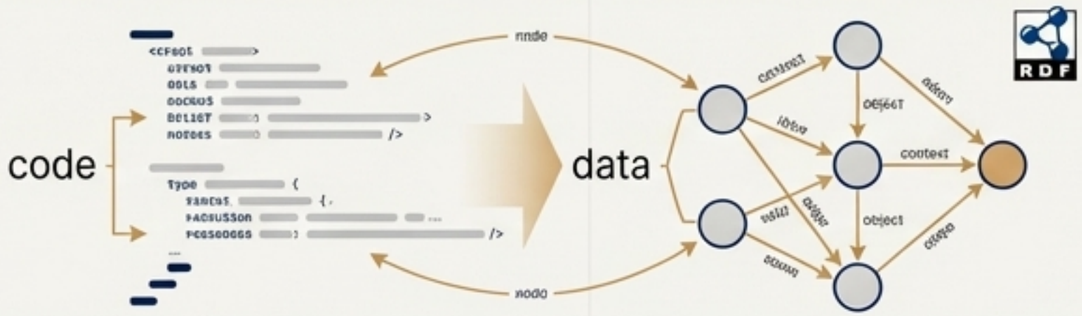
Traditional Agent Programming Languages (APLs) treat rules and beliefs as separate entities. This limits an agent's ability to reason about its own behavior or share it effectively.

The S-APL Axioms

1. **Everything is a belief**: Goals, commitments, and behavioral rules are all just complex beliefs.
2. **Every belief is a semantic statement**: Represented as an RDF triple (subject-predicate-object) or a set of such statements.
3. **Every belief has a context**: Beliefs are only valid within a specific scope, allowing for hierarchical and modular knowledge.

Key Advantage

Because S-APL code *is* RDF data, an agent's behaviors (roles, plans, rules) can be queried, shared, and reasoned about by other agents at runtime using the same mechanisms as any other data.



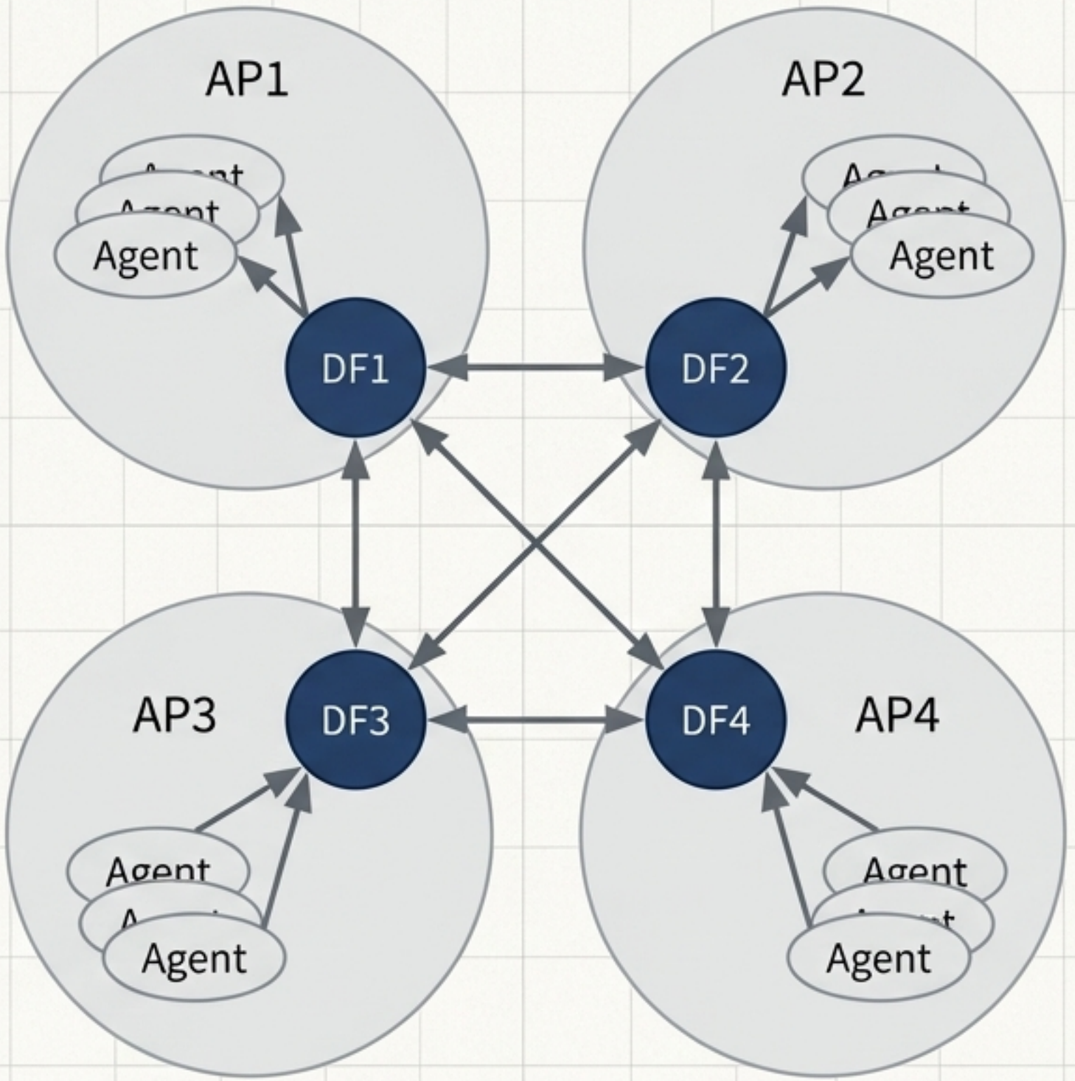
Scaling Discovery in a Multi-Platform Environment

The Challenge: In a real-world deployment, multiple UbiWare platforms (applications) will coexist. How can an agent on Platform A discover a service or agent on Platform B without a centralized registry?

Solution: Peer-to-Peer (P2P) Discovery

We extend the UbiWare framework with mechanisms for inter-platform discovery. Three models are explored:

1. **Centralized Directory:** A single platform acts as a master directory (simple but not robust).
2. **Federated Directories:** Directories register with each other, propagating queries through the federation (based on FIPA standards).
3. **Pure P2P (Gnutella-style):** Each platform ("servent") discovers others by propagating "Ping" messages, building a dynamic network topology.



The Human Interface: Making a Complex World Comprehensible

The Data Overload Problem

The GUE connects a potentially huge number of real and virtual resources. Classical search and browsing models fail because they are context-free and overwhelm the user.

The Solution: The 4i (Intelligent Interface for Integrated Information) Framework

A technology for dynamic, context-aware 3D/2D visualization. It enables the creation of a smart human interface through the collaboration of multiple agents: the user's agent, agents of the resources of interest, and agents representing visualization services (**MetaProviders**).

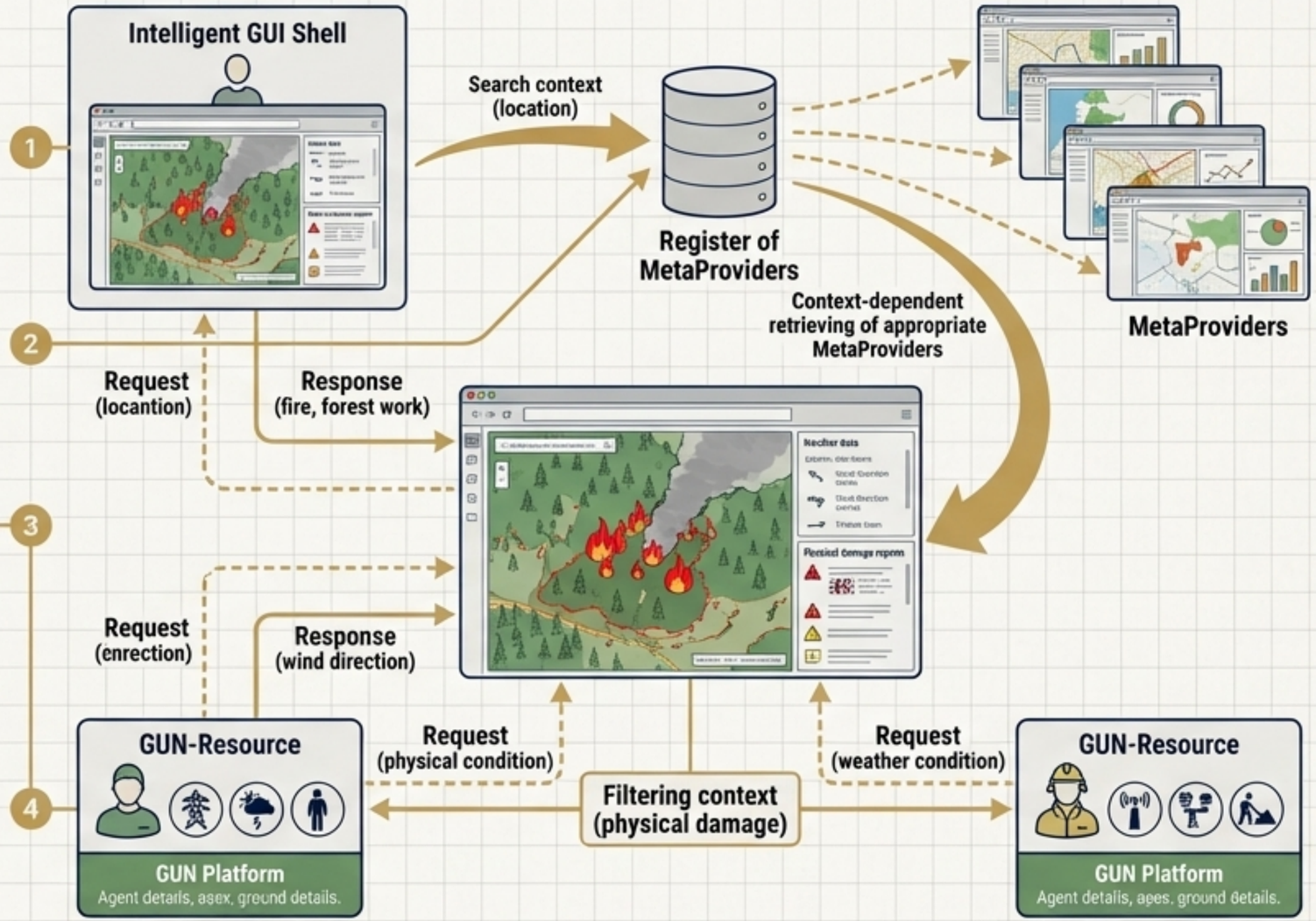
What is a MetaProvider?

A specialized service that provides a specific visual representation (a map, a chart, a 3D model) for certain types of data and contexts. They act as thematic portals to the GUE's resources.

4i in Action: Context-Dependent Visualization

Scenario Walkthrough

1. **User Context:** A human operator is managing a forest fire. Their 'Intelligent GUI Shell' sets the initial context (e.g., location, task).
2. **MetaProvider Discovery:** The shell queries a 'Register of MetaProviders' and, based on the context, selects the most appropriate one (e.g., a cartographic/GIS MetaProvider).
3. **Resource Integration:** The user requests data from relevant GUN-Resources (e.g., weather sensors, ground teams). The resources' agents
provide their data (wind direction, physical damage reports).
4. **Contextual Filtering & Display:** The MetaProvider receives this data, filters it based on the user's task (the "filtering context"), and renders an integrated, intuitive view (e.g., overlaying wind data and damage reports on a map). The user gets integrated information, not raw data streams.



The UbiWare Roadmap: From Core Technologies to the GUE

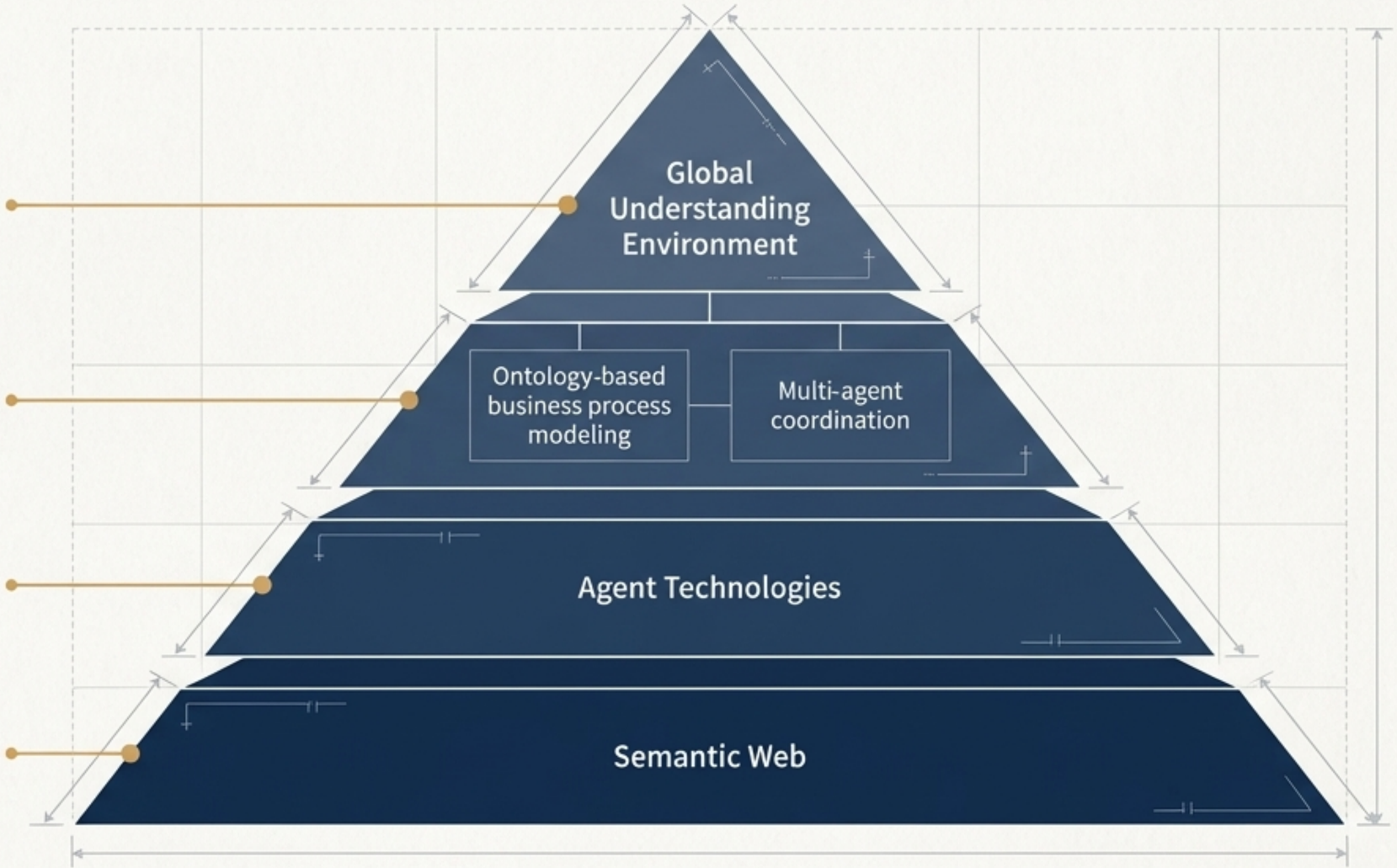
Connecting the Pieces

This complete stack forms the technological foundation required to realize the Global Understanding Environment.

Integration combines these using Ontology-based business process modeling and multi-agent coordination.

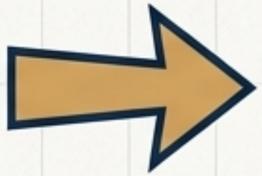
Automation is built upon this with proactive Agent Technologies.

Interoperability is achieved through Semantic Web standards at the base layer.



Transforming the Interoperability Nightmare into Global Understanding

Interoperability Nightmare



The UbiWare platform provides a concrete architectural **blueprint** and a robust set of technologies to build the future of ubiquitous computing.

An **agent-oriented approach** for true autonomy and self-management.

A deeply integrated **semantic core (S-APL)** where behavior and data are unified.

A **context-aware visualization framework (4i)** to make system complexity humanly manageable.

This is not just an incremental improvement; it is a foundational framework for building systems that can proactively understand and cooperate on a global scale.