

POnto – An ontology for the Polkadot multichain ecosystem

Article

Draft version 1.0 – March 29th, 2023

Web3 Foundation Grants Program

Contributors

Rafael Brandão (rafael@mobr.ai)

Marcio Moreno (mmoreno@mobr.ai)

1. Research overview

The current proposal tackles the development of a conceptual framework for the Polkadot multi-chain ecosystem. Ultimately, this framework will be used to promote data integration, knowledge reasoning, and better communicability for the ecosystem community. It is a first step towards creating a rich and convenient asset for performing query searching and data analytics on Polkadot's ecosystem.

We conducted three main steps to fulfill the first milestone [3] of the current research grant. Namely, a broad literature review, a study on the blockchain ontology fundamentals, and the creation of a first version of **POnto**, an ontology describing the Polkadot ecosystem.

Our approach began by structuring a comprehensive systematic literature review to identify concepts and principles in published articles relating semantic web technologies and Web3, and more specifically ontologies and blockchains. Then, we conducted a research to assess existing applications and ontological frameworks developed for Web3, as well as checked if there was any research relating knowledge-oriented approaches to the Polkadot ecosystem at all. The outcome of this step is documented in a technical report [1].

In the next step of the proposed approach, we explored key concepts and principles identified in the state-of-the-art. We identified fundamentals of blockchain ontologies and the related work available. The outcome of this step is documented in a technical report [2].

Finally, for the concluding step of this milestone we created a conceptual framework incorporating and relating concepts and principles that govern Polkadot's ecosystem. This framework was designed to be flexible enough to accommodate the ongoing evolution of the technology. The next section presents the first draft of **POnto**, an ontology for the Polkadot multichain ecosystem.

Note that, the **POnto** specification and the delivered technical reports are currently in draft version, as expected. They will be changed and altered throughout the research process.

2. POnto – An ontology for the Polkadot multichain ecosystem (First Draft)

POnto is an ontology designed to represent and relate the main entities of the Polkadot ecosystem. The ontology name is a contraction of Polkadot Ontology, but it also has inspiration on the meaning of the word *ponto* in Portuguese, which can be translated to both *point* and *dot*.

The ontology is specified in the Turtle format, a textual syntax for RDF that allows an RDF graph to be completely written in a compact and natural text form, with abbreviations for common usage patterns and datatypes. Turtle also provides levels of compatibility with the N-Triples¹ format as well as the triple pattern syntax of the SPARQL² W3C Recommendation.

Three tools were used to conceptualize and specify the ontology. Visual Studio Code³ for the textual specification in the turtle format. Protégé⁴ to visualize and validate the ontology. All of the tools are free or open source.

POnto entities always have a label that describes the symbolic name of the entity and a comment that provides a brief description of its purpose and characteristics. Aiming at a better organization and support for evolution, POnto was designed in a modular fashion, using different modules that describe the various aspects of the Polkadot system. Currently there are eleven modules. The main module describes the basic concepts of system and distributed ledger architectures. The other modules group concepts for the Polkadot Ecosystem components: consensus, governance, ledger, oracle, stake, token, tooling, transaction, and wallet. There is also a module to state the ecosystem rules and constraints, called RBox, which is a common terminology used to name a container of rule descriptions. Each module is a separate ontology that is imported into the main module using the owl:imports property. The following sections detail these modules.

2.1 The Core module

The core module of the ontology starts with the usual prefixes to reuse owl⁵, rdf⁶ and rdfs⁷ specifications. In addition, it uses Dublin Core⁸ metadata elements to provide information about the ontology, like its authors, contributors, title, version, license, etc. Figure 1 illustrates these

¹ <https://www.w3.org/TR/turtle/#bib-N-TRIPLES>

² <http://www.w3.org/TR/sparql11-query/>

³ <https://code.visualstudio.com>

⁴ <https://protege.stanford.edu>

⁵ <http://www.w3.org/2002/07/owl>

⁶ <http://www.w3.org/1999/02/22-rdf-syntax-ns>

⁷ <http://www.w3.org/2000/01/rdf-schema#>

⁸ <http://purl.org/dc/elements/1.1/>

metadata and brings some metrics about the ontology, which is devised under the Apache License, Version 2.0⁹.



Annotations 		Metrics	
dc:title	POnto – An ontology for the Polkadot ecosystem	Axiom	394
dc:creator	MOBR Systems	Logical axiom count	154
dc:contributor	Marcio Moreno	Declaration axioms count	69
dc:contributor	Rafael Brandao	Class count	74
rdfs:comment [language: en]	This Polkadot ontology specification uses W3C RDF Schema and the Web Ontology Language. Its main source of information comes from the Polkadot Whitepaper written by Dr. Gavin Wood.	Object property count	35
rdfs:seeAlso	https://polkadot.network/whitepaper/	Data property count	3
rdfs:seeAlso	https://github.com/mobr-ai/POnto	Individual count	8
dc:license	 LICENSE-2.0	Annotation Property count	18
owl:versionInfo	0.1		
preferredNamespacePrefix	ponto		
term status	unstable		

Fig. 1 – POnto metadata and metrics overview.

The core model brings some basic concepts about system architectures, which will be used to improve the class hierarchy representation in the ontology. Specifically, the following entities are defined as the base:

- **Architecture:** A high-level structure of a software system that defines its components, their relationships, and their interactions.
- **Component:** A specific technology component that composes ledger architecture or implementation.
- **partOf:** A property that relates a component to the architecture it is a part of.
- **hasComponent:** A property that relates an architecture to the components it is composed of.

⁹ <https://www.apache.org/licenses/LICENSE-2.0>

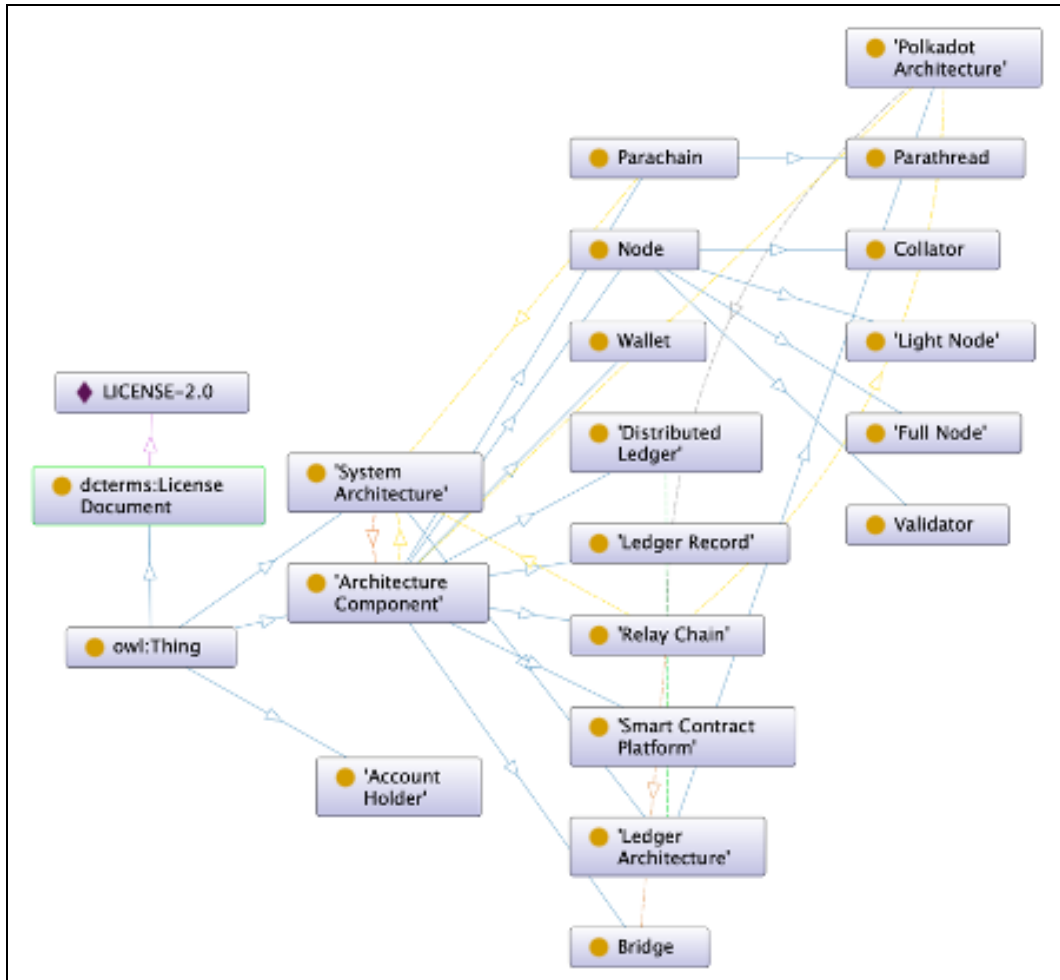


Fig. 2 – POnto Core module.

The core module (visual representation on Figure 2) states several classes and properties that represent fundamental concepts and relationships within the Polkadot ecosystem.

Polkadot Architecture is defined as a subclass of Ledger Architecture and is the central architecture of the Polkadot Ecosystem. It consists of several different types of chains, each with its own specific purpose and functionality. The Relay Chain is defined as the main chain in the Polkadot ecosystem that facilitates communication between parachains. Parachains are defined as independent blockchains that connect and rely on the Polkadot Relay Chain for security and interoperability purposes.

Bridges are mechanisms for connecting external blockchains to the Polkadot ecosystem and are defined as a subclass of Component and part of Parachain. Wallets are defined as software applications that store private keys and allow users to manage their cryptocurrencies. Nodes are components that participate in the Polkadot network, where Full Nodes are nodes that store a full copy of the blockchain. Light Nodes, on the other hand, only store a subset of the blockchain data and rely on other nodes for information.

Validators are nodes responsible for validating transactions and blocks and can operate as either Full Nodes or Light Nodes, depending on their resources and network requirements. Collators are nodes on a parachain that are responsible for collecting transactions and building new blocks. While they communicate with the Relay Chain, they are not on the Relay Chain themselves.

The core module also defines several relationships between the different classes. The "implements" property specifies that a Ledger Architecture implements an architecture. The "partOf" property specifies that a Parachain is part of the Polkadot Architecture, while the "hasParachain" property specifies that a Polkadot Architecture has a Parachain. The "usesBridge" property specifies that Relay Chains and Parachains may use a Bridge to connect external blockchains to the Polkadot ecosystem. Additionally, it defines a disjoint class that specifies that a Bridge cannot be part of both a Parachain and a Relay Chain at the same time.

2.2 Consensus module

The Consensus module (visual representation on Figure 3) describes classes and their relationships related to the consensus mechanisms used in the Polkadot ecosystem.

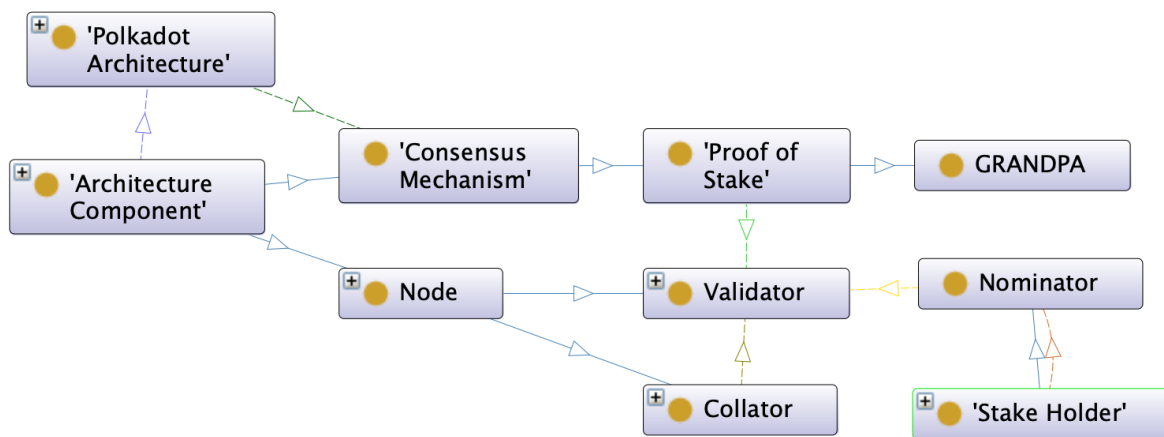


Fig. 3 – Simplified visual representation of Consensus module.

In this module, `ConsensusMechanism` is a subclass of `Component` and represents the mechanism by which nodes on a distributed ledger agree on a set of valid transactions and blocks. The class `ProofOfStake` is a subclass of `ConsensusMechanism` and represents a specific consensus mechanism in which validators are chosen based on the amount of stake they hold in the network. The `GRANDPA` entity is a subclass of `ProofOfStake` and represents the proof-of-stake consensus mechanism used in the Polkadot ecosystem.

The module also includes several relationships between the classes. The first relationship is `hasValidator`, which connects `Collator` and `Validator` classes, indicating that a collator has one or more validators. The second relationship is `hasNominator`, which connects `StakeHolder` and `Nominator` classes, indicating that a stakeholder has one or more nominators. The third relationship is `isConsensusMechanism`, which connects `PolkadotArchitecture` and `ConsensusMechanism` classes, indicating that the Polkadot architecture includes a consensus mechanism.

The module also includes several properties that describe the actions and roles of the different components in the consensus mechanism. The properties include `proposesBlock`, which connects `Validator` and `Transaction` classes and indicates that a validator node proposes a block in the Polkadot network. The property `validatesBlock` connects `Validator` and `Transaction` classes and indicates that a validator node validates a block in the Polkadot network. The property `validatesTransaction` connects `Validator` and `Transaction` classes and indicates that a validator node validates a transaction. The property `buildsBlock` connects `Collator` and `Transaction` classes and indicates that a collator node builds a block for a specific parachain. The property `collectsTransaction` connects `Collator` and `Transaction` classes and indicates that a collator node collects transactions for a specific parachain. Finally, the property `usesProofOfStake` connects `ProofOfStake` and `Validator` classes and indicates that a validator participates in the `ProofOfStake` consensus mechanism.

2.3 Governance module

The Governance module (visual representation on Figure 4) encloses four main classes to represent entities and their relationships related to the governance specified in the Polkadot ecosystem.

`GovernanceComponent` is a class to represent a technology component that is part of the governance mechanism in the Polkadot ecosystem. The `Treasury` class represents a governance component capable of creating a pool of funds used to support the development and maintenance of the Polkadot ecosystem. The `Governance` class is the governance component that represents the process in which decisions are made about the rules and policies that govern the Polkadot ecosystem. Another governance component in this module is the `Council` class, which represents a group of stakeholders responsible for making decisions about the direction of the ecosystem.

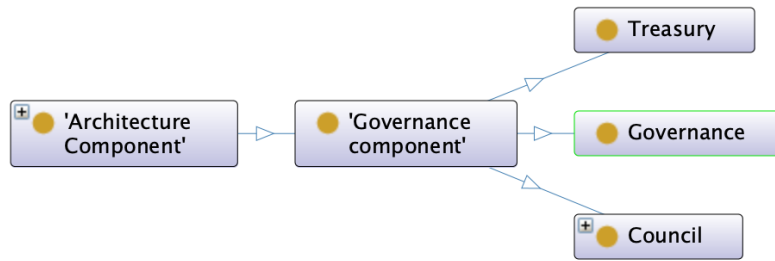


Fig. 4 – Simplified visual representation of Governance module.

The Governance module also defines some properties and relationships, including a relationship between the Governance and Council classes using the `hasGovernanceCouncil` property, indicating that a Governance component may have one or more Council components associated with it.

2.4 Ledger module

The ledger module goal is to define a set of distributed and decentralized ledgers that are part of or related to the Polkadot ecosystem, along with their relationships. Figure 5 presents a visual representation for this module.

The first part of this module defines the classes of different ledgers such as Polkadot, Kusama, Westend, Westmint, Rococo, Statemint, Statemine, Moonbeam, Astar, Acala, and Phala. Each one of these classes is a subclass of the `DistributedLedger` class.

The second part defines the relationships between the Polkadot ecosystem and the different ledgers. The "implements" relationship is defined as a property that has a domain of `PolkadotArchitecture` and a range of either `Polkadot` or `Kusama`. This relationship implies that the `PolkadotArchitecture` is implemented by either `Polkadot` or `Kusama`, indicating that they both are implementations of the same architecture.

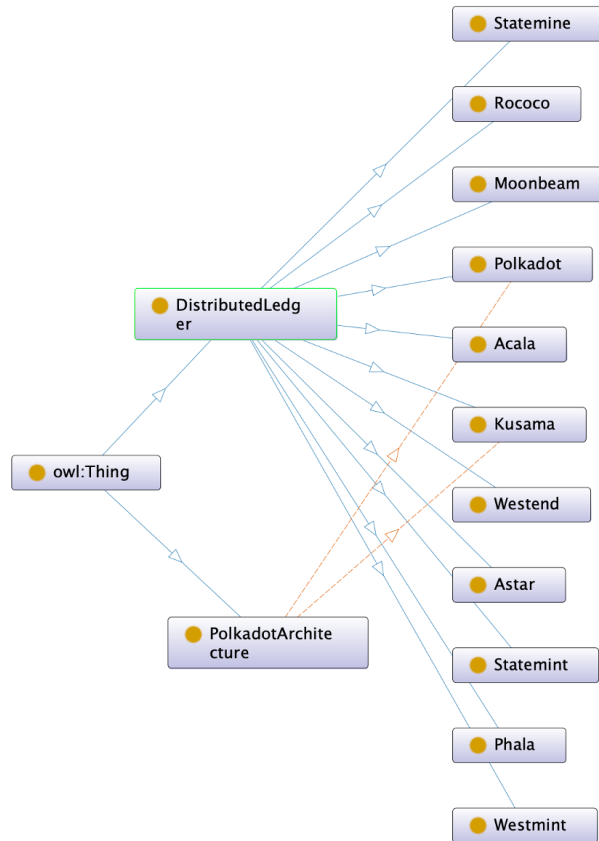


Fig. 5 – Simplified visual representation of Ledger module.

2.5 Oracle module

The Oracle module's main goal is to structure the description of oracle services in the Polkadot ecosystem. Figure 6 presents a visual representation for the module's main entities relationships.

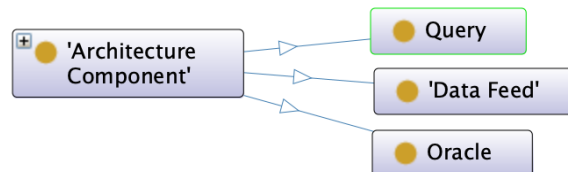


Fig. 6 – Simplified visual representation of Oracle module.

The module encompasses three main components of the oracle services in the Polkadot ecosystem: oracle, data feed, and query. This module could be populated automatically through a script so all data feed instances with smart contract information could be easily retrieved

through simple querying. For instance, to enable answering queries such as “What are the ChainLink data feeds available in the Polkadot Ecosystem?”

2.6 Stake module

The Stake module represents the staking entities in the Polkadot Ecosystem, which are key in both governance and consensus mechanisms. Figure 7 presents a visual representation of POnto’s Stake module.

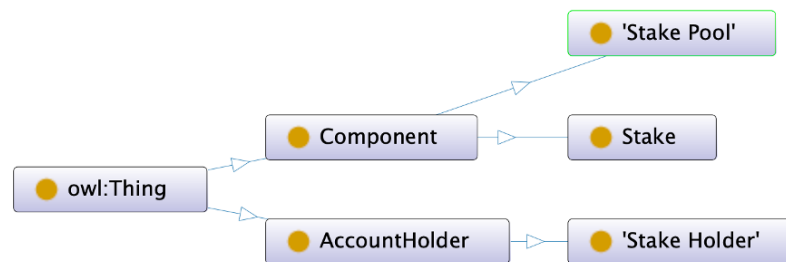


Fig. 7 – Simplified visual representation of Stake module.

The module has three main classes: Stake, StakeHolder, and StakePool. Stake represents a measure of the amount of tokens held by an account, used to determine the account's influence in the consensus and governance mechanisms. StakeHolder represents an account holder who holds a stake in the ecosystem and participates in the consensus and governance mechanisms. StakePool is a group of stakeholders who combine their stakes to increase their chances of being selected as validators.

The module also defines three properties: hasStake, belongsToPool, and hasStakeHolder. hasStake relates a StakeHolder to the amount of stake they hold in the Polkadot network. belongsToPool relates a StakeHolder to the StakePool they belong to, while hasStakeHolder relates a StakePool to the StakeHolders that belong to it.

2.7 Token module

The Token module, illustrated on Figure 8, provides a structured representation of the different types of tokens and accounts in the Polkadot ecosystem.

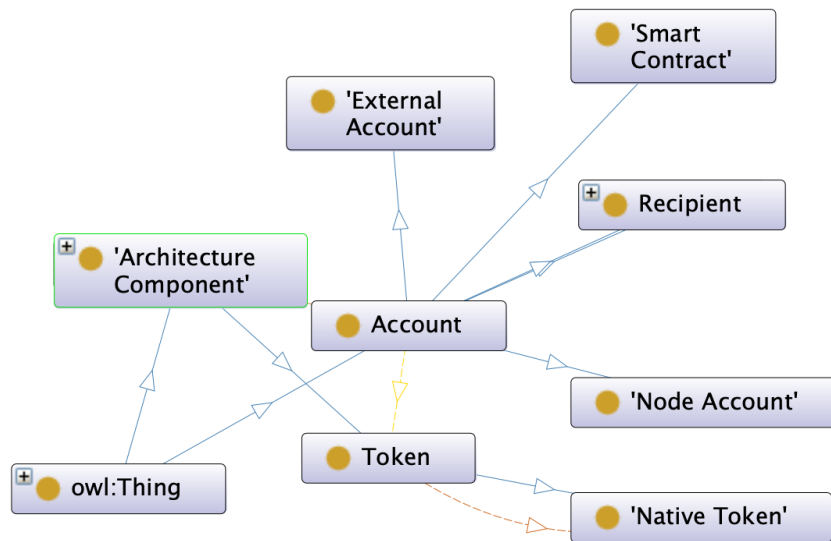


Fig. 8 – Simplified visual representation of Token module.

The Token module encompasses a Token class to represent assets on the Polkadot Ecosystem. It also defines the NativeToken class as a subclass of Token to represent the primary token used as the currency in a distributed ledger ecosystem, typically issued by the ledger protocol itself.

The module also contains several classes for different types of accounts, including Account, ExternalAccount, NodeAccount, and SmartContract. Account is a general class representing an address or record on the blockchain used to store and manage cryptocurrency holdings or other types of assets. ExternalAccount is an account controlled by an external system, such as an exchange or custodian. NodeAccount is an account used by a node to participate in block production or validation. SmartContract is a self-executing contract stored on the distributed ledger that can interact with other contracts or accounts.

The Token module defines several relationships between its classes. "hasAccount" relates a node to an account, indicating that a node has an associated account. "isAccountOf" is the inverse of "hasAccount" and indicates that an account belongs to a node. "hasToken" relates an account to a token, indicating that an account holds a certain amount of a specific token. "belongsTo" is the inverse of "hasToken" and indicates that a token is held by a specific account. "isNativeToken" relates a token to a NativeToken, indicating that a specific token is a NativeToken.

This module also could be populated automatically through a script so minted or available tokens in the Polkadot Ecosystem could be easily retrieved. For instance, to enable answering queries such as "Give me all non-fungible tokens minted on the Polkadot Ecosystem?"

2.8 Tooling module

The Tooling module's main goal is to define concepts about the toolings that are part of the Polkadot ecosystem. A simplistic visualization of the Tolling module is illustrated on Figure 9.

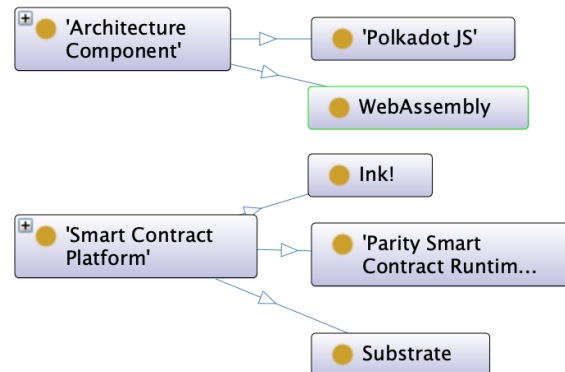


Fig. 9 – Simplified visual representation of Tooling module.

The module defines different smart contract platforms and tooling available in the Polkadot ecosystem. The Substrate class is a subclass of SmartContractPlatform and represents the Substrate Polkadot's blockchain development framework that provides a set of tools and libraries that developers can use to build customized blockchain networks and smart contract platforms. The Ink class is also a subclass of SmartContractPlatform and represents a high-level programming language designed specifically for writing smart contracts on the Polkadot network. PSCR is another subclass of SmartContractPlatform and represents a lightweight and modular runtime environment for executing smart contracts on the Polkadot network.

The module also describes the class Wasm as a low-level virtual machine that is used to run smart contracts on Polkadot, and the class PolkadotJS, which represents a collection of libraries and tools for building Polkadot applications in JavaScript.

2.9 Transaction module

The Transaction module (see Figure 10) provides a structured way to represent and reason about transactions in the Polkadot ecosystem.

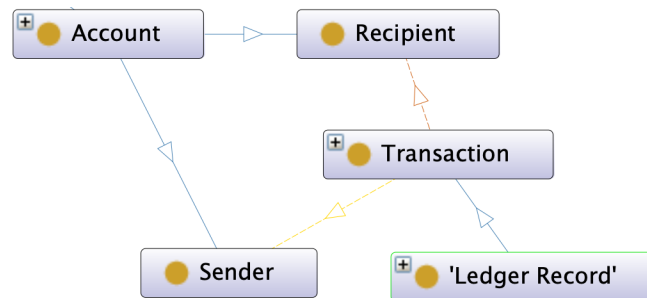


Fig. 10 – Simplified visual representation of Transaction module.

The module encompasses the Transaction class to represent a transaction in the Polkadot blockchain, which involves the transfer of assets between two parties. The Sender and Recipient classes are two subclasses of the Account class. The Sender class represents the entity sending assets in a transaction, while the Recipient class represents the entity receiving assets in a transaction.

The module also includes three datatype properties: `amountTransferred`, `typeOfAsset`, and `uniqueID`. The `amountTransferred` property is used to specify the amount of assets transferred in a transaction. The `typeOfAsset` property is used to specify the type of asset transferred in a Polkadot blockchain transaction. The `uniqueID` property is used to specify the unique identification number of a record in the Polkadot blockchain.

In addition, the module includes four object properties: `hasRecord`, `hasSender`, `hasRecipient`, and `belongsToBlock`. The `hasRecord` property is used to associate a record with a Polkadot blockchain transaction. The `hasSender` property is used to associate a sender with a Polkadot blockchain transaction. The `hasRecipient` property is used to associate a recipient with a Polkadot blockchain transaction. The `belongsToBlock` property is used to associate a transaction with a block in which it is recorded.

2.10 Wallet module

The Wallet module (see Figure 11) provides a structured way to represent and reason about wallets in the Polkadot ecosystem.

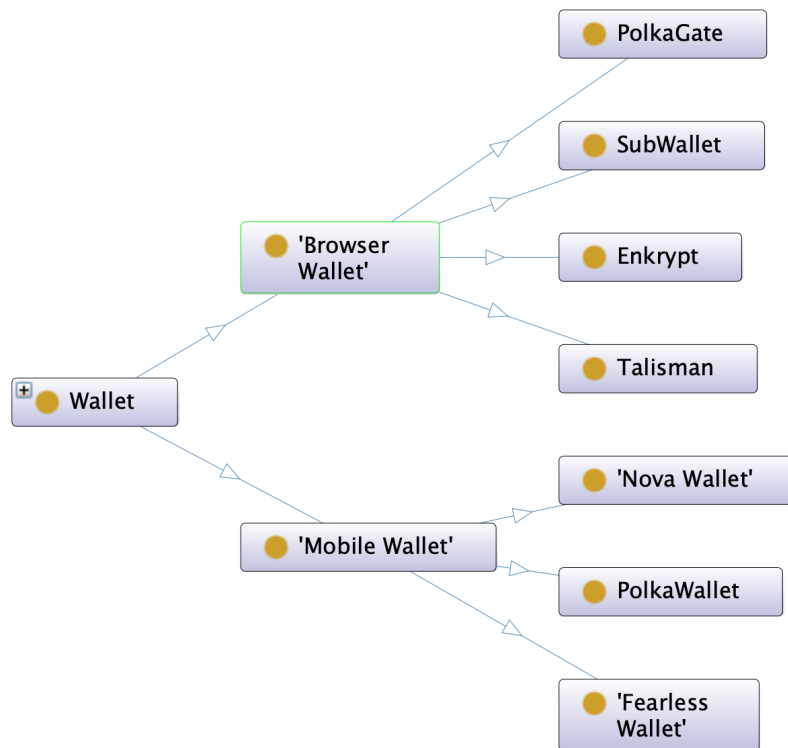


Fig. 11 – Simplified visual representation of Wallet module.

The Wallet module defines a BrowserWallet class to represent a wallet that runs in a web browser and can be accessed from any device, and a MobileWallet class, representing a wallet that runs on a mobile device. Both classes are defined as a subclass of Wallet.

The module also specifies four classes to represent different wallets from the Polkadot ecosystem: "Enkrypt", "PolkaGate", "SubWallet", and "Talisman". These wallets are all defined as subclasses of "BrowserWallet", indicating that they are a type of browser wallet. The MobileWallet class is also part of the Wallet module, Aiming at representing mobile wallets, which is a super class of three different wallets: "FearlessWallet", "NovaWallet", and "PolkaWallet".

Two relationships are also part of the module. The hasAccount relationship specifies that a wallet may have one or more accounts. This relationship can be used to link a wallet to its associated account(s). The "hasWallet" relationship specifies that an account holder may have one or more wallets, and can be used to link an account holder to their associated wallet(s).

2.11 RBox

The Rule Box specifies rules and constraints for the entities present in the POnto. These rules states the following:

1. A StakeHolder must have at least one valid stake in the Polkadot network.
2. A node in the Polkadot network must belong to at least one ledger.
3. A parachain can only be connected to one relay chain.
4. A validator node must have at least one nominator stakeholder in the Polkadot network.
5. A governance council in the Polkadot network must have at least one and at most 100 members.
6. A StakeHolder can only belong to one StakePool in the Polkadot network

3. Final remarks

The conducted systematic literature review [1] identified, evaluated, and synthesized evidence related to the general application of semantic web technologies in Web3. It also assessed the literature available referring to the Polkadot ecosystem in that matter. A revealing aspect of the conducted study was the low number of publications that specifically relate Polkadot with ontologies and semantic web technologies in general. In the conducted search, only three of the screened articles refer to Polkadot, none of them contribute to the theme. Supported by the outcome of the literature review, we assessed blockchain ontology fundamentals and related work, gathering and structuring the information in a separate technical report [2].

One point that was clear during the research activities for the current milestone [3] is that research solely based on free and open-source tooling and open-access databases still is challenging. For instance, while open-access databases provide researchers with access to a vast amount of data, not all relevant data may be available or accessible through these sources. In addition, open-source tooling and open-access databases often lack standardization, making it difficult to compare and integrate data from different sources. Let alone the fact researchers may require technical expertise to work with such tooling, which can be a barrier to entry for some.

The first draft version of **POnto** formalizes key concepts along with their relationships and provides a structured representation of the Polkadot Ecosystem. It enables reuse of domain knowledge and also its natural expansion. The goal is to use the ontology to support development of applications and tools to interact with the Polkadot Ecosystem more consistently. Moreover, it can be used as a framework for building more complex models and simulations of the Polkadot network. This can help researchers and developers better understand the behavior of the network and identify areas for improvement. The draft version discussed in this article is a step towards creating a standard and formal understanding of the Polkadot ecosystem. As the Polkadot network continues to evolve and grow, the community is encouraged to adapt and expand POnto to reflect new concepts and relationships.

The current research grant comprehends a first-step towards creating a rich and convenient asset for performing query searching and data analytics on the Polkadot's multi-chain environment. At this moment, we are not considering the development and deployment of any functional asset or prototype, but specifically the ontology representation. The next milestone [4] in our roadmap, will be a case study to explore the future application of POnto to support building a controlled natural language (CNL) and a query engine in Polkadot's multi-chain ecosystem.

Acknowledgement

This work was supported by a research grant from the Web3 Foundation and is publicly available at [5].

References

- [1] Technical Report – Systematic Literature Review. Available on github at <https://github.com/mobr-ai/POnto>
- [2] Technical Report – Blockchain ontology fundamentals and related work. Available on github at <https://github.com/mobr-ai/POnto>
- [3] Milestone 1 – Literature review and conceptual framework specification. Available on github at <https://github.com/w3f/Grants-Program/blob/master/applications/Knowledge-Oriented-Framework.md#milestone-1--literature-review-and-conceptual-framework-specification>
- [4] Milestone 2 – Case study for query engine. Available on github at <https://github.com/w3f/Grants-Program/blob/master/applications/Knowledge-Oriented-Framework.md#milestone-2--case-study-for-query-engine>
- [5] Final paper version to be available on arXiv