2

Architecture Overview Specification

The overall architecture defines all required components and subsystems, their basic functionality and behaviour, as well as their interfaces and interaction patterns in accordance with the user stories and the requirements specified in the project. The detailed specification of the i3-MARKET components and interfaces are reported in the chapters below.

In particular, the high-level architecture covers:

a) the i3-MARKET Backplane solutions with its core functionalities;
b) the interaction of existing data spaces and marketplaces with the i3-MARKET Backplane and each other (for secure data access) based on open interfaces;
c) the engagement of data providers, consumers, owners via smart wallets and applications, and the interactions with the i3-MARKET Backplane for the sake of privacy preservation and access control to their personal or industrial data assets.

2.1 Architecture

We describe the architecture in the 4 + 1 architectural view model. This is a standard model, commonly used for documenting software architectures.

The complete architecture is available on the i3-MARKET Wiki pages. It is available to all partners to be viewed or modified. The drawing tool used is either Gliffy or Draw.io.

2.1.1 The 4 + 1 architectural view

The *4 + 1 architectural view model* contains different views [1] as depicted in Figure 2.1.

The 4 + 1 architectural view model was adapted to fit our purposes. To make sure the different interfaces and the communication between our proposed system and the external systems are analysed, the so-called context view is added to the view model.

Table 2.1 describes the different views of the adapted 4 + 1 architectural view model.

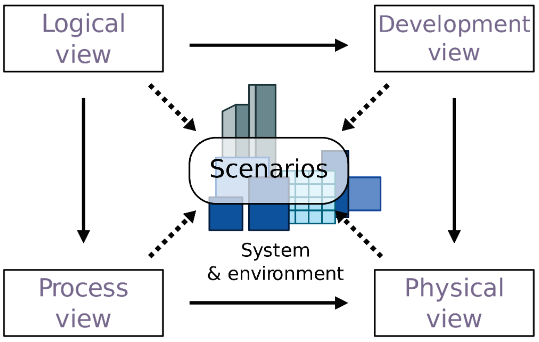


Figure 2.1 4 + 1 Architectural view model.

Table 2.1 Views of the adapted 4 + 1 architectural view model.

Architectural view	Description	Diagrams to use
Context view	• System as a blackbox • Interfaces and communication between blackbox and external systems	Context diagrams
Logical view	• Functionality that the system provides to end-users	Class and state diagrams
Process view	• Dynamic aspects of the system • System processes • Runtime behaviour of the system	Sequence, communication, and activity diagrams
Development/ implementation view	• System from a programmer's perspective • Software management	Component diagrams
Physical/deployment view	• System from a system engineer's point of view • Topology of software components on the physical layer (and their communication)	Deployment diagrams
Scenarios/use case view	• Sequence of interactions between objects and between processes • To identify architectural elements and to illustrate and validate the architecture design. • Starting point for tests	Use case diagrams

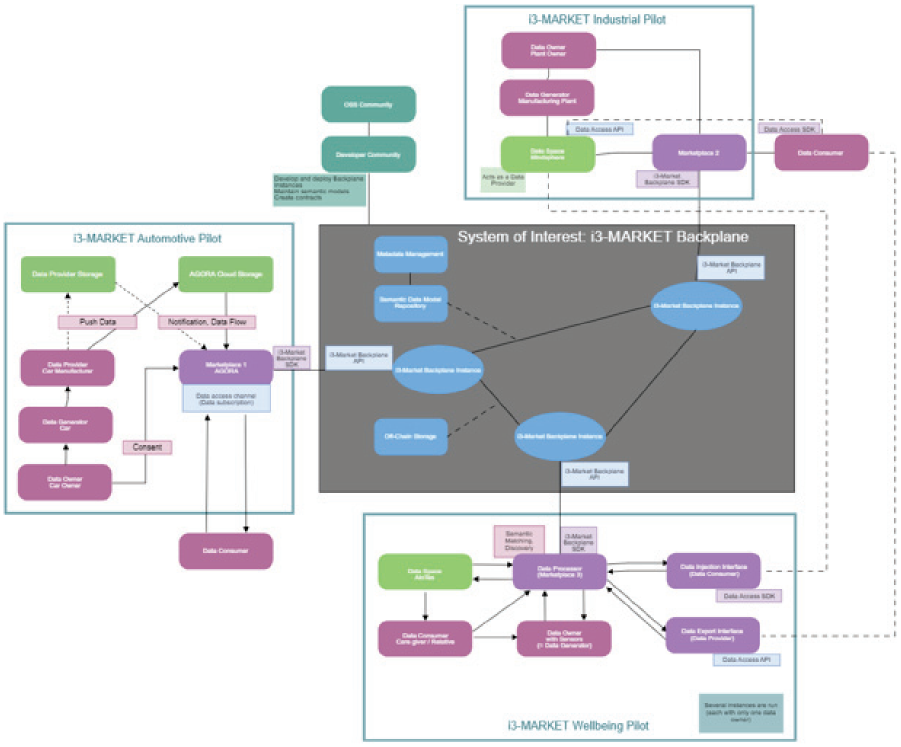
2.2 Context View

The context view shows a system as a whole and its interfaces to external factors. System context diagrams represent all external entities that may interact with a system; such a diagram pictures the system at the centre, with no details of its interior structure, surrounded by all its interacting systems, environments, and activities. The objective of the system context diagram is to focus attention on external factors and events that should be considered in developing a complete set of system requirements and constraints.

System context views are used early in a project to get agreement on the scope of the system. Context diagrams are typically included in a requirements document. These diagrams must be agreed on by all project stakeholders and thus should be written in plain language so that the stakeholders can understand items within the document.

The so-called system of interest, the i3-MARKET Backplane, is the centre of this diagram but is considered as a grey-box, showing only little internal details of the system.

The focus of this view is the external actors that have interfaces to systems. In this case, the external actors are the three pilots of the i3-MARKET project:



i3-MARKET Industrial Pilot
Data Owner Plant Owner
Data Generator Manufacturing Plant
Data Access API
Data Access SDK
Data Space Windsphere
Marketplace 2
Data Consumer
Acts as a Data Provider
i3-Market Backplane SDK
OSS Community
Developer Community
Develop and deploy Backplane instances
Maintain semantic models
Create contracts
i3-MARKET Automotive Pilot
Data Provider Storage
AGORA Cloud Storage
Push Data
Notification, Data Flow
Data Provider Car Manufacturer
Marketplace 1 AGORA
i3-Market Backplane SDK
Data access channel (Data subscription)
Data Generator Car
Consent
Data Owner Car Owner
Data Consumer
System of Interest: i3-MARKET Backplane
Metadata Management
Semantic Data Model Repository
i3-Market Backplane API
i3-Market Backplane Instance
i3-Market Backplane API
i3-Market Backplane Instance
Off-Chain Storage
i3-Market Backplane Instance
i3-Market Backplane API
Semantic Matching, Discovery
i3-Market Backplane SDK
Data Space Airflex
Data Processor (Marketplace 3)
Data Injection Interface (Data Consumer)
Data Access SDK
Data Consumer Care-giver / Relative
Data Owner with Sensors (= Data Generator)
Data Export Interface (Data Provider)
Data Access API
Several instances are run (each with only one data owner)
i3-MARKET Wellbeing Pilot

Figure 2.2 Context view with i3-MARKET as a blackbox.

- i3-MARKET automotive pilot
- i3-MARKET wellbeing pilot
- i3-MARKET industrial pilot

Figure 2.2 shows how data consumer and data provider exchange data via the data access API. The marketplace interacts with the i3-MARKET Backplane via an SDK. Both the API and the SDK were developed in the project.

The three pilot boxes also show some internal elements of the pilots. Each pilot has their own internal structure, but they share the same interface to the i3-MARKET Backplane. This enables seamless data exchange between all marketplaces.

2.3 Logical View

The logical view represented in Figure 2.3 shows the functionality that the system provides to end-users.

The objective of the logical view is twofold. On one hand, this view shows the i3-MARKET system (green box) and the link between the stakeholders and the marketplaces. On the other hand, the logical view pursues showing the internal decomposition of i3-MARKET system into the logical subsystems and components, which implement the i3-MARKET Backplane API and secure data access API (SDA API).

In general terms, i3-MARKET supports actors with the i3-MARKET Backplane functionality by means of the two following main entry points:

- The *Backplane API* and *SDA API* (depicted as green lines in the picture), or in other words, the direct access to the i3-MARKET Backplane. These two APIs enable access to all integrated building blocks. This is the use case of these actors which follow a more *ad-hoc* integration with i3-MARKET.
- The *i3-MARKET SDK (i3M SDK)* (depicted as pink boxes in the picture), to support the end-users' developers with the integration of Backplane API and SDA API. This product is intended for these actors that pursue a more "assisted" support.

Regarding the link with the stakeholder and marketplaces, in the case of the data marketplace actors, i3-MARKET assists them with a full version of the Backplane API and the i3M SDK (Backplane module), which gives support for interacting with the Backplane API.

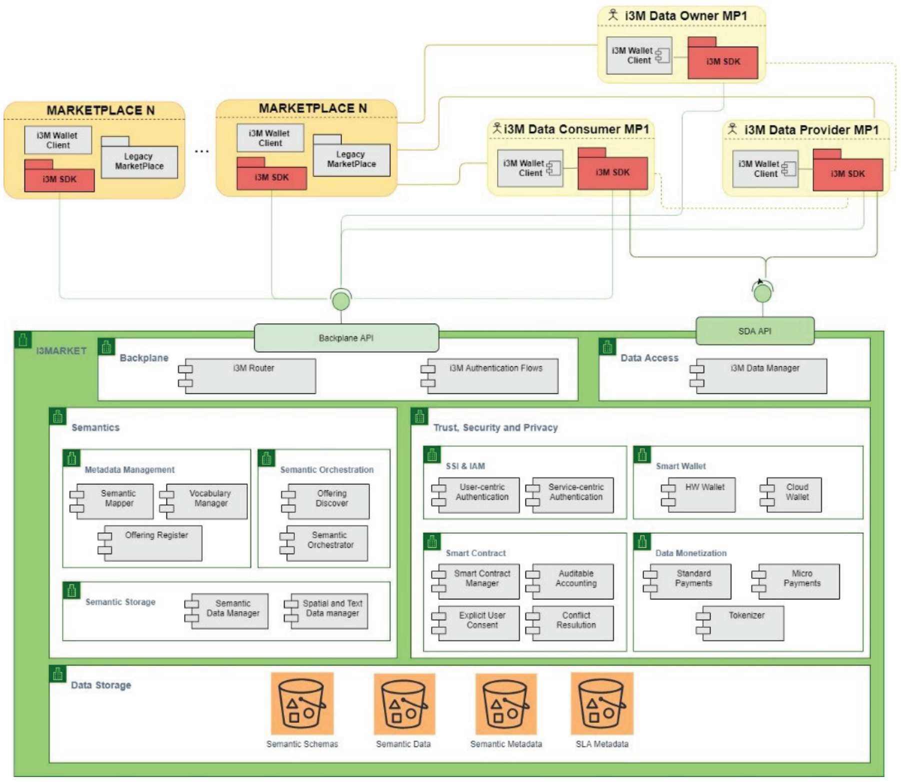


Figure 2.3 Logical view with i3-MARKET.

In the case of the data owners, data providers, and data consumers, the normal operating mode is the access to i3-MARKET Backplane through their own data marketplace. However, for some particular data marketplace cases, data owners, data providers, and data consumers will have the possibility to directly interface with i3-MARKET system through the available SDKs and APIs. More specifically, i3-MARKET will allow direct communication with the stakeholder by means of the following components:

- Data owner, through the i3M SDK (Backplane module), which gives support for interacting with the Backplane API (light green lines in the picture).
- Data provider, through the i3M SDK-Backplane module which gives support for interacting with the Backplane API (light green lines in the picture) and the i3M SDK-secure data access API, which gives support

for interacting with the secure data access API (dark green lines in the picture).
- Data consumer, through the i3M SDK-Backplane module, which gives support for interacting with the Backplane API and the i3M SDK-secure data access API, which gives support for interacting with the secure data access API.

In order to guarantee the authentication mechanisms proposed by i3-MARKET, a Wallet Client should be installed into the end-user side in order to store the user private keys.

2.4 i3-MARKET Microservice-based Architecture

i3-MARKET Backplane is mostly a set of semi-independent subsystems with self-contained functionalities such as the identity and access management system, the semantic engine subsystem, data access subsystem, etc. Most of these subsystems have broken down their functionalities into atomic and loosely coupled sub-components exposing their functionality through a REST API, which yields a microservice-based nature to the i3-MARKET system.

This microservice-based architecture brings i3-MARKET a set of very well-known benefits such as:

- facilitating the communication between the components in a system;
- have been independently developed and deployed into a more efficient management;
- facilitating the identification of dependencies between the components;
- modular architecture allows each application to use only those functionalities that are needed;
- helps to manage the complexity of the overall system.

Figure 2.4 shows a detailed landscape of the current set of microservices (cubes), APIs (little yellow rectangles), components (blue rectangles), and storages (white rectangles) on i3-MARKET. Each arrow in the picture denotes a dependency between the subject and object involved in the arrow. Finally, for linking each of the service/microservices/library depicted in the diagram with the component diagrams in section development view "development view", we have categorized each service/microservice/library according to the system (green dashed boxes) and subsystems (brown dashed boxes) they belong. Finally, remark that the RPC distributed ledger is one and single instance, but it has been put as several instances for picture legibility.

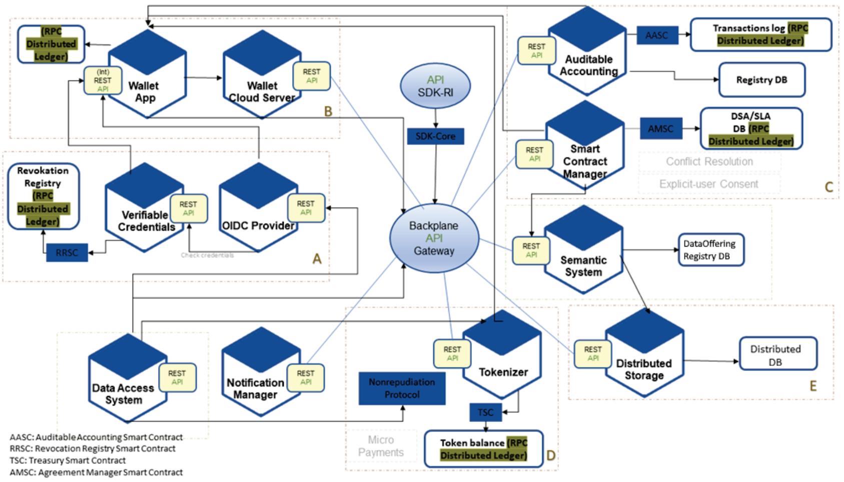


Figure 2.4 i3-MARKET microservice layout.

Figure 2.5 shows the identified dependencies between i3-MARKET components:

- SDK system: For a more grounded view of this subsystem, refer to Chapter 17 on i3-MARKET SDK and marketplace reference implementation.
  - *SDK-core* libraries for making easier the development of applications that make use of the Backplane API. It interfaces with the Backplane gateway.
  - *SDK-RI* future common pilots-driven complex workflows based on the Backplane services. It interfaces with the SDK-core library.
- Trust, security, and privacy system:
  - SSI and IAM subsystem (label A). For a more grounded picture of this subsystem, see “SSI and IAM subsystem” in Chapter 3 on “i3-MARKET identity and access management”.
    - “User-centric authentication” component, responsible of providing the management of self-sovereign identity based on DID and VC and the compatibility with OIDC standard. Microservices:
      - *Verifiable Credential* microservice, which provides DID, Verifiable Credential management, and compatibility

with OIDC standard. It interfaces with the wallet because the Verifiable Credential assumes that the user created and controls with his crypto wallet their identities and with the RPC ledger storage for updating the revocation of credential.

- "Service-centric authentication" component, responsible for providing authentication and authorization of users and client with standard OIDC/OAuth flows, integrating the user-centric authentication component. Microservices:
    - *OIDC provider* microservice, which implements the OIDC compatibility (based on Verifiable Credential). It interfaces with the Verifiable Credential for allowing the token creation based on the Verifiable Credentials and with the wallet for sending the credentials.

○ Smart Wallet subsystem (label B). For a more grounded picture of this subsystem, see "Smart Wallet subsystem" in Chapters 4 and 5.

- *Wallet APP* for storing user private keys. It interfaces on the RPC ledger storage.

○ Smart contract subsystem (label C). For a more grounded picture of this subsystem, see "smart contract subsystem" in Chapter 4 and Chapter 7 in Book Series Part II.

- *Smart contract manager* component/microservice responsible for providing a gateway to access the smart contracts. It was conceived mainly for managing the SLA and DSA smart contract (business smart contracts), and the extension of its purpose for other smart contracts is still under discussion. It interfaces with the RPC ledger storage for storing the data sharing agreement object and the semantic engine for creating data purchase.
- *Auditable accounting* component/microservice component responsible for logging and auditing interactions between components and recording the registries in the blockchain. It interfaces with the RPC ledger storage for registering auditable data and in the future might be connected with the distributed storage for storing proofs.

    ○ Data monetization subsystem (label D). For a more grounded picture of this subsystem, see "data monetization subsystem" in Text on "i3-MARKET Crypto token and data monetization".
    ○ *Non-repudiation protocol library*. It interfaces with the Backplane API for interacting with auditable accounting.

- Semantic system:
    ○ *Semantic system* service responsible for managing the offerings/discovery and semantic data model in the i3-MARKET. It interfaces with contract manager managing contractual parameters, and it depends on the registry DB store and the distributed DB service's API. It might interact with ledger for Verifiable Credentials and DID IDs. And it interacts with the notification manager service for reporting new data offerings. For a more grounded picture of this component, see Chapter 4 "i3-MARKET Semantic Models" and Chapter 9 in Book Series Part II "i3-MARKET Semantic Model Repository and Community."
- Data access system:
    ○ *Data access* service in charge of providing the means for allowing the transfer of data between data provider and the data consumer. It interacts with the non-repudiation protocol library and Backplane (API) for enforcing smart contract. For a more grounded picture of this system, see Chapter 6 on data access & transfer – system.
- Storage system:
    ○ Distributed storage subsystem (label D):
        1. *Distributed storage* component/microservice responsible of storing i3-MARKET offerings index. Other uses of the distributed storage are still under discussion. Its interaction with the RPC ledger storage for storing proof for reliability of data is still under discussion.
- *Backplane gateway* component responsible for providing a gateway for all the internal services conforming to the Backplane. This gateway is the single-entry point for all clients. For more details about this component, see Chapter 3 in Book Series Part III Backplane API gateway. It depends on the availability of all the services masked behind it.

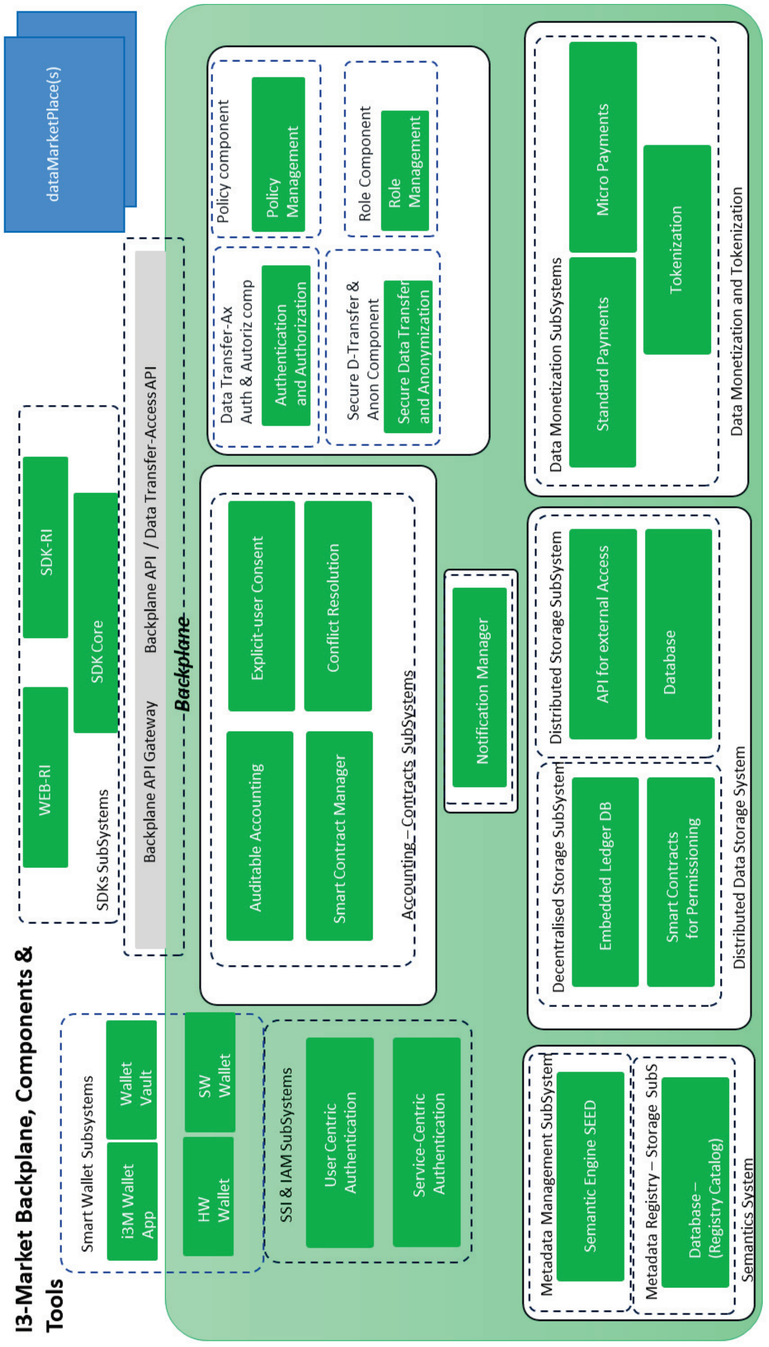

I3-Market Backplane, Components & Tools
dataMarketPlace(s)
SDKs SubSystems
WEB-RI
SDK-RI
SDK Core
Backplane API Gateway
Backplane API / Data Transfer-Access API
Backplane
Smart Wallet Subsystems
i3M Wallet App
Wallet Vault
HW Wallet
SW Wallet
SSI & IAM SubSystems
User Centric Authentication
Service-Centric Authentication
Accounting – Contracts SubSystems
Auditable Accounting
Smart Contract Manager
Explicit-user Consent
Conflict Resolution
Notification Manager
Data Transfer-Ax Auth & Autoriz comp
Authentication and Authorization
Policy component
Policy Management
Secure D-Transfer & Anon Component
Secure Data Transfer and Anonymization
Role Component
Role Management
Metadata Management SubSystem
Semantic Engine SEED
Metadata Registry – Storage SubS
Database – (Registry Catalog)
Semantics System
Decentralised Storage SubSystem
Embedded Ledger DB
Smart Contracts for Permissioning
Distributed Storage SubSystem
API for external Access
Database
Distributed Data Storage System
Data Monetization SubSystems
Standard Payments
Micro Payments
Tokenization
Data Monetization and Tokenization

Figure 2.5 High-level view of the i3-MARKET Backplane architecture and blocks for systems and artefacts.

2.5 i3-MARKET Core Functionality

i3-MARKET provides a set of core components in charge of providing the following capabilities:

1. **Authentication-identity/authorization-access:** i3-MARKET should allow users to authenticate themselves and get authorization to access the blockchain, secure and centralized data storage, and secured services on i3-MARKET.
2. **User management:** i3-MARKET should allow data providers and consumers to register, update, or delete from the system. In particular, i3-MARKET assists the data marketplaces and stakeholders with following functionalities: identity creation, user registration, identity update, and user deletion (which ensures that his identity on the blockchain cannot be mapped to his real identity anymore. $\rightarrow$ automatically consent termination/cancellation).
3. **Offering registration:** i3-MARKET must provide mechanisms for data providers to publish their datasets on i3-MARKET. i3-MARKET provides semantic data models to describe data offerings and data subscriptions/demands.
4. **Offering discovery:** i3-MARKET must provide mechanisms to allow data consumers to perform data queries based on the provided semantic models. It will have two variants: discover and retrieve locally or discover and retrieve in federated i3-MARKET data marketplaces network.
5. **Data subscription:** i3-MARKET must provide mechanisms to allow data consumers to express the intention of buying data and to request an SLA/SLS between the data consumer and a data provider, after a match was encountered.
6. **Consent:** i3-MARKET has to provide the mechanisms in order for the data owner to consent access for enabling the trading of data assets across domains and stakeholder boundaries, without the need for developers of an application (data consumer) to learn about the meaning of the data from the data provider or through manual analysis or experimentation with the data.
   a. Explicit consent: i3-MARKET must provide mechanisms to allow the data owner to give his consent before his data is transferred (data owner). When a user is deleted, all the data and metadata related to the user should be removed from any platform.

  b. Consent termination/cancellation: i3-MARKET must provide the mechanisms for ending the commercialization between involved parties on a smart contract. i3-MARKET provides the mechanisms to end running smart contracts at any time.

7. **Contracting:** i3-MARKET has to provide mechanisms that allow to complete data sharing agreements (SLA/SLS) between the data provider and the data consumer. The smart contracts are then generated from the SLA/SLS, which the participants agreed upon (data provider and data consumer).
8. **Data access:** i3-MARKET provides a data access API enabling an authorization of the data provider and the data consumer to allow a secure data transfer (peer-to-peer or i3-MARKET-channel subscription). The data access API provides a mechanism to monitor the data transfer and is tightly coupled with the signed smart contracts. This functionality was broken down into the following modules:

   i. Authentication and authorization
   ii. Data transfer transparency
   iii. Data management
   iv. Secure data transfer and anonymization

   On the other hand, i3-MARKET supports the following types of data transfer:

   a. On-demand → Data stream (see common vocabulary below).
   b. Subscription → Data batch transfer (see common vocabulary below).

9. **Data monetization/payment:** i3-MARKET provides functionality for data monetization, which aligns based on the pricing model defined in the offering description and amount consumed.

These capabilities have been validated in the i3-MARKET basic workflows (described in the following section). In concrete:

- "User management": For all end-users (data providers and data consumers), an identity should be created in advance for getting authentication-identity and authorization-access. Therefore, a user management activity will take place as pre-requisite for starting any interaction with any i3-MARKET instance.

- "Authentication-identity/authorization-access" is used as starting point for initiating any connection with i3-MARKET instances. Therefore, the process of authentication (and authorization) can be reflected at the beginning of most of the workflows. These are:
    - "Registering a new offering": The workflow starts with the authentication of the data provider as described in the diagram "authentication with end-user interaction" in Chapter 4 "i3-MARKET Semantic Models".
    - "Purchase data", "create and manage search alerts", and "transfer operational data": The workflow starts with the authentication of the data provider and data consumer, as described in the diagram "authentication with end-user interaction" in Chapter 4 "i3-MARKET Semantic Models".
- "Offering registration", the behaviour of the offering registration capability is directly shown in the "register a new offering" workflow.
- "Offering discovery", "data subscription", and "contracting" capabilities take place in the "purchase data" workflow.
- "Data access" and "data monetization" are the most significant steps in the "transfer operational data" workflow.

2.6 i3-MARKET Basic Workflows

i3-MARKET includes the implementation of three pilots to validate the functionalities of the i3-MARKET network. Even though every pilot has its own way of how it works and its specific requirements, there are some basic workflows, which apply to all of them. Those workflows are described for the five most important scenarios.

The scenarios are:

- Generate data.
- Register new data offerings.
- Search, discover, and retrieve data offerings in local and federated registry catalogues.
- Purchase data.
- Manage notifications.
- Access and transfer operational data.

The scenarios are described in detail in i3-MARKET deliverables about "Use Cases, Requirements and Overall i3-MARKET Architecture Specifications" available at https://www.i3-market.eu/research-and-technology-library/.

For each of the scenarios (which are part of the problem space), technical workflows have been derived. These workflows represent the technical realization in the solution space. They represent the dynamic behaviour of the system when stakeholders interact with it.

2.7 Process View

According to [2], the process view "takes into account some non-functional requirements, such as performance and availability. It addresses issues of concurrency and distribution, of system's integrity, of fault-tolerance, and how the main abstractions from the logical view fit within the process architecture—on which thread of control is an operation for an object actually executed".

Following this approach, the i3-MARKET process view should show the interaction between the process and threads of the system representing, among others, non-functional characteristics, concurrency, synchronization, availability, or performance.

From a diagram point of view, Booch stated "… the static and dynamic aspects of this view are captured in the same kinds of diagrams as for the design view – i.e. class diagrams, interaction diagrams, activity diagrams and statechart diagrams, but with a focus on the active classes that represent these threads and processes" [3].

A fine-grained detail is demanded for identifying the active objects (process and threads), which are instances of active classes, and the way they communicate between each other (synchronous/asynchronous) which is needed for this view. Due to that, the i3-MARKET process view definition was accomplished between the different technical task implementations.

2.8 Development View

According to [2], the development view "focuses on the actual software module organization…The software is packaged in small chunks—program libraries, or subsystems—that can be developed by one or a small number of developers. The subsystems are organized in a hierarchy of layers, each layer providing a narrow and well-defined interface to the layers above it".

Therefore, the objective of the development view is twofold:

- Giving a system view from a programmer's perspective, which might help in the development process.
- Supporting the software management by monitoring the accomplishment of subsystems and components depicted in the diagrams.

For the process of defining the development view, i3-MARKET has followed the guidelines proposed by arc42 template [4] for architecture construction and documentation, which is summarized in the following section.

2.8.1 Approach

The i3-MARKET "development view" is documented following the "building block view" as depicted in Figure 2.6 from arc42 template. Following are the cited instructions provided by the template:

- **Content:**

The building block view shows the static decomposition of the system into building blocks (modules, components, subsystems, classes, interfaces, packages, libraries, frameworks, layers, partitions, tiers, functions, macros, operations, data structures, etc.) as well as their dependencies (relationships, associations, etc.).
This view is mandatory for every architecture documentation. In analogy to a house, this is the floor plan.

- **Motivation:**

Maintain an overview of your source code by making its structure understandable through abstraction.

This allows you to communicate with your stakeholder on an abstract level without disclosing implementation details.

- **Form:**

The building block view is a hierarchical collection of black boxes and white boxes (see Figure 2.6) and their descriptions.

***Level 1** is the white box description of the overall system together with black box descriptions of all contained building blocks.*

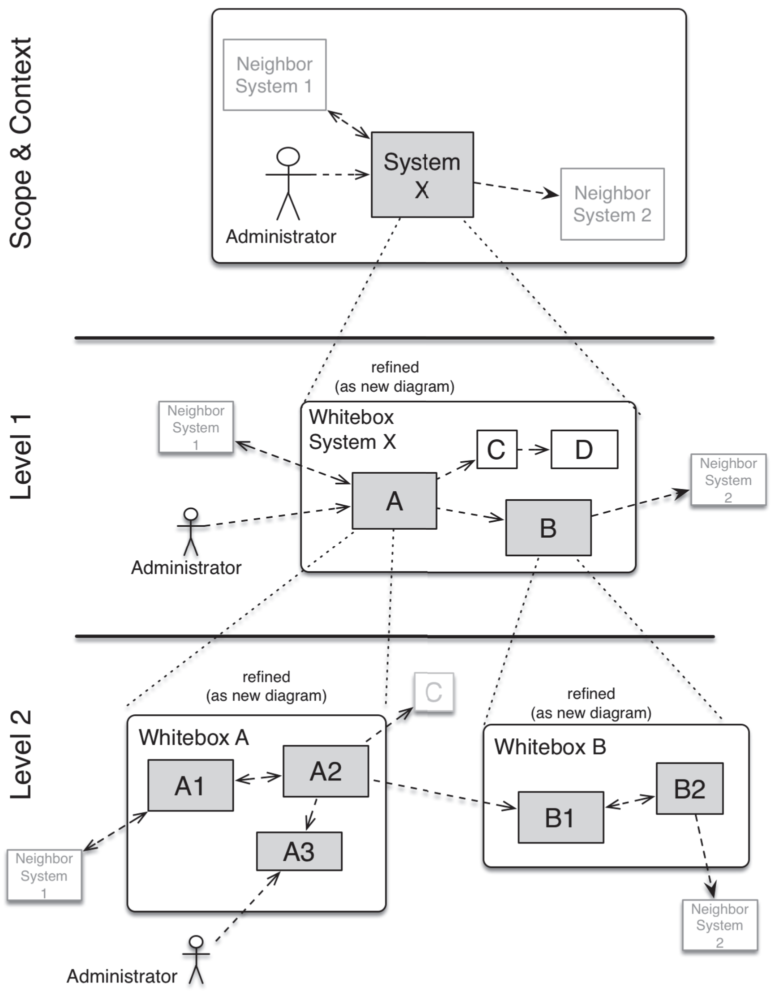


Figure 2.6 Building block hierarchy [3].

Level 2 *zooms into some building blocks of level 1. Thus, it contains the white box description of selected building blocks of level 1, together with black box descriptions of their internal building blocks.*

Level 3 *zooms into selected building blocks of level 2, and so on.*

In i3-MARKET, we have the following arc42-based templates for the documentation of the Level 1 development view.

Here you describe the decomposition of the overall system using the following white box template. It contains:

- an overview diagram;
- a motivation for the decomposition;
- black box descriptions of the contained building blocks (use a list of black box descriptions of the building blocks according to the black box template (see below). Depending on your choice of tool, this list could be sub-chapters (in text files), sub-pages (in a Wiki), or nested elements (in a modelling tool)".