POOKA™

An Architectural Style for Human–AI Information Architecture

A DESIGN PAPER

AUTHOR Maarten Meijer

STATUS Draft

VERSION 0.3

DATE July 2026

pooka.info · github.com/conceptblenders/POOKA

Contents

1. Introduction

1.1 About the Name

The name **POOKA** originally stood for **Personal Object-Oriented Knowledge Architecture**.

The concept originated from the practical challenge of organizing personal knowledge to enable more effective collaboration with AI. Initially, the focus was on structuring information, context and behavior within a personal knowledge environment.

As the underlying concepts evolved, it became clear that the architectural principles extended beyond personal knowledge management. The central question gradually shifted from *"How should I organize my own knowledge?"* to *"How should information be architected so that humans and AI can collaborate through a shared semantic structure?"*

Although the original acronym reflects the historical origin of the work, it no longer defines its scope. Today, **POOKA** refers to the architectural style itself. The original name has intentionally been preserved as a reference to the project's origin while allowing the architecture to evolve beyond its initial interpretation.

1.2 Purpose

The rapid adoption of generative AI has introduced a new way of working with information. Large Language Models have become increasingly capable of interpreting, generating, transforming and reasoning over knowledge. Yet the quality of collaboration between humans and AI remains highly dependent on the way information is organized, contextualized and governed.

Many current AI workflows reconstruct context during every interaction. Prompts are used to establish roles, context, behavioral expectations, terminology and relevant knowledge at runtime. While this provides considerable flexibility, it also makes collaboration dependent on prompt quality, repetition and conversational continuity. As a consequence, context remains transient, semantics remain implicit, and knowledge is difficult to reuse consistently across domains, tools and AI models.

POOKA addresses this challenge by defining context, semantics, governance and behavior as persistent architectural elements rather than transient prompt content. Instead of reconstructing these concepts during every interaction, POOKA defines them as part of the information architecture itself, shifting a significant part of the responsibility for context from runtime prompt engineering to architectural design.

POOKA defines an architectural style for **Human–AI Information Architecture**. It defines a conceptual model in which information, semantics, governance and behavior are organized explicitly, allowing humans and AI systems to collaborate from a shared architectural foundation.

POOKA does not define software, programming models, storage technologies or implementation frameworks. It defines a design philosophy for organizing information independently of AI models, applications and technical implementations.

The purpose of this paper is to define that architectural philosophy, establish a common conceptual language, and provide a foundation for discussion, implementation, evaluation and further refinement.

2. The Architectural Challenge

The emergence of Large Language Models has fundamentally changed the way humans interact with information. Rather than retrieving information through predefined interfaces, AI enables users to engage in natural language conversations capable of reasoning, summarizing, generating and transforming knowledge.

Despite these advances, the underlying organization of information has changed far less than the interaction itself. Most information remains fragmented across documents, applications, cloud services, conversations and personal notes. Structure, meaning and governance are often implicit, requiring AI to reconstruct them during every interaction.

As a consequence, collaboration increasingly depends on prompt engineering. Prompts establish roles, objectives, behavioral expectations, terminology, context and relevant background information at runtime. While this approach is flexible and often highly effective, it also produces context that is transient, difficult to maintain and challenging to reuse consistently across conversations, AI models and technical environments.

In architectural terms, a prompt is the means by which context and behavior are supplied to an AI system at the moment of interaction, and prompt engineering is the practice of constructing these instructions. Understood in this way, a prompt is a runtime carrier of context rather than a persistent part of the information architecture.

POOKA therefore interprets the growing importance of prompt engineering not as an architectural solution, but as an architectural symptom: it suggests the absence of persistent structures capable of representing meaning, context, governance and behavior independently of individual conversations.

Human collaboration rarely depends on repeatedly explaining identity, responsibilities, terminology or organizational context before every interaction. These elements persist beyond individual conversations and form part of a shared understanding. POOKA assumes that Human–AI collaboration requires similar persistence if it is to scale beyond isolated interactions.

POOKA is based on the assumption that this challenge is not primarily technological. More capable AI models reduce many limitations of reasoning, language understanding and generation, but they are not expected to eliminate the need for explicit organization of information. On this view, improvements in AI performance complement rather than replace information architecture.

POOKA approaches this challenge by treating information architecture as the primary carrier of context. Rather than reconstructing architectural knowledge through prompts, POOKA defines persistent architectural concepts that can be interpreted consistently by both humans and AI.

In this paper, persistence refers to the architectural representation of such concepts rather than to their content. A Context, for example, is represented persistently as part of the information architecture, while the Context itself remains adaptive and continues to evolve as work, conversations and objectives change. Persistent therefore means explicitly represented and maintained across interactions, rather than static.

The architectural challenge addressed by POOKA is therefore not how to create better prompts, but how to design information environments in which prompts become consumers of architecture rather than its primary source. This contrast is illustrated in Figure 1 (diagrams/prompt-vs-architecture.drawio).

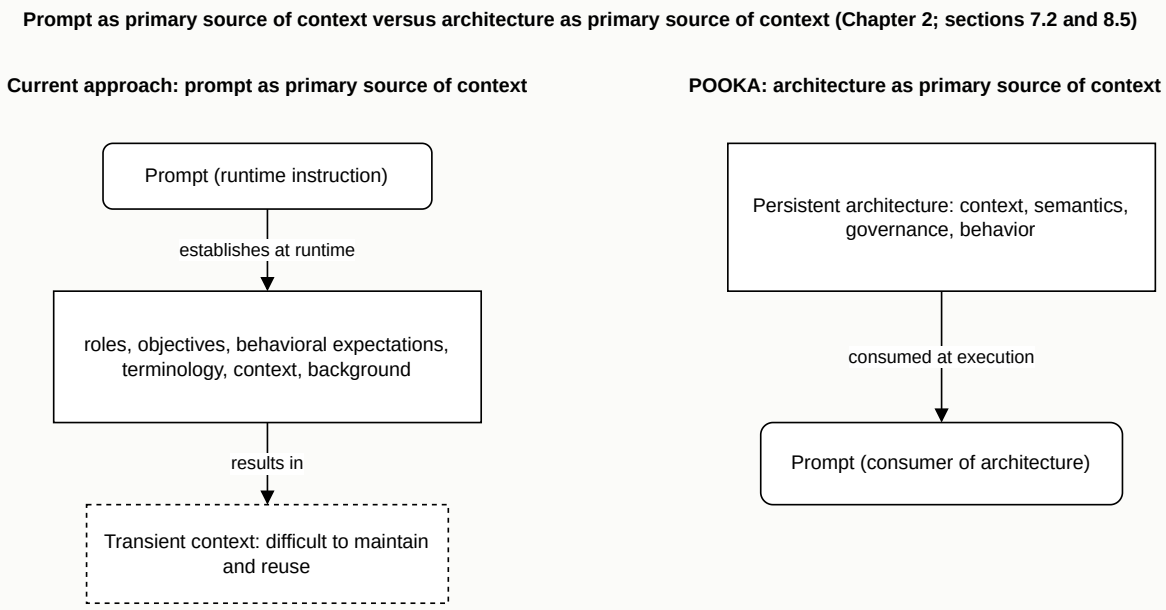

Figure 1. Runtime prompt context contrasted with persistent architecture.

3. Status

POOKA is an architectural philosophy emerging from practical experimentation with Human–AI collaboration.

It is not a scientific theory, formal standard or industry specification. The concepts presented in this paper are design proposals derived from practical observation, iterative refinement and architectural reasoning. They have not been empirically validated through controlled scientific research and should therefore not be interpreted as established facts.

POOKA intentionally builds upon existing knowledge from information architecture, software architecture, knowledge management, object-oriented design, graph-based modeling, identity management and AI-assisted knowledge work. Where existing concepts adequately describe a problem, they are adopted rather than reinvented. Where existing concepts do not sufficiently address the architectural challenges of Human–AI collaboration, POOKA introduces new abstractions and relationships.

This paper therefore should be read as a design paper: an invitation to discussion, implementation, validation and further evolution.

4. Scope

POOKA defines an architectural style for organizing information intended for collaboration between humans and AI.

The architecture focuses on the explicit organization of meaning, context, governance and behavior. It defines a conceptual model that can be implemented using different technologies, storage mechanisms and AI systems.

POOKA intentionally does not prescribe:

- programming languages;
- software architectures;
- database technologies;
- file formats;
- AI models;
- implementation frameworks;
- user interfaces.

Likewise, POOKA does not attempt to replace existing architectural disciplines such as Enterprise Architecture, Information Architecture, Knowledge Management or Identity & Access Management. Instead, it provides an architectural perspective specifically focused on sustainable Human–AI collaboration.

Any implementation that preserves the conceptual model while using different technical solutions may still conform to POOKA.

5. Foundational Assumptions

POOKA is based on several architectural assumptions. These assumptions are not presented as universally proven truths but as the foundation upon which the architecture has been developed.

Information outlives interaction

Conversations are temporary. Information is persistent. Sustainable Human–AI collaboration requires knowledge to exist independently of individual conversations.

Meaning should be explicit

Information becomes more reusable when its semantics are represented explicitly rather than being inferred from conversational context.

Context is architecture

Context should not primarily be reconstructed through prompts. It should be represented as a persistent architectural concept.

Governance is part of information

Questions such as who may act, on whose behalf, under which conditions and within which boundaries belong to the architecture itself rather than to implementation logic.

Structure precedes automation

AI should operate on well-defined structures instead of compensating for missing structure through increasingly complex prompts.

Human judgement remains external

AI may analyse, suggest, summarize, transform and reason. Responsibility for judgement and decision-making remains outside the AI itself.

Architectures should survive implementations

Architectural concepts should remain applicable regardless of future AI models, software platforms or storage technologies.

6. Existing Perspectives

POOKA is not developed in isolation. It builds upon concepts that have emerged across multiple disciplines, including Information Architecture, Software Architecture, Knowledge Management, Identity & Access Management, Object-Oriented Design, graph-based modeling and Human–Computer Interaction.

Rather than replacing these disciplines, POOKA brings together selected principles within a single architectural perspective focused on sustainable Human–AI collaboration.

6.1 Information Architecture

Traditional Information Architecture focuses on organizing, structuring and presenting information so that it can be effectively understood and accessed by people.

POOKA extends this perspective by treating AI systems as architectural participants alongside human users. Information is therefore organized not only for human interpretation, but also for consistent interpretation by AI.

6.2 Knowledge Management

Knowledge Management addresses the creation, organization, sharing and preservation of organizational knowledge.

POOKA shares these objectives but focuses specifically on the architectural organization of knowledge as a persistent foundation for Human–AI collaboration, independent of specific organizational processes or technologies.

6.3 Object-Oriented Design

Object-Oriented Design introduced concepts such as encapsulation, composition and explicit relationships between objects.

POOKA adopts a similar architectural mindset by treating information as explicit architectural concepts with clearly defined responsibilities and relationships. However, these concepts represent information architecture rather than software components.

6.4 Domain-Driven Design

Domain-Driven Design emphasizes the importance of modeling complex domains through explicit language, bounded contexts and domain concepts.

POOKA adopts similar principles for organizing information but applies them beyond software design. Domains and Contexts become architectural constructs for organizing human and AI knowledge rather than software models alone.

6.5 Identity & Access Management

Identity & Access Management defines how identities are authenticated, authorized and represented within digital systems.

POOKA adopts the distinction between Identity and operational actors but extends it through explicit Delegation, allowing humans, AI systems and technical services to operate on behalf of an Identity while preserving governance and accountability.

6.6 Graph-Based Modeling

Graph-based models represent knowledge through nodes and explicit relationships.

POOKA shares the emphasis on explicit relationships but intentionally remains independent of graph technologies. Relations are architectural concepts that may be implemented using graph databases, relational databases, documents or other storage mechanisms.

6.7 Local-First Computing

Local-First Computing promotes user ownership, resilience and long-term control over information.

POOKA adopts similar principles by encouraging architectures in which knowledge remains under the control of the owning Identity whenever practical, while allowing cloud-based services and AI models to extend capabilities where appropriate.

6.8 Software Architecture

Software Architecture is concerned with the structure of software systems, including their components, responsibilities and the principles that guide their design and evolution.

POOKA adopts an architectural mindset but applies it to the organization of information rather than to software components. Its concepts describe an information architecture that remains independent of any particular software system.

6.9 Human–Computer Interaction

Human–Computer Interaction studies how people interact with computing systems and how those systems can be designed to support human understanding and control.

POOKA shares this concern for human understanding and control but addresses it at the level of information architecture. It focuses on how information is organized so that humans and AI can collaborate, rather than on the design of specific user interfaces.

6.10 Context-Aware Computing

Context-Aware Computing studies how systems can perceive and respond to the situation in which they are used. Within this field, context is commonly understood as any information that characterizes the situation of an entity.

POOKA shares the recognition that context determines how information should be interpreted, but treats Context as an explicit architectural construct rather than a condition sensed at runtime. Context in POOKA is represented, selected and governed as part of the information architecture rather than inferred from the environment.

6.11 Provenance Models

Provenance models, such as the W3C PROV family of specifications, describe how information came into existence by modeling entities, activities and agents, including agents acting on behalf of other agents.

Several POOKA concepts have close counterparts in these models: Artifacts resemble entities, Events resemble activities, Actors resemble agents and Delegation resembles acting on behalf of another. POOKA builds on this lineage but differs in scope: provenance models describe how information was produced, whereas POOKA organizes information, semantics, governance and behavior for ongoing Human–AI collaboration.

6.12 Personal Data Stores

Personal data store initiatives, such as Solid, place personal information under the control of an identity and allow applications and agents to access that information through explicit permissions.

POOKA shares this identity-centric perspective and its emphasis on explicit access, particularly in personal knowledge environments. It differs by defining a technology-independent architectural style rather than a platform or protocol: a personal data store may provide one possible implementation environment for POOKA concepts, but POOKA itself prescribes no specific infrastructure.

6.13 Multi-Agent Systems

Research on multi-agent systems addresses how autonomous software agents cooperate, including how norms and organizational rules constrain agent behavior.

POOKA addresses a related concern through Behavior, Delegation and Boundaries, which constrain how Actors, including AI systems, may operate. It differs in focus: multi-agent systems research primarily concerns interaction among autonomous software agents, whereas POOKA defines the information architecture within which humans and AI collaborate.

6.14 Positioning

POOKA should not be viewed as an alternative to these disciplines. Instead, it can be understood as an architectural synthesis that combines established concepts into a coherent model for Human–AI Information Architecture.

Its contribution lies not in replacing existing theories, but in connecting them within a common architectural language designed specifically for collaboration between humans and AI.

7. Design Principles

The following Design Principles form the architectural foundation of POOKA. They describe how information should be organized to enable sustainable collaboration between humans and AI. They are normative principles that guide the architecture itself rather than specific technical implementations.

7.1 Meaning Before Mechanics

Architecture begins with meaning, not implementation.

Information should first describe what something is before defining how it is stored, processed or presented. Technologies, storage mechanisms and AI models evolve continuously; meaning should remain stable.

POOKA therefore models information independently from implementation technologies.

7.2 Context Before Prompt

Context is an architectural concept rather than a conversational construct.

Current AI workflows frequently establish context through prompts at runtime. While effective for individual interactions, this makes context transient and difficult to reuse consistently.

POOKA defines Context explicitly as part of the information architecture, allowing prompts to become consumers of context rather than its primary source. It is the representation of Context that persists; the Context itself remains adaptive.

7.3 Explicit Semantics

Semantics should never rely solely on interpretation.

Concepts, relationships and meaning should be represented explicitly wherever possible, reducing ambiguity for both humans and AI.

Explicit semantics are intended to improve consistency, discoverability and long-term maintainability of knowledge.

7.4 Explicit Relationships

Knowledge does not exist in isolation.

Relationships between concepts should be represented explicitly rather than inferred through textual proximity or conversational history.

Understanding often emerges from relationships rather than from individual information objects.

7.5 Boundaries Are Architecture

Boundaries define the architecture as much as the information itself.

Domains, Contexts and other architectural constructs require explicit boundaries that determine scope, visibility, ownership, semantics and behavior.

Boundaries enable multiple perspectives to coexist without unnecessary coupling.

7.6 Representation Must Be Explicit

An Actor never implicitly represents an Identity.

Whenever an Actor performs actions on behalf of an Identity, this relationship should be explicitly defined through Delegation.

Representation determines authority, responsibility and behavioral constraints and therefore forms part of the architecture itself.

7.7 Structure Before Automation

POOKA assumes that Artificial Intelligence operates most effectively on well-structured information.

Rather than compensating for missing structure through increasingly complex prompts, POOKA encourages explicit modeling of knowledge before introducing automation.

Automation should emerge from architecture rather than replace it.

7.8 Minimum Necessary Context

AI should only receive the information required to perform the intended task.

Providing unnecessary context is expected to increase complexity, computational cost and the likelihood of unintended reasoning.

POOKA therefore promotes the explicit selection of relevant Context rather than exposing complete knowledge environments by default.

7.9 Local First, Cloud When Necessary

Architectural decisions should preserve autonomy wherever practical.

Knowledge should remain under the control of its owning Identity whenever possible. External services and cloud-based AI can extend capabilities but should not become architectural dependencies without explicit reason.

This principle promotes portability, resilience and long-term sustainability.

7.10 Architectures Outlive Implementations

Architectural concepts should survive technological change.

POOKA intentionally separates conceptual architecture from implementation. Storage technologies, AI models, programming languages and software platforms may change over time without affecting the underlying architectural principles.

The architecture should remain valid even when today's technologies no longer exist.

8. Core Concepts

POOKA defines a small set of fundamental architectural concepts. Together these concepts establish a common language for organizing information, semantics, governance and behavior within Human–AI Information Architecture.

POOKA intentionally limits itself to a small set of Core Concepts. These concepts are not intended to describe every aspect of knowledge or collaboration, but to capture the architectural abstractions considered necessary to describe Human–AI Information Architecture. Further detail is expected to emerge within implementations and domains rather than within the architecture itself.

The Core Concepts are intentionally heterogeneous: they include persistent structures, relationships, semantic abstractions, behavioral abstractions and architectural constraints. All are presented as Core Concepts because each represents an irreducible architectural abstraction required to describe a POOKA-conformant Human–AI Information Architecture. Equal status as a Core Concept therefore does not imply an identical architectural role.

Each concept has a single, well-defined meaning throughout this paper. Implementations may extend these concepts, but should not redefine them. The Core Concepts and their relationships are illustrated in Figure 2 (diagrams/core-concepts.drawio).

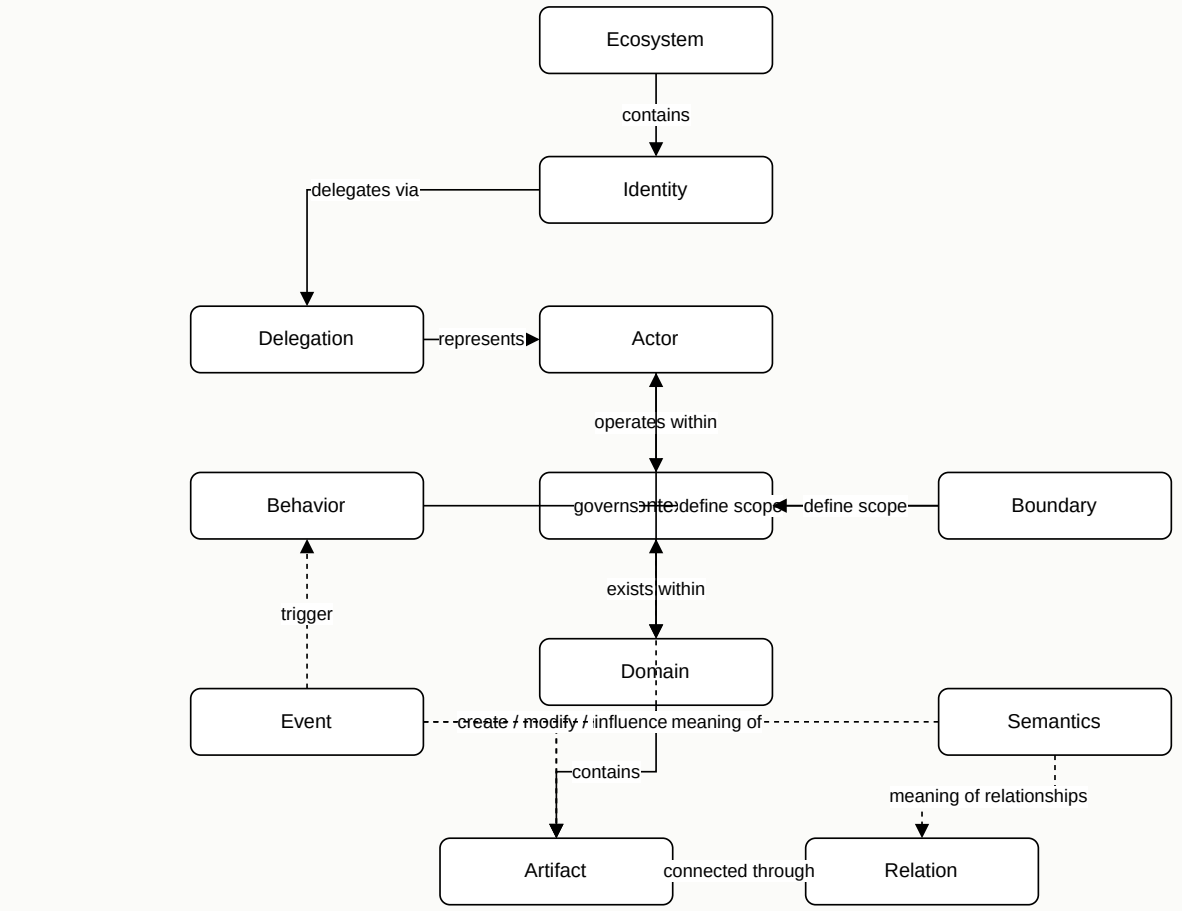

Figure 2. The Core Concepts and their relationships.

8.1 Ecosystem

An **Ecosystem** defines the complete architectural environment within which one or more Identities, Actors, Domains, Contexts and Artifacts coexist.

An Ecosystem establishes the highest architectural boundary of a POOKA implementation. It provides the environment in which information, governance and behavior are organized.

An Ecosystem may represent an individual, a family, an organization, a community or any other coherent knowledge environment.

8.2 Identity

An **Identity** represents a persistent person, organization or other identifiable subject within an Ecosystem.

Identity represents the enduring subject of the architecture rather than the individual Actors operating within it.

Multiple Actors may represent a single Identity through explicit Delegation.

8.3 Actor

An **Actor** defines any human, AI or technical system capable of performing actions within an Ecosystem.

Actors consume, create, modify and relate information according to the architectural constraints defined by POOKA.

An Actor never implicitly represents an Identity.

8.4 Delegation

Delegation defines how an Actor may represent an Identity.

Delegation specifies the scope, permissions, constraints and responsibilities under which an Actor operates.

Representation within POOKA is always explicit.

Delegation therefore forms the architectural bridge between Identity and Actor.

8.5 Domain

A **Domain** defines a durable and bounded knowledge environment.

Domains organize information around a coherent subject, objective or responsibility. They establish stable boundaries for knowledge, semantics, governance and behavior.

Domains are intended to remain relatively stable over time.

8.6 Context

A **Context** defines the active scope in which information is interpreted.

Contexts exist within Domains and provide the situational perspective required for collaboration between humans and AI.

Unlike Domains, Contexts are expected to change as work, conversations and objectives evolve.

8.7 Artifact

An **Artifact** defines any addressable unit of information.

Artifacts may represent documents, people, decisions, meetings, projects, ideas, tasks, datasets or other meaningful information objects.

Artifacts exist independently of the technology used to store them.

8.8 Relation

A **Relation** defines an explicit connection between architectural concepts.

Relations express meaning by describing how concepts are connected rather than relying upon textual proximity or implicit interpretation.

Relations may exist between Artifacts, Domains, Contexts, Actors or other architectural concepts.

8.9 Semantics

Semantics define the explicit meaning assigned to information.

Semantics describe what information represents rather than how it is stored or presented.

Explicit semantics are intended to reduce ambiguity and improve consistency for both humans and AI.

8.10 Behavior

Behavior defines how Actors or AI are expected, permitted or constrained to operate within the architecture.

Behavior may include operational rules, interaction patterns, stylistic guidance, reasoning constraints or other behavioral expectations.

Behavior is defined explicitly rather than embedded implicitly within prompts.

8.11 Boundary

A **Boundary** defines architectural separation.

Boundaries determine visibility, ownership, accessibility, semantic scope, behavioral scope and the conditions under which information may cross between architectural elements.

Boundaries are explicit architectural constructs rather than implementation details.

8.12 Event

An **Event** defines something that occurs at a specific point or period in time.

Events may create, modify or relate Artifacts, influence Context or trigger Behavior.

Unlike Artifacts, Events are inherently temporal and describe change rather than persistent knowledge.

9. Architectural Model

The architectural model describes how the Core Concepts interact to establish a sustainable environment for Human–AI collaboration.

Rather than focusing on software components or implementation details, POOKA models the architectural relationships between information, meaning, governance and behavior.

The architecture intentionally separates relatively stable concepts from concepts that are expected to change over time. This distinction allows knowledge to remain durable while enabling AI interactions to remain adaptive.

9.1 Architectural Layers

POOKA distinguishes three complementary architectural layers. These layers are illustrated in Figure 3 (diagrams/architectural-layers.drawio).

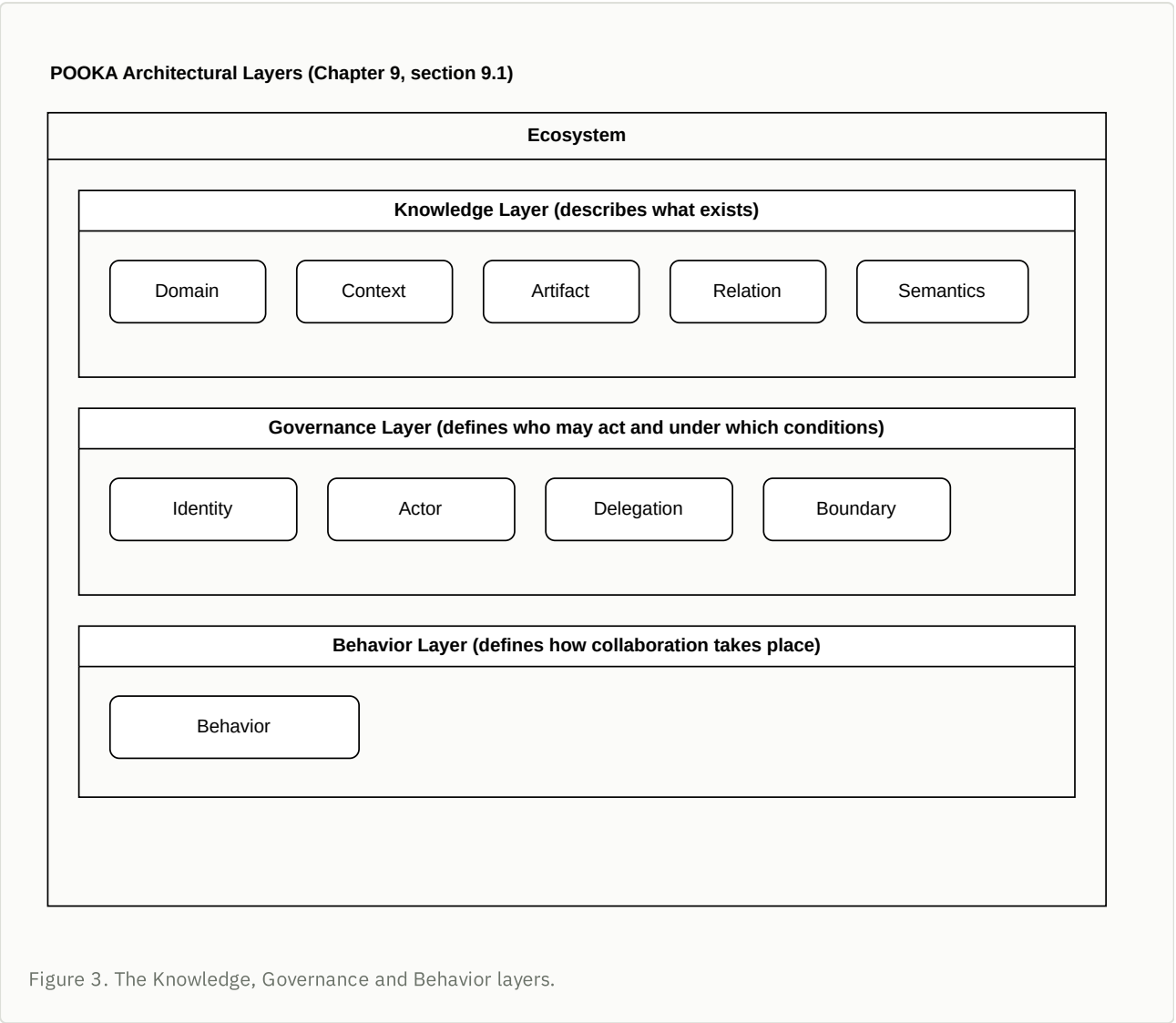

Figure 3. The Knowledge, Governance and Behavior layers.

Knowledge Layer

The Knowledge Layer describes what exists.

It contains the persistent representation of knowledge, independent of any specific interaction. Concepts such as Domains, Contexts, Artifacts, Relations and Semantics define the informational structure upon which collaboration is based.

Knowledge should remain stable regardless of which Actor accesses it or which AI model processes it.

Governance Layer

The Governance Layer defines who may act and under which conditions.

Identity, Actor, Delegation and Boundaries determine authority, representation, ownership and responsibility throughout the architecture.

Governance is considered part of the information architecture rather than application logic.

Behavior Layer

The Behavior Layer defines how collaboration takes place.

Behavior specifies how Actors, including AI systems, are expected or permitted to operate within a particular Context.

Unlike knowledge, behavior is intentionally adaptive. It may vary between Domains, Contexts or Delegations while remaining constrained by the architectural model.

9.2 Architectural Relationships

The Core Concepts form a connected architectural graph. This graph is illustrated in Figure 2 (diagrams/core-concepts.drawio).

An Ecosystem contains one or more Identities.

An Identity may delegate representation to one or more Actors.

Actors operate within one or more Contexts.

Contexts exist within Domains.

Domains contain Artifacts.

Artifacts are connected through explicit Relations.

Semantics define the meaning of architectural concepts and their relationships.

Behavior governs how Actors interact with information.

Boundaries constrain the movement of information, semantics and behavior between architectural elements.

Events introduce change into the architecture by creating, modifying or relating Artifacts and Contexts over time.

9.3 Persistence and Change

Not every architectural concept changes at the same rate.

POOKA distinguishes between relatively persistent architectural concepts and concepts that are expected to evolve continuously.

Examples of relatively persistent concepts include:

- Ecosystem
- Identity
- Domain
- Artifact
- Semantics

Examples of adaptive concepts include:

- Context
- Delegation
- Behavior
- Event

This distinction allows architectural stability without reducing operational flexibility.

9.4 Architectural Independence

The architectural model is independent of implementation technology.

The concepts defined by POOKA may be implemented using files, databases, graph technologies, object-oriented systems, document stores or future technologies not yet available.

Compliance with POOKA depends upon preserving the architectural concepts and their relationships rather than adopting a particular technical implementation.

9.5 An Illustrative Example

The following example illustrates how the Core Concepts operate together. It is deliberately conceptual and technology-independent: it describes an architecture, not an implementation, and none of its elements prescribe any particular tool, platform or product.

Consider an independent researcher who organizes a personal knowledge environment as an **Ecosystem**.

The researcher is represented as an **Identity**: the persistent subject within the Ecosystem, remaining stable regardless of which tools or assistants are used.

Two **Actors** operate within the Ecosystem: the researcher, acting as a human Actor, and an AI assistant.

The AI assistant does not implicitly represent the researcher. An explicit **Delegation** defines that the assistant may organize and summarize research material on behalf of the researcher, within a defined scope and without the authority to publish results or make decisions.

The Ecosystem contains a durable **Domain** for the research work, alongside other Domains such as personal administration.

Within the research Domain, the study currently being carried out forms a **Context**: the active scope in which information is interpreted. When the researcher begins a new study, the Context changes while the Domain remains stable. Each Context is represented persistently as part of the architecture, even though the active Context itself is adaptive.

The research Domain contains **Artifacts** such as interview notes, source materials and a draft article.

When a new interview takes place, this **Event** introduces change into the architecture: it creates a new interview-notes Artifact and influences the active Context.

An explicit **Relation** connects the draft article to the interview notes on which it is based, so that this connection does not depend on textual proximity or on the memory of a conversation.

Semantics define explicitly what these Artifacts represent: what an interview note is, what a source is and how they differ. The researcher and the AI assistant can therefore interpret the same information consistently.

Behavior defines how the AI assistant is expected to operate within this Context: for example, that summaries must remain neutral and must be distinguishable from the researcher's own conclusions.

A **Boundary** separates the research Domain from the personal administration Domain. The Delegation grants the AI assistant access to the former but not to the latter, and information does not cross this Boundary implicitly.

The same architecture could be realized using files, databases, graph structures or technologies that do not yet exist, and it corresponds to the Personal Knowledge Environment pattern described in Chapter 10.

10. Architectural Patterns

POOKA defines an architectural style rather than a single architecture. As a result, multiple architectural patterns may conform to POOKA while serving different purposes, scales and organizational contexts.

The following patterns illustrate how the Core Concepts may be combined without prescribing specific technologies, software platforms or implementation approaches.

10.1 Personal Knowledge Environment

A Personal Knowledge Environment represents the information architecture of a single Identity.

The Identity owns one or more Domains containing Contexts, Artifacts, Relations and Semantics. Human Actors, AI assistants and technical systems may collaborate through explicit Delegation while operating within the same architectural model.

This pattern provides a persistent information architecture independent of individual AI conversations.

10.2 Shared Knowledge Environment

A Shared Knowledge Environment enables multiple Identities to collaborate within a common Ecosystem.

Each Identity retains its own governance while sharing selected Domains, Contexts or Artifacts through explicitly defined Boundaries and Delegations.

The architecture supports collaboration without requiring ownership, authority or behavioral responsibilities to become implicit.

10.3 Organizational Knowledge Environment

Organizations often consist of multiple Domains, teams, departments and operational responsibilities.

POOKA enables these organizational structures to be represented explicitly while allowing human Actors, AI systems and technical services to operate within clearly defined architectural Boundaries.

Governance remains separated from implementation technology, allowing organizational responsibilities to evolve independently of software systems.

10.4 Distributed Knowledge Environment

Knowledge increasingly exists across multiple applications, cloud services, repositories and devices.

Rather than attempting to centralize all information, POOKA allows distributed information sources to participate within a common architectural model, provided their semantics, relationships and governance remain explicit.

The architecture therefore emphasizes conceptual consistency rather than physical centralization.

10.5 Federated Knowledge Environment

Independent Ecosystems may collaborate without becoming a single architecture.

Federation allows information, Contexts and selected Artifacts to be shared across Ecosystem Boundaries while preserving the autonomy, governance and Identity of each participating Ecosystem.

This pattern supports collaboration between individuals, organizations and technical platforms without requiring complete architectural integration.

10.6 Pattern Independence

These patterns are illustrative rather than exhaustive.

POOKA intentionally does not prescribe how architectural concepts should be combined for specific use cases. Additional patterns may emerge as practical experience with Human–AI Information Architecture evolves.

Conformance to POOKA is determined by adherence to its architectural principles and Core Concepts rather than by adopting a predefined architectural pattern.

11. Implementation Principles

POOKA intentionally separates architecture from implementation.

The architectural concepts defined by POOKA are independent of programming languages, storage technologies, AI models and software platforms. Multiple implementations may therefore conform to POOKA while making fundamentally different technical choices.

The purpose of this chapter is not to prescribe implementation, but to describe the characteristics that a conforming implementation should preserve.

11.1 Preserve Meaning

The primary responsibility of an implementation is to preserve meaning.

Architectural concepts such as Identity, Domain, Context and Artifact should remain explicit regardless of the underlying technology.

Implementations should avoid reducing architectural concepts to technical constructs that obscure their semantic meaning.

11.2 Preserve Relationships

Relationships between architectural concepts should remain explicit.

Implementations may represent relationships using references, graphs, object references, hyperlinks or other mechanisms, provided the relationship itself remains identifiable and meaningful.

Relationships should never rely solely on textual proximity or implicit interpretation.

11.3 Preserve Boundaries

Architectural Boundaries should remain visible throughout the implementation.

Whether implemented using folders, databases, access control, namespaces or graph partitions, Boundaries should continue to express ownership, visibility, responsibility and semantic scope.

Boundaries should not become implementation details.

11.4 Separate Architecture from AI

Artificial Intelligence consumes the architecture but does not define it.

AI models should operate upon the architectural concepts rather than replacing them.

Changes in AI technology should not require changes to the conceptual architecture.

11.5 Design Before Prompt

Prompt engineering remains a valuable implementation technique.

POOKA does not replace prompts; it reduces their architectural responsibility.

Information that is stable across interactions should preferably be represented as architecture rather than repeatedly reconstructed through prompts.

Prompt engineering therefore shifts from defining architecture at runtime towards consuming architecture during execution.

11.6 Technology Independence

A POOKA implementation should remain portable.

The architecture should not become dependent on any particular:

- AI model;
- software platform;
- storage technology;
- programming language;
- file format;
- cloud provider;
- vendor ecosystem.

Technology choices may change over time while preserving the architectural model.

11.7 Evolution

Architectures evolve.

Implementations should therefore allow Domains, Contexts, Behavior, Relations and Semantics to evolve without requiring structural redesign.

POOKA encourages iterative refinement while preserving conceptual consistency over time.

11.8 Conformance

Conformance to POOKA is conceptual rather than technical.

An implementation conforms to POOKA when it preserves the Core Concepts, their Relations and their Boundaries, maintains the conceptual distinction between the Knowledge, Governance and Behavior layers, and remains consistent with the Design Principles. These layers represent distinct architectural viewpoints on the same model rather than isolated partitions; conformance requires preserving their conceptual distinctions, not separating the concepts themselves. The architectural meaning of each concept should remain explicit and recognizable regardless of the technologies used to represent it.

Conformance therefore depends on preserving architecture rather than adopting particular tools. Two implementations may make entirely different technical choices and both conform, provided each retains the concepts, relationships and governance the architecture defines.

Conformance is a matter of architectural fidelity rather than certification. This paper defines no compliance tests, certification criteria or mandatory technologies. An implementation may preserve the architecture more or less completely, and partial adoption remains meaningful as long as the concepts it uses retain their defined meaning.

12. Limitations

Like every architectural style, POOKA addresses a specific class of architectural problems and should not be regarded as a universal solution for every Human–AI scenario.

The architecture intentionally focuses on the organization of information, semantics, governance and behavior. It does not attempt to replace existing disciplines, implementation technologies or AI capabilities.

12.1 Not an AI Framework

POOKA does not define AI functionality.

It does not prescribe language models, prompting techniques, reasoning methods, machine learning algorithms or agent implementations. These technologies may evolve independently while the architectural concepts remain applicable.

12.2 Not a Software Architecture

POOKA is not a software architecture or application framework.

It does not prescribe programming languages, APIs, databases, cloud platforms or deployment models. Multiple technical implementations may conform to POOKA while making entirely different engineering decisions.

12.3 Not a Knowledge Model

POOKA defines how knowledge is organized rather than what knowledge should contain.

The architecture intentionally avoids prescribing domain vocabularies, ontologies, taxonomies or semantic standards. These remain the responsibility of individual implementations and domains.

12.4 Human Responsibility

POOKA supports collaboration between humans and AI but does not transfer responsibility from humans to AI systems.

Architectural governance may define delegation, authority and behavioral constraints, but accountability for decisions remains with the responsible human or organization.

POOKA therefore assumes meaningful human oversight wherever decisions carry legal, ethical or organizational consequences.

12.5 No Guaranteed Outcomes

Adopting POOKA does not guarantee better AI performance, improved knowledge quality or organizational success.

The effectiveness of any implementation depends upon the quality of its information, governance, maintenance and practical application.

POOKA provides an architectural foundation rather than a guarantee of outcomes.

12.6 Evolution

The concepts described in this paper represent the current state of the architectural style.

Future experience, technological developments and community feedback may lead to refinement, extension or replacement of individual concepts while preserving the overall architectural philosophy.

POOKA should therefore be regarded as an evolving architectural style rather than a fixed specification.

13. Discussion

POOKA does not attempt to replace existing disciplines such as Information Architecture, Enterprise Architecture, Knowledge Management, Identity & Access Management or software architecture. Instead, it introduces an architectural perspective specifically focused on sustainable collaboration between humans and AI.

Many of the concepts described in POOKA build upon existing architectural principles. Concepts such as explicit semantics, bounded contexts, data minimization, local-first thinking, object-oriented design and graph-based relationships have well-established foundations in existing literature and practice. POOKA does not seek to redefine these concepts, but to integrate them into a coherent architectural style for Human–AI Information Architecture.

The primary contribution of POOKA is therefore not the invention of entirely new concepts, but the architectural synthesis of existing principles into a model that explicitly separates information, governance and behavior while treating context as a persistent architectural construct rather than transient conversational state.

POOKA should not be interpreted as a universal solution for every AI application. Different domains, organizations and technical environments may require alternative architectural decisions or additional concepts. The architecture intentionally remains technology-independent and leaves implementation choices to individual implementations.

As AI technology continues to evolve, new architectural challenges will inevitably emerge. POOKA should therefore be regarded as an evolving architectural philosophy rather than a fixed standard. Its long-term value depends on practical application, critical evaluation and continuous refinement by the broader community.

The concepts presented in this paper are intended to provide a common language for discussing Human–AI Information Architecture. Whether individual concepts remain unchanged, evolve or are eventually replaced is less important than establishing an explicit architectural dialogue around the organization of information for sustainable Human–AI collaboration.

14. Future Evolution

POOKA is intentionally designed as an evolving architectural philosophy.

The concepts presented in this paper should not be regarded as complete or final. As Human–AI collaboration continues to evolve, new architectural challenges, implementation patterns and conceptual refinements are expected to emerge.

Future evolution of POOKA may include, but is not limited to:

- refinement of the Core Concepts and their relationships;
- additional architectural patterns for specific domains;
- formal semantic models;
- reference implementations across different technologies;
- interoperability with existing architectural standards;
- methods for validating architectural consistency;
- governance models for collaborative knowledge environments.

The long-term development of POOKA depends on practical application, critical discussion and contributions from a broader community of practitioners, architects and researchers.

Architectural evolution should preserve conceptual consistency while allowing the model to adapt to new technologies, new forms of collaboration and future generations of AI.

15. Conclusion

Artificial Intelligence continues to advance rapidly. Models become more capable, interactions become more natural and new applications emerge almost daily. This paper has argued that sustainable Human–AI collaboration depends on more than increasingly capable AI systems: it also depends on the architecture through which information, meaning, context, governance and behavior are organized.

POOKA defines an architectural style that approaches this challenge from the perspective of information architecture rather than prompt engineering or software implementation. By treating context, semantics, governance and behavior as explicit architectural concepts, POOKA aims to establish a durable foundation upon which both humans and AI can collaborate.

The architecture intentionally remains independent of specific technologies, AI models and implementation choices. Its purpose is not to prescribe a single way of building Human–AI systems, but to provide a coherent conceptual framework that can be discussed, implemented, challenged and refined.

Whether POOKA ultimately proves valuable will not be determined by this paper alone, but by its ability to support practical implementations, encourage architectural dialogue and evolve through continued application and critical evaluation.

Like every architectural style, POOKA will ultimately be judged not by the elegance of its concepts, but by the quality, longevity and adaptability of the systems built upon it.

Terminology

Glossary of the POOKA Core Concepts. One concept has exactly one definition. The definitions below are derived from Chapter 8 of the paper, which remains the source of truth.

TERM	DEFINITION
Ecosystem	The complete architectural environment within which one or more Identities, Actors, Domains, Contexts and Artifacts coexist.
Identity	A persistent person, organization or other identifiable subject within an Ecosystem.
Actor	Any human, AI or technical system capable of performing actions within an Ecosystem.
Delegation	Defines how an Actor may represent an Identity, specifying the scope, permissions, constraints and responsibilities under which the Actor operates.
Domain	A durable and bounded knowledge environment.
Context	The active scope in which information is interpreted.
Artifact	Any addressable unit of information.
Relation	An explicit connection between architectural concepts.
Semantics	The explicit meaning assigned to information.
Behavior	How Actors or AI are expected, permitted or constrained to operate within the architecture.
Boundary	Architectural separation, determining visibility, ownership, accessibility, semantic scope, behavioral scope and the conditions under which information may cross between architectural elements.
Event	Something that occurs at a specific point or period in time and may create, modify or relate Artifacts, influence Context or trigger Behavior.

Bibliography

Well-established publications relevant to the disciplines and concepts discussed in the paper, in particular in Chapter 6 (Existing Perspectives). These references are prepared for future use; they are not yet cited inline in the paper.

Information Architecture

- Rosenfeld, L., Morville, P., & Arango, J. (2015). *Information Architecture: For the Web and Beyond* (4th ed.). O'Reilly Media.
- Wurman, R. S. (1997). *Information Architects*. Graphis.

Knowledge Management

- Nonaka, I., & Takeuchi, H. (1995). *The Knowledge-Creating Company*. Oxford University Press.
- Davenport, T. H., & Prusak, L. (1998). *Working Knowledge: How Organizations Manage What They Know*. Harvard Business School Press.
- Polanyi, M. (1966). *The Tacit Dimension*. University of Chicago Press.

Object-Oriented Design

- Booch, G., Maksimchuk, R. A., Engle, M. W., Young, B. J., Conallen, J., & Houston, K. A. (2007). *Object-Oriented Analysis and Design with Applications* (3rd ed.). Addison-Wesley.
- Meyer, B. (1997). *Object-Oriented Software Construction* (2nd ed.). Prentice Hall.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Domain-Driven Design

- Evans, E. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.
- Vernon, V. (2013). *Implementing Domain-Driven Design*. Addison-Wesley.

Identity & Access Management

- Windley, P. J. (2005). *Digital Identity*. O'Reilly Media.
- Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-Based Access Control Models. *IEEE Computer*, 29(2), 38–47.
- Hardt, D. (Ed.). (2012). *The OAuth 2.0 Authorization Framework* (RFC 6749). Internet Engineering Task Force.

Graph-Based Knowledge Representation

- Sowa, J. F. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole.
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5), 34–43.
- Hogan, A., Blomqvist, E., Cochez, M., et al. (2021). Knowledge Graphs. *ACM Computing Surveys*, 54(4), 1–37.

Local-First Software

- Kleppmann, M., Wiggins, A., van Hardenberg, P., & McGranaghan, M. (2019). Local-First Software: You Own Your Data, in Spite of the Cloud. In *Proceedings of Onward! 2019*. ACM.

Software Architecture

- Bass, L., Clements, P., & Kazman, R. (2012). *Software Architecture in Practice* (3rd ed.). Addison-Wesley.
- Shaw, M., & Garlan, D. (1996). *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall.

Human–Computer Interaction

- Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates.
- Norman, D. A. (2013). *The Design of Everyday Things* (Revised and expanded ed.). Basic Books.

Context-Aware Computing

- Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999). Towards a Better Understanding of Context and Context-Awareness. In *Handheld and Ubiquitous Computing (HUC '99)*. Springer.
- Dey, A. K. (2001). Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1), 4–7.

Provenance Models

- Moreau, L., & Groth, P. (2013). *Provenance: An Introduction to PROV*. Morgan & Claypool.
- W3C (2013). *PROV-DM: The PROV Data Model* (W3C Recommendation).

Personal Data Stores

- Mansour, E., Sambra, A. V., Hawke, S., Zereba, M., Capadisli, S., Ghanem, A., Aboulnaga, A., & Berners-Lee, T. (2016). A Demonstration of the Solid Platform for Social Web Applications. In *Proceedings of WWW '16 Companion*. ACM.

Multi-Agent Systems

- Wooldridge, M. (2009). *An Introduction to MultiAgent Systems* (2nd ed.). Wiley.