CHANGE2TWIN

MARKETPLACE DESIGN AND METHODOLOGIES

Deliverable D2.2

CIRCULATION
Public

VERSION
1.0

DATE
May 31, 2021

AUTHORS
Robert Woitsch (BOC)
Wilfrid Utz (BOC)
Roman Dolhai (CB)
Michał Kulczewski (PSNC)
Paweł Gruchalski (PSNC)

LEAD PARTNER
BOC

CONTRIBUTING PARTNERS
TNO, UNIBO, PSNC (Adam Olszewski, Jan Węglarz, Maciej Stroiński)

QUALITY CONTROLLERS
Michal Kulczewski (PSNC),
Armir Bujari (UNIBO)
Jeroen Broekhuijsen (TNO)

HORIZON 2020

Consisting of

SINTEF	SINTEF AS
TTTECH-IND	TTTECH INDUSTRIAL AUTOMATION AG
Jotne	JOTNE EPM TECHNOLOGY AS
FBA	FUNDINGBOX ACCELERATOR SP ZOO
TNO	NEDERLANDSE ORGANISATIE VOOR TOEGEPAST NATUURWETENSCHAPPELIJK ONDERZOEK TNO
BOC	BOC ASSET MANAGEMENT GMBH
UNIBO	ALMA MATER STUDIORUM - UNIVERSITA DI BOLOGNA
CLOUDBROKER	CLOUDBROKER GMBH
IR	ASSOCIATION IMAGES & RESEAUX
PSNC	INSTYTUT CHEMII BIOORGANICZNEJ POLSKIEJ AKADEMII NAUK
SPS	SPACE STRUCTURES GMBH
CORDIS	CORDIS AUTOMATION B.V
Unit040	UNIT040 ONTWERP BV
Author-e	AUTHOR-E BV
Additive	ADDITIVE INDUSTRIES BV
CT INGENIEROS	CT INGENIEROS AERONAUTICOS DE AUTOMOCION E INDUSTRIALES SL
AETNA GROUP	AETNA GROUP SPA
Graphenstone	INDUSTRIA ESPANOLA PARA EL DESARROLLO E INVESTIGACION 2100

Document History

Version	Issue Date	Stage	Content and Changes
1.0	31.05.2021	Final version	

EXECUTIVE SUMMARY

This deliverable is the second deliverable of WP2 and aims to define the design methodology for the C2T marketplace. The approach embodies the following aspects:

- **designed** using a common knowledge representation format to support various interaction streams on business, technical and conceptual level,
- **configuring** the related components in the ecosystem of marketplaces (landing pages for user interaction, middleware for item enactment, infrastructure/platform for operation),
- **embedding** domain-specific knowledge in the use and application of items from the marketplace, and
- **sustainable** beyond the project runtime as the marketplace model itself is considered a blueprint for offerings provided individually by DIHs.

To this end, we first introduce the idea of a model-based design methodology from a conceptual viewpoint. This means, that our understanding of a marketplace is in the form of a *knowledge representation* that enables various types of user interaction and processing techniques. This knowledge representation is provided in the form of a modelling method (in accordance with the definition in [1] and applied in [2]). The modelling method is domain-specific (in a sense that the constructs are applicable for the design of marketplaces) and context sensitive (in a sense of content description).

This deliverable specifies the modelling method from a conceptual viewpoint and materialized in the form of a platform-independent and platform-specific implementation, resulting in a modelling tool prototype demonstrating the capabilities of the design methodologies for marketplaces.

As a result, this deliverable introduces the modelling method (metamodel and model processing functionalities) and the resulting prototype to design marketplaces and integrate those with operational systems (middleware and infrastructure/platforms) as positioned initially in D2.1 of the project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES AND CODE FRAGMENTS

LIST OF ABBREVIATIONS

Abbreviation	Definition
SuS	System under Study
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
BPaaS	Business Process as a Service
XaaS	Anything as a Service
DIH	Digital Innovation Hub
TRL	Technology Readiness Level
RTO	Research and Technology Organization
SP	Service Provider
MULO	Make – Use – License – Own
REST	Representational State Transfer
RDF	Resource Description Framework
OLIVE	OMILAB Integrated Virtual Environment
DAaaS	Data Analytics as a Service
C2T	Change2Twin Project

1 INTRODUCTION

In this introductory chapter, the project context of the work performed is introduced and the approach applied is discussed. The deliverable builds on the insights gained from the realisation of the initial deliverable in the workpackage and addresses challenges derived from this work and related workpackages.

Project Context

This deliverable "**Marketplace Design and Methodologies**" introduces the model-based approach to:

a) design the marketplace and to
b) represent knowledge that is necessary for offerings, operating and improving the marketplace.

The concept of the "**marketplace**" is understood as "an ecosystem that enables the provisioning of offerings – potentially in a form – from providers to consumers within a given framework". In our case, the offering must address digital twin technologies, however we would like to be flexible and not posed any apriori constraints to the technological readiness of items provided via the marketplace.

The concept of "**marketplace design**" is understood as the application of a model-based approach that observes and describes the so-called "system under study (SuS)" – in our case the marketplace for "digital twin technologies" with the goal to abstract the marketplace itself and items offered using a conceptual representation in order to apply algorithms on that model. We expect algorithms to search, filter, compare, map or translate the content of the marketplace as the most prominent that are visible for the end users. In addition, we consider algorithms to select the correct deployment package, install a cross-layer monitoring environment or to automatically process a release workflow of an offering that are mainly visible to so-called brokers of the marketplaces. Ideally, we address Digital Innovation Hubs (DIH) to become brokers of their own marketplaces.

The concept of the "**design methodologies**" is understood as a high-level and hence generic applicable methodology using a systematic approach to identify and describe offerings and apply aforementioned features on those description to offer added value to the user – such as the end users or the broker. Model-based approaches are commonly applied to implement (a) the modelling language, which is specified in form of a meta-model, (b) the algorithms, which can be applied on the models as well as (c) the modelling procedure, which is the intended use (creation, analysis, transformation, etc.) process of the model.

Therefore, this deliverable applies a **model-based approach** to describe the **marketplace** in order to enable the **description of offerings**, the integration of other marketplaces as an integration aspect, transformation and mappings of offerings, **provide algorithms** to filter, search, assess or map offerings to users' requirements as well as provide feature to **configure and monitor** the usage and operation of the marketplace.

This is performed in a collaborative manner, with other workpackages as introduced in Figure 1.

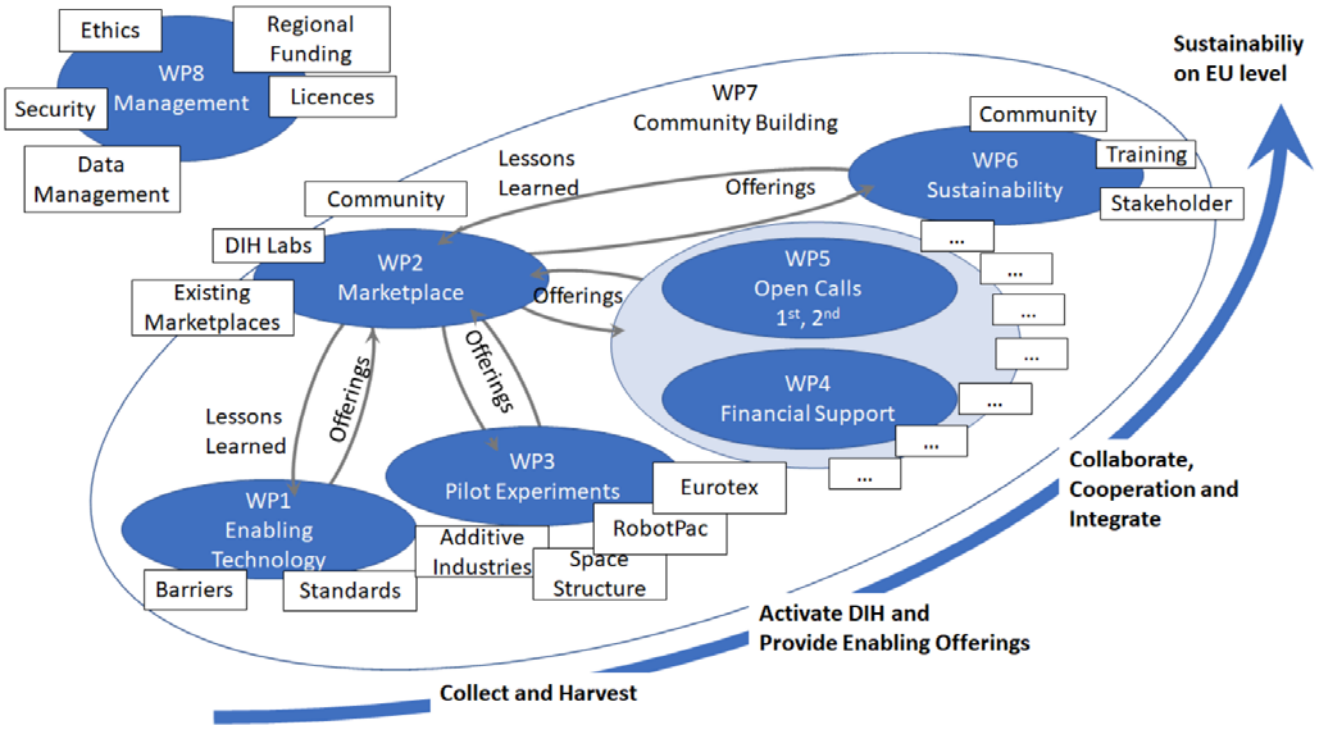


FIGURE 1 CHANGE2TWIN WORKPACKAGE DEPENDENCIES

Workpackage 2 and the corresponding task 2.1 of creating the marketplace model acts as a moderator between the other tasks of WP2 as well as tasks in WP1, WP3 and WP6. After new pilots have been started in WP4 and WP5, we also expect feedback from those results and achievements.

WP1 contributes with the common understanding and a taxonomy of digital twin offerings, which is understood as requirements to establish a marketplace model that enables this description.

WP3 contributes with concrete applications and solutions that have been selected for the particular needs of the pilot cases. Therefore, we understand this as the requirements from the users and broker who either uses the marketplace model to find such solutions for the particular needs, as well as for the broker to offer solutions, or to operate a marketplace. Hence, the user expectations and the required features are extracted by the cooperation with WP3.

WP6 deals with the exploitation and the offering of the partners as well as with the collection of offerings from external partners. Hence, we retrieve the actual content of the marketplace from WP6 in form of a list of offerings.

In the following we consider that this deliverable describes the marketplace model that enables the design of a marketplace for the application scenarios like those applied in WP3, using a description like those defined in WP1 for offerings like those provided in WP6.

As implicitly indicated in the above figure, this cooperation is considered as an iterative approach, where initial requirements – basically from similar platforms in previous projects – are used as an input, an initial prototype of the marketplace is developed, the corresponding marketplace model is derived. This sequence is iteratively performed by improved requirements for common understanding (WP1), usage and applicability (WP3) and concrete content that ensures the practical usage (WP6).

The current status is that the first marketplace model is specified, based on all internal offerings, reflecting all pilots and considering currently identified barriers and descriptions.

Deliverable Structure

First, the marketplace, innovation shop and model-based approach is elaborated. This includes the different perspectives, (a) business perspective – how to collaborate with interested 3rd parties as consumer / provider, (b) technical perspective – how to enable an open source and standard based approach that interprets the model, (c) how to conceptually describe the offerings at a marketplace to allow integration.

Second, the requirement analysis. For the business perspective the model needs to be populated by executing an onboarding process, the technical requirements consider information that is needed by platforms like deployment or price information, the conceptual requirements consider the (smart / autonomous) interpretation of the model for marketplace behaviour.

2 MODEL-BASED MARKETPLACES

The chapter addresses preliminary background of developing marketplaces applying a model-based approach as stated and positioned in D2.1 of the Change2Twin project. The starting points for the discussion are scenarios how such systems are designed, used and applied. These findings are the result of previous research and innovation projects and available marketplace systems.

2.1 SCENARIOS OF MARKETPLACE AND INNOVATION SHOP

"Marketplace" is a widely used term, which gains more and more popularity due to the digital transformation, where IT-infrastructure (e.g. IaaS, PaaS), software services or workflows (e.g. SaaS, BPaaS see [3]) or potentially anything else (e.g. XaaS) can be provided as a digitized service, useable for well-defined application scenarios. Besides the business impact for the IT industry that shifts from on premise sales to cloud-based rental of software, the interesting part is if and how non-software services can be digitized as well and offered in a similar manner.

There are 5 digital business strategies: (a) Customer Centric, for making customers life easier, (b) Extra Frugal, driving the "less is more" culture, (c) Data-Powered, providing intelligence via analytics and software intelligence, (d) SkyNet, using machines to improve productivity and (e) Open and Liquid, creating an ecosystem and a sharing community [4].

Such strategies relate to the digitization of the offering. The **extra frugal** culture "simplifies" the offerings to such an extent that even complex interaction and tasks can be easily handled by the user. This does not mean that c**ustomer centric** approach cannot be digitized e.g. by providing virtual customer support or distance training, but the idea of placing an offer on a marketplace, which can be used by a consumer without further interaction of the provider, is well exemplified by the extra frugal approach. The **data powered** strategy undoubtable requires digitization in order to have data from real world artefacts and apply intelligent technologies. Such intelligent technologies are provided e.g. as Data Analytics as a Service (DAaaS as in e.g. [5]) or in form of artificial intelligence platforms as a Service (e.g. Tensorflow environment). The so-called **SkyNet** approach uses automatisms and robotics for improvement of productivity where digital offerings can help [6]. The **Open and Liquid** approach involves the user to become part of the community by establishing an ecosystem.

The challenge is to provide offerings like online shops, virtual meetings, virtual trainings and conferences as well as robotic and sensing technologies that enable to work from distance, in a form that the consumer can access, evaluate and use it.

We particularly state here that an offering can either be bought – which implicitly indicates that there is a high Technology Readiness Level (TRL), along with a service level agreement (SLA) and a price – or the

innovation item can be used experimentally – which implicitly indicated that there is a low TRL, with a “best effort” agreement and an academic or community code of conduct.

Some marketplaces as well as innovation shops (resulting from past project collaboration) that have informed the development of the context are listed below and shown in Figure 2 and Figure 3.

- https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/horizon-results-platform/search
- https://portal.gaia-x-demonstrator.eu/services
- https://www.ibm.com/marketplace/workbench/
- https://appsource.microsoft.com/en-us/marketplace/apps?product=teams

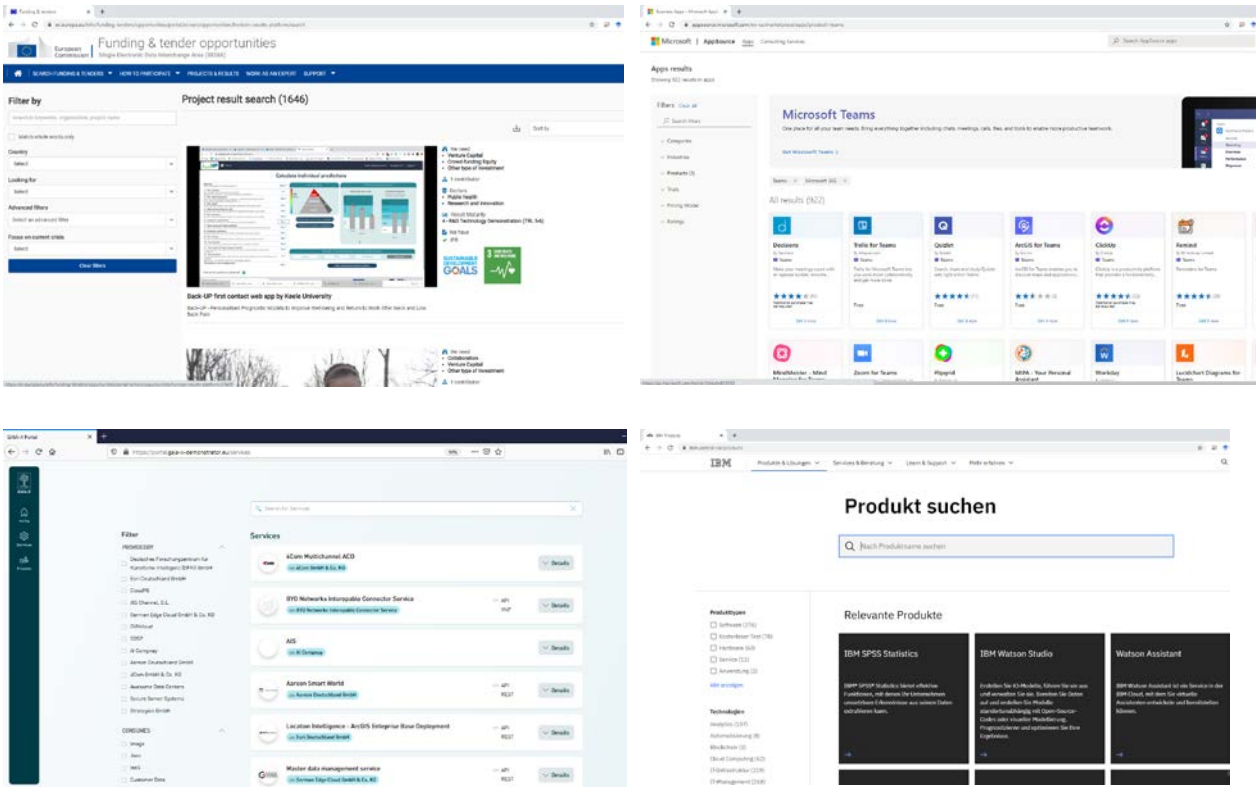


FIGURE 2 EXAMPLES MARKETPLACES

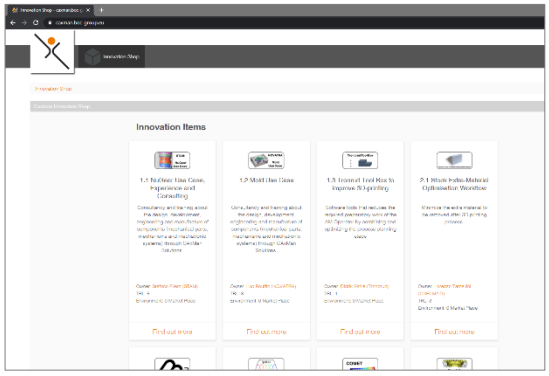
Such marketplaces often not only provide offerings to the customer but aim sometimes to also provide a whole ecosystem. Such full-fledged ecosystem may include not only the offering but also the operation of the services, payments, bonus material, additional collaboration and cooperation services, smart interactions and the like.

We therefore envision such marketplaces not only as a web-landing pages where a user finds appealing information but as an “offering hub” in a network of hubs that evolve towards a full-fletched problem-solving

ecosystem. This includes the aim to motivate Digital Innovation Hubs (DIH) to also participate as such a "hub" to realize the interconnection of several, potentially specialized marketplaces.

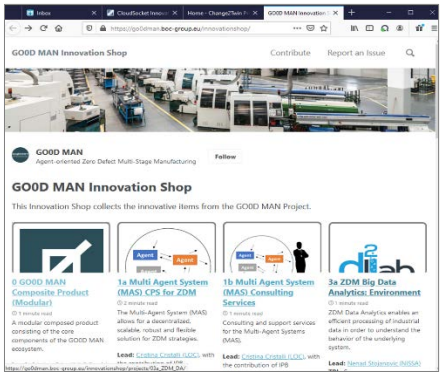
The differentiation between marketplaces and innovation shops and the development in Change2Twin is twofold:

- Domain-specific context: the marketplace in Change2Twin specifically targets digital twinning, as such the domain is embedded within the design and realisation process
- Marketplace ecosystem: we envision that, based on the above observation, the Change2Twin marketplace is an open infrastructure that accommodates both, innovative research results and commercial offerings in a way that users and providers can align their requirements and create novel techniques in the domain of Digital Twins.



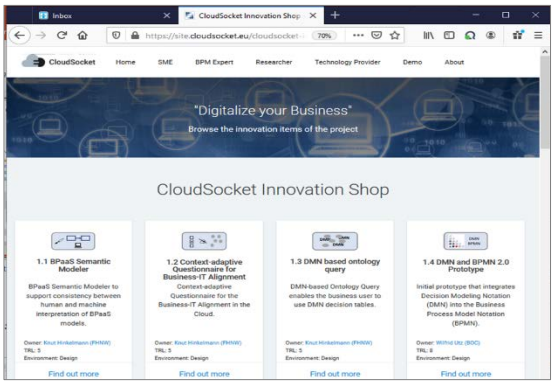
23 Innovation Items

https://caxman.boc-group.eu/innovation-shop



15 Innovation Items

https://go0dman.boc-group.eu/innovationshop/



16 Innovation Items

https://site.cloudsocket.eu/cloudsocket-innovation-shop

FIGURE 3 EXAMPLES INNOVATION SHOPS

We therefore see the following distinctive elements of the Change2Twin marketplace approach:

- From a single marketplace to a network of hubs: by inviting and motivation associations and organizations to also host such a marketplace. Initially addressed focus groups are DIHs.

- From marketplace offerings to innovation offerings, by distinguishing between offerings with a market ready technical readiness (TRL > = 8) and so-called innovation items that also allow lower technical readiness (TRL < 8).
- From marketplace to solution providing an ecosystem, by providing additional tools and methods that support users to identify their needs, perform digital transformation workshops, apply assessment tools, and enable the accompaniment of the customer from the design of the challenge till its solution in the field.

This design and marketplace model hence aims at establishing a common conceptual infrastructure that can be used individually for one marketplace, but also linked and/or used by several marketplaces to provide a network of solutions and enable the provision of additional tools to establish an ecosystem of tools and methods.

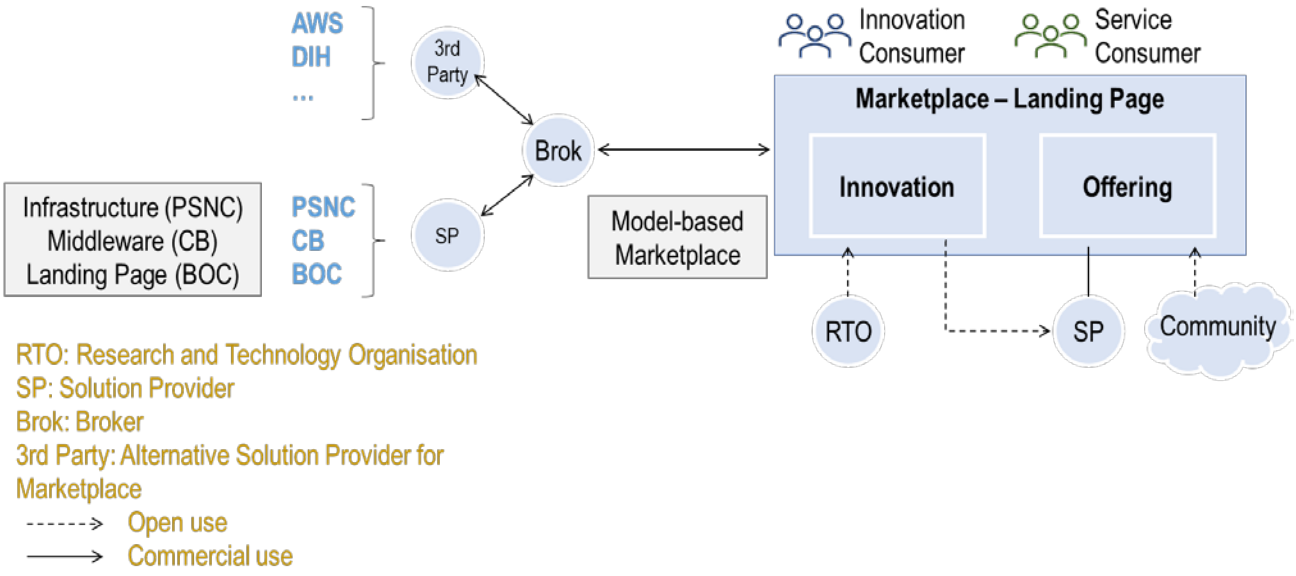


FIGURE 4 OVERALL TOPOLOGY OF THE MARKETPLACE

Figure 4 introduces the overall topology of our marketplace concept. Initially, the marketplace landing page is a webpage that is used by the consumer for either accessing commercial offerings, or innovation items, which are either provided by RTO – Research and Technology Organization, SP – Service Provider, or by a community like but not limited to a standardization association or open-source community.

Such a marketplace hub is operated and provided by a "Broker", where we apply similar concepts of the brokerage of offerings like it is known from the cloud services industry.

The broker offers services that are directly related to the operation of such marketplace, like in our case the landing page, the middleware that is tracking offering and consumers and the appropriate IT infrastructure to operate the different software and data services.

In addition, the broker can introduce additional service – we mentioned 3rd party services – where useful features can be added to complement the marketplace and its offerings. In our case, we introduce assessment tools and innovation workshop environments to better identify the users challenge and select the most appropriate solution. As an open approach, the system allows for the extension and addition in the sense of further guidance that helps the user to better identify the path towards a solution.

FIGURE 5 provides a high-level representation of the envisioned approach:

- Marketplace model: represents the conceptual representation of a specific marketplace (domain-specific, DIH context) and enables expert interaction,
- Marketplace landing page: is generated from the model and enables user interaction.

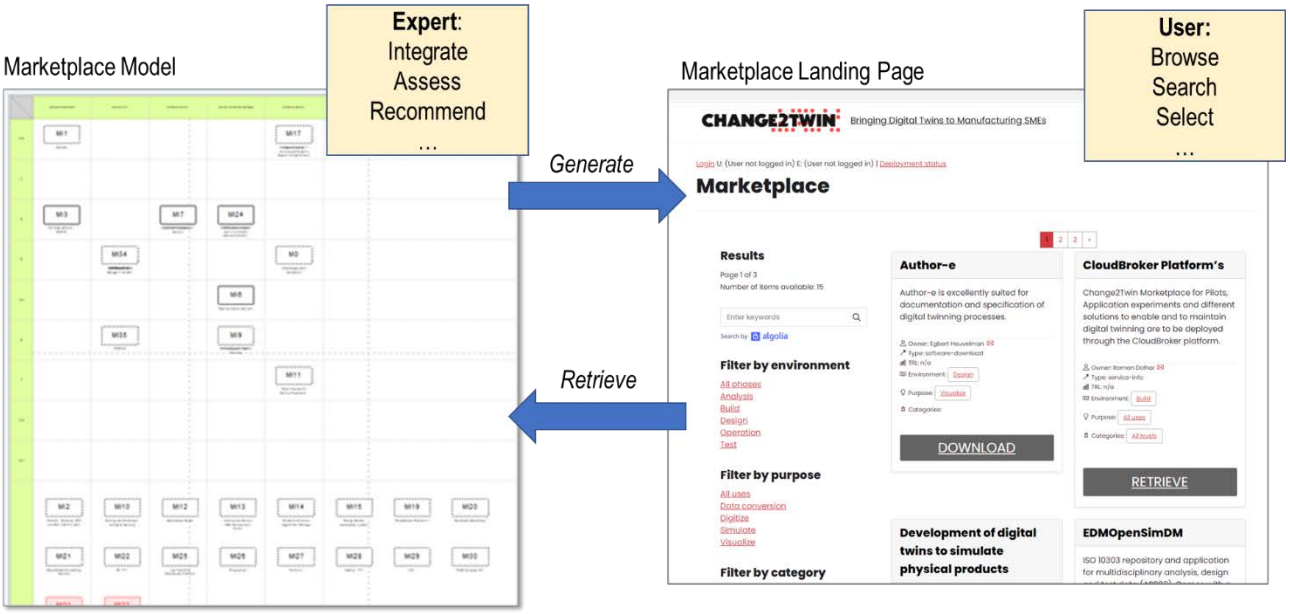


FIGURE 5 C2T MARKETPLACE INTERACTIONS AND CAPABILITIES

This flexibility requires a separation of concerns, whereas the basis technology is capable to deal with offerings independent of the domain, technical readiness and nature of the offering. In a second step the domain specific context – in our case the offerings of "digital twinning for SME manufacturer" is added to provide domain-specific support.

2.2 BUSINESS PERSPECTIVE OF MARKETPLACE

The business perspective of our marketplace model has a close relationship to the individual exploitation plans of each partner. The corresponding offering to each individual exploitation item is collected from the partners – in cooperation with the exploitation activities – and added as items on the marketplace.

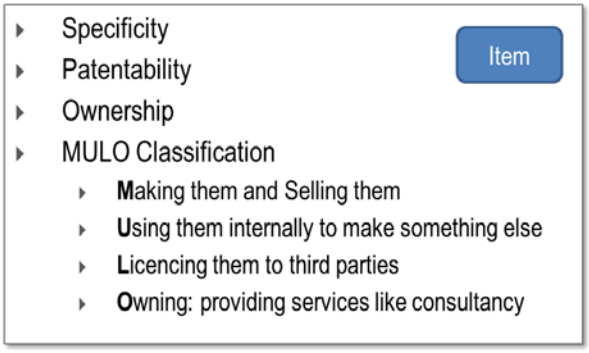


FIGURE 6 CLASSIFICATION OF EXPLOITATION ASSETS

Figure 6 introduces the classification of exploitation assets that is provided in form of guidelines for European Projects.

We apply the criteria specificity, patentability and ownership to identify the item. The "MULO" classification is used to introduce the nature of the offering. As a support instrument in the sense of openness, we introduce the "onboarding process", which is a systematic way to:

- identify potential interested parties that want to place an offering on the marketplace,
- describe the item in a cooperative workshop with DIH experts and experts from the marketplace,
- review the item concerning the correct representation, the conceptual correctness and the technical appropriateness, and finally
- release the item on the marketplace.

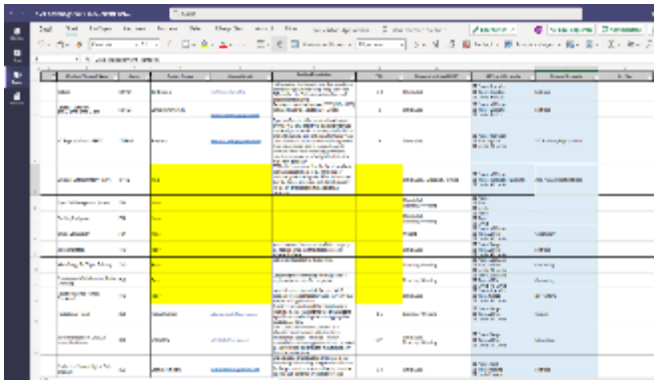


FIGURE 7 SAMPLE OF USING EXCEL AND TEAMS FOR ONBOARDING

Figure 7 indicates that initially we are using an Excel sheet in the collaboration platform TEAMS to collect the items. As those items are not finally approved, we kept the figure small to avoid that incorrect offerings

and descriptions are displayed. Figure 7 gives an impression, on how the collection process is performed within a trusted network like in our consortium.

Additional an approach, like a ticket handling workflow with publicly accessible forms on the web are proposed and will be provided (see section 6.2.2 for details). However, other techniques of onboarding may be required for the interested DIHs that do not belong to the consortium.

2.3 TECHNICAL APPROACH TO THE MARKETPLACE

The technical perspective of the marketplace is prominently accessible in form of the landing page that shows the different offerings and provide some additional information for the users. Although the appealing appearance of a marketplace is a very important aspect, we want to point out that the underlying concept is that customers will be supported by DIH or consulting companies in form of a moderator or facilitator. Hence the technical challenge is about connecting a wide range of different offerings and enable the combination of assessment and consulting tools by approaching the marketplace as a repository of solutions.

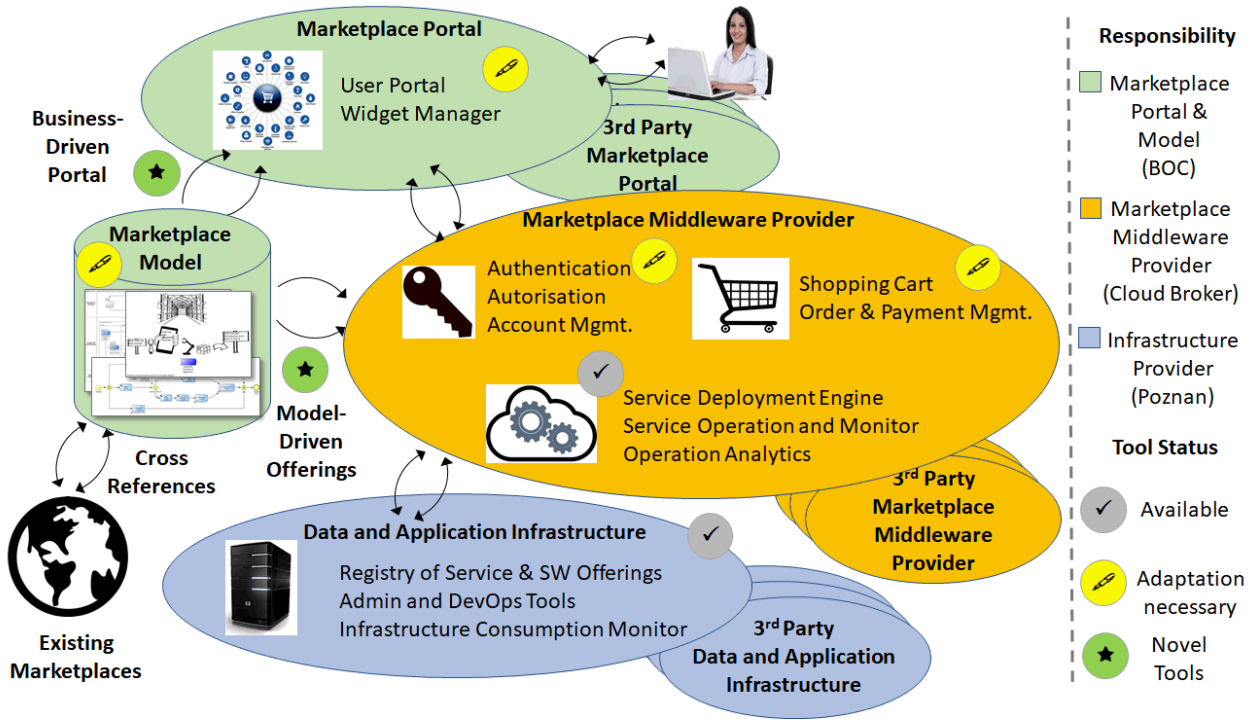


FIGURE 8 HIGH LEVEL ARCHITECTURE OF THE MARKETPLACE

Figure 8 depicts the three layers as well as the roles involved:

(a) **Infrastructure Provider:** provides infrastructure services, administration and DevOps Tools, infrastructure consumption monitoring and appropriate 3rd party data and application infrastructure. Currently, we use this infrastructure for the marketplace itself, and so-called software offerings that run on this infrastructure. For the software offerings that run on another provider infrastructure, the

so-called marketplace infrastructure is then not used for that service. Furthermore, there are also consulting or information offerings, which also do not need this infrastructure.

(b) **Middleware Provider:** realizes the so-called program or business logic of the software package that creates the marketplace. User authorization, shopping cart, order and payment services, as well as a deployment engine for those software services that run on the marketplace infrastructure.

(c) **Marketplace Portal & Model:** consists of a landing page that is created using a set of open source and standard technologies to ensure an vendor independent front end. The marketplace model, which is one of the key aspects discussed in this document, enables "smartness" and "interoperability" within the ecosystem. Concept models and the corresponding repository are provided in form of a meta-modelling platform that enables the mapping and transformation from concepts from other platforms and the integration of smart assessment and search tools not only on the landing page, but as an interface to a full-fledged application that accesses the marketplace to retrieve a structured set of solutions.

Before we explain details on the used technology, the possibilities to extend current solutions and the current status of the development later in the document, we introduce the marketplace perspective in three dimensions: business, technical, conceptual.

2.4 CONCEPTUAL DESIGN PERSPECTIVE OF MARKETPLACE

The conceptual perspective of the marketplace enables the "interconnection" with other marketplaces and the potential usage of "smart" features by providing concept models and the corresponding repository that describe all items on the marketplace. The conceptual model of the marketplace applies modelling concepts and techniques in order to provide a conceptual / digital representation of the marketplace that enables features such as search, comparison, mapping, assessment and the like.

Figure 9 introduces the role of the marketplace model that - in its simplest form - could be understood as a list of items, which uses a taxonomy to annotate each item with the corresponding context. In the later section, we introduce the full-fledged concept model and visualize that it is not only a list of items but a concept model that provides additional capabilities based on the model representation.

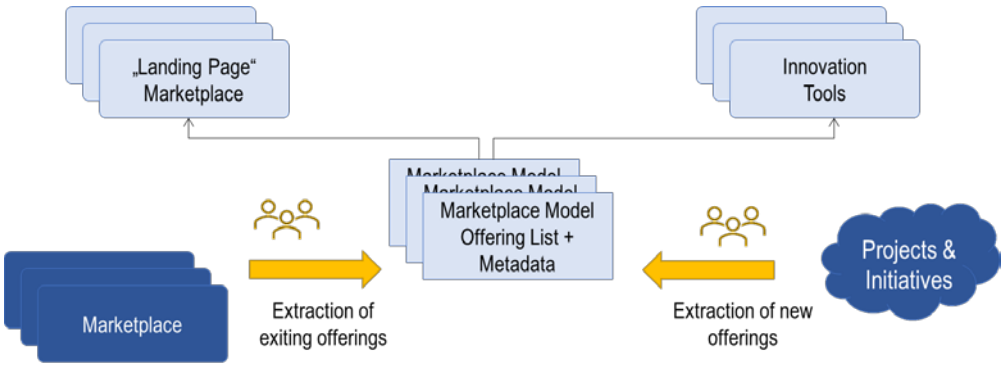


FIGURE 9 MARKETPLACE MODEL AS CONCEPTUAL INTEGRATOR

Those conceptual models, where each marketplace is represented as a descriptive model of the items and each application sub-domain is represented as context models, are established in an open modelling format and technically stored in a meta model repository that is accessible via a ReST interfaces. This enables that a special software package that exports the descriptive item model of the marketplace can be used to generate the “items description” automatically on the landing page and tools enabling searching, mapping, comparison or inferences access the “item context” model. We therefore see two usage paths:

- The user and / or expert, who accesses the landing page of one of the available marketplaces.
- The expert, who accesses a consulting tool is accessing the marketplace via the marketplace model.

The onboarding process is applied for items that are already offered and need a description and a mapping for referencing the existing offerings by using the marketplace model. The newly created items, like individual exploitation results of project partners, are extracted and described to be listed on the marketplace.

For the “expert” interaction path, an example is graphically shown in Figure 10 and provided as a proof-of-concept prototype embedded in the modelling tool. The approach chosen builds on the semantic lifting technique introduced by annotating capabilities and requirements with defined concepts (technically in RDF representation, utilizing conceptual graphs as a representation formalism [7]. Matching and discovery mechanisms that utilize projection of conceptual structure allow for a retrieval of adequate items from the marketplace model.

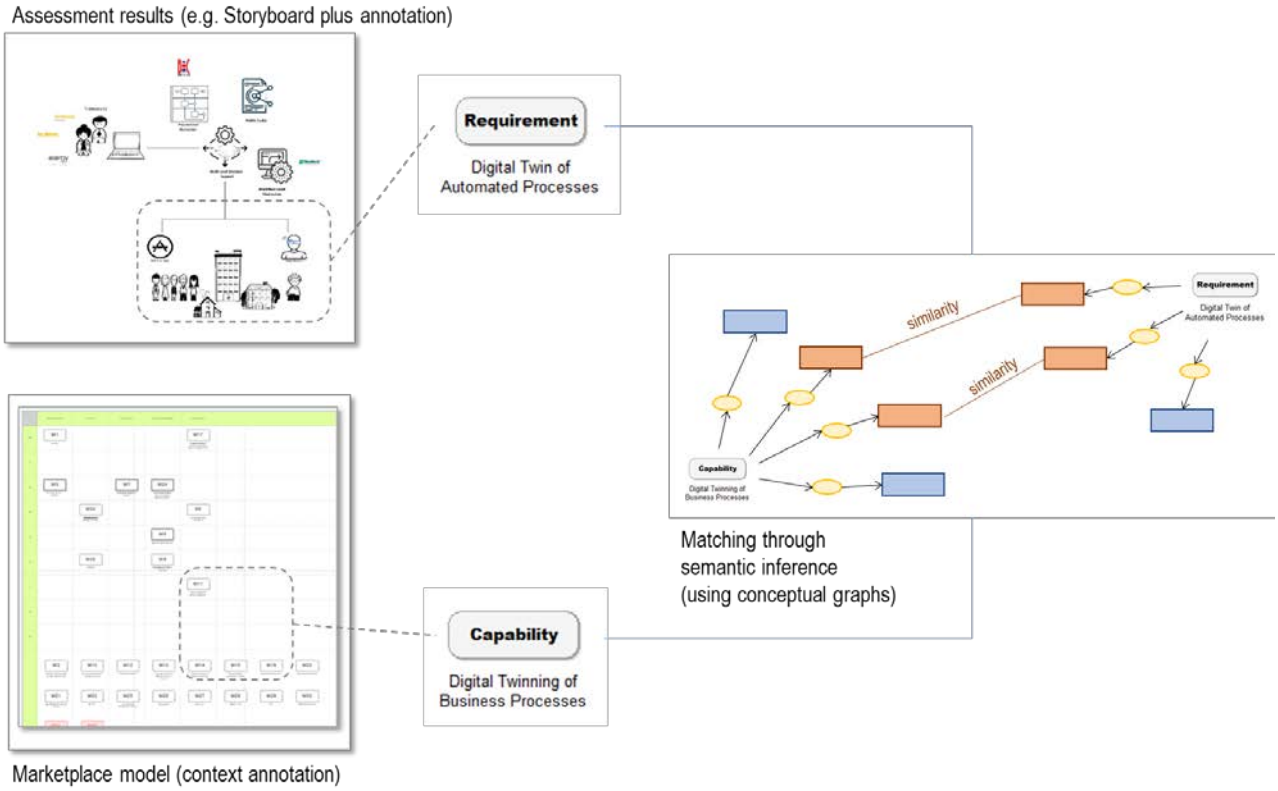


FIGURE 10 SIMILIARITY MATCHING USING SEMANTIC INFERENCE

2.5 INITIAL LIST OF IDENTIFIED DIGITAL TWIN OFFERINGS

The following items have been identified initially and been used to establish the concrete requirements towards the model-based approach. The template applied for each item and based on the achievements of D2.2 is provided in Annex B.

ID	Owner	Long-Title	TRL
MI1.1	SINTEF	Geometry Tools for CAD with support for IgA	5
MI1.2	SINTEF	SINTEF Spline Library	8
MI3.1	TTTECH	Industrial edge computing platform	9
MI3.2	JOTNE	Open Standard Digital Twin – PLM IoT platform	9
MI3.3	JOTNE	EDMSoftwareDevelopmentKit	9
MI4.1	FBA	Open Call Management System	9
MI4.2	FBA	Open Innovation Services	
MI4.3	FBA	Online Community	9
MI5.1	TNO	Digitalisation Assessment	8
MI5.2	TNO	Smart Connected Factory	7
MI5.3	TNO	Seven Step strategy for Digital Twins	7
MI6.1	BOC	Marketplace Model enabling Smart Marketplaces	4
MI6.2	BOC	OMiLAB Innovation Corner – Innovation and Assessment Service	5
MI6.3	BOC	Digital Twin of Production Process - Academic	6
MI6.4	BOC	Digital Twin of Production Process - Business	9
MI7.1	UNIBO	Design of Hybrid Digital Twin Solutions	6
MI7.2	UNIBO	Quality-Aware Cloud-to-thing Management&Control Platform	6
MI8.1	CB	CloudBroker Platform	8
MI8.2	CB	Marketplace Download Package to install own marketplace	5
MI10.1	PSNC	Marketplace landing page running as a service	6
MI10.2	PSNC	Cloud, HPC and data storage infrastructure and software	9
MI11.1	SPS	Development of digital twins to simulate physical products	9
MI13.1	Unit040	Prespective Digital Twin Software for Unity	9
MI16.1	CT-I	Interface Development	8
MI16.2	CT-I	Software and consulting services industry 4.0	8

3 REQUIREMENTS FOR MARKETPLACE MODEL

Based on the above observations, the requirements for the model-based design methodology as well as the marketplace model are derived. Requirements are distinguished and classified according to the business, technical and conceptual dimensions. Requirements are introduced and summarized in the following sections and provide input for the definition of the model and design methodology in section 4-6. As such, this chapter details the observation of chapter 2 and sets them in the context of the work performed in this task.

3.1 REQUIREMENTS FROM BUSINESS PERSPECTIVE

We approach the elicitation of requirements that are needed to describe business-aspect of an offering in the following ways:

- From the Exploitation Perspective: to provide capabilities to represent offerings for commercial exploitation,
- From the Business Modelling Perspective: to support novel and innovative business models, and
- From the Pilot Perspective: to enable pilot interaction with the marketplace to create innovative solutions.

3.1.1.1 Identified Requirements from the Exploitation Perspective

We use our own project and the corresponding individual exploitation plans of the partners as our initial set of offerings that are provided on the marketplace to elicit the requirements in this dimension. Hence, we consider each exploitation asset resulting from the project as a candidate to become an item on the marketplace.

Different Types of Offerings

We observe different types of offerings – not limited to – based on the characteristic of the provider in the form of:

- Companies who provide commercial products and services,
- Research institutions and Universities who provide concepts, frameworks, consulting and teaching,
- Partners – independent if they are companies, research institutes or Universities – who provide prototypes that are not yet market ready,
- Pilot Partners and corresponding consulting companies or technology provider who do not provide a particular offering, but identify, select, integrate, apply and test solutions for a particular pilot,
- Partners who provide or contribute to data, standards, models and formats,

- Partners who provide knowledge, competence and experience in analysing a use case challenge and developing a solution for the concrete use case challenge,
- Communities who provide open-source or open use packages as well as standards or de-facto standards that are provided with the goal to retrieve larger acceptance and distribution.

Following an open approach, offerings are independent of the nature or type, everything that may be useful to a potential customer is provided with the limitation that it can be identified as an offering or asset.

Different Technology Readiness of Offerings

In research projects we observe:

- Market ready products that are provided in the project for usage,
- Research prototypes that are not yet market ready, but due to evolution in such innovation actions they aim to become market ready soon,
- Ideas, concepts and findings that show disruption potential, but are currently not appropriate to be installed and demonstrated in a real-world environment.

Although our aim is to identify, describe and structure the provided offerings, we also want to enable a broad field of offerings, where innovations that are currently not yet market ready can be transparently distributed.

We understand this principle as “Innovation Shop” that enables a loose coupling between research, innovation and the business domain. The underlying concept is based on the “non-physical knowledge product” in the domain of knowledge management, where experience, knowledge, consulting, and the like are defined – based on the SECI model from Nonaka / Takeushi from the 90s [8] – in (a) information provision – in case of knowledge externalization, (b) human interaction – in case of knowledge socialization and (c) application provisioning – in case of knowledge combination.

This ideas of “knowledge products” has been elaborated in the field of knowledge management consulting and applied for innovation management, by complementing the approach with the evolution along a lifecycle – e.g. the knowledge maturity model developed in the MATURE project Maier [9], [10] in order to support research and innovation projects.

EU-Projects (H2020) like CaxMan [11], CloudSocket[12], GO0DMAN successfully applied the principle of the so-called “Innovation Shop”, hence we also apply the open approach that allows **“any identifiable asset – independent on the technology readiness that has the potential to be useful to the customer”** to be listed and described on our marketplace.

Individual vs. Joint Exploitation

From a business dimension, the approach of “Value Adding Networks” is followed where in case of a joint offering the corresponding partners agree on a joint offering in the sense that the contact to the customer is

made by one partner, but the provider – customer liability stays between the individual providers and the customer. We consider this a "coordinated" value adding network, where the customer owner is taking the leads, but when it comes to responsibilities, the customer has individual agreements with the different providers.

From a marketplace point of view, we need to provide two types of offerings:

- Individual offerings as atomic solution sets, and
- Solutions that are provided by a set of individual offerings.

This does not limit the possibility that a joint offering evolves from a "value adding network" towards an offering where the former "coordinator" of the network takes the full responsibility and treats the other partners as "sub-contractors" by taking the full responsibility to the customer. Independent on the legal and organizational construct, this is treated as an individual offering on the marketplace, as for the customer who enters the marketplace, the bilateral agreements between the provider and its sub-contractors are considered as transparent.

3.1.2 Identified Requirements from the Business Model Perspective

The generation of business models is an important and organisational specific and the concepts and foundations of developing a business model is out of scope of this marketplace. However, we consider current approaches for "digital transformation / digital optimisation" with the aim to support them with the marketplace.

Within the project we aim at supporting the following approach, which we consider as sample of similar methodologies, and hence do not limit the marketplace to concrete instances within the project only, but rather focus on the basis needs of such approaches.

- *Assessment Tools:* We consider assessment tools that identify the potentials of digital transformation or optimisation – in our case the usage of digital twin technologies – as a possible access point to the marketplaces. Aforementioned different types of offerings should be provided during the assessment of a pilot case, in order to provide similar cases, identify experts for the particular challenge or identify potential solutions and offerings to solve concrete challenges from the pilot.
- *Business Modelling Tools:* There are several business modelling tools like e3value model, the Business Model Canvas and other forms that identify and describe current and / or future dependencies of business partners. Such tools are mainly used in form of graphical concept modelling tools, hence we enable the identification of a concrete offering from the marketplace also in form of concepts via the market-place model.

- *Innovation Tools:* There is a plethora of innovation tools that are applied to support the digital transformation or optimisation of organizations. This includes the usage of laboratories – like the OMiLAB Digital Innovation Environment (DIEn) [13] that is provided in Change2Twin. We currently consider the identification and selection of offerings as a combination of the aforementioned assessment tools or modelling tools, where the offerings are either provided by special interfaces to existing assessment tools, by models that can be exchanged, or by manually searching on a landing page. However, we also do see potential in the element of the so-called "physical experiments", which demonstrate in a sandbox the intended solution. When more pilots use such "physical experiments" we need to revisit the requirement specification in order to decide, if the usage or demonstration in form of "physical experiments" has influenced the requirements on how an item needs to be described.

Aforementioned requirements indicate that the usage of the marketplace is not limited to a user who visits a webpage – e.g. the landing page of the marketplace – but that we proactively aim to push the concepts of the innovation shop and / or marketplaces to the business layer, where in early phases of ideation the potential of new approaches is worked out and demonstrated.

3.1.3 Identified Requirements from the Pilot Perspective

As a proof of concept, we check if the marketplace has a benefit to the pilots that have been selected in the project and working jointly on solutions. This check is implicitly performed by continuously collaborating with the people that are responsible for developing the pilots and checking, if the used solutions can be and are described in our marketplace model and what is an appropriate provisioning of the marketplace items.

For this reason, we have regular workshops – internal weekly handholding meetings within a period of about two months – where each partner – provider and consumer – participates to a workshop on how to describe offerings that they are currently using in the pilot cases. The results can be summarized as:

- The concept of the marketplace is very well received, some partners – even 3rd parties from outside the project, raised their interest to place own offerings on the marketplace.
- The complexity on correctly describing the offerings has been partly underestimated, especially when it comes to low technology readiness items.
- Sometimes iterative work is needed to identify the solution when a simple search and selection of items is not appropriate.
- The wrong expectation in particular technologies cannot be resolved by simply offering or selling this technology but requires interactive consulting to correctly map expectation with offering.

The requirements on the marketplace can therefore be summarized that it should be a collection of items that are explained in such a form that a consultant – e.g. a DIH – can use the catalogue of items in order to improve the consulting experience on the customer side.

3.2 REQUIREMENTS FROM TECHNICAL PERSPECTIVE

We distinguish between the following technological aspects:

- Architecture principles on the major building blocks to ensure flexibility
- Usage of technology to support an open development and avoid vendor lock-in.
- Operation of marketplaces for Digital Innovation Hubs

The marketplace is constructed with the goal to be as flexible as possible and avoid vendor lock-in. Hence, we provide loosely coupled components that

- exchange data in widely used formats like Markdown files or Docker-Images
- interact in simple ways like triggering events or performing only basic queries.

The goal is to keep the architecture very light weighted making the re-implementation or exchange of components as easy as possible.

3.2.1 Core Components of the Marketplace

We consider 3 building blocks for the “core marketplace”:

- *Landing Page:* The landing page is prominently displayed to the end user. Hence, we assume that it will either be integrated in the platform of the marketplace provider, or integrated in other tools like assessment or consulting tools. We therefore propose to use a lightweight front-end framework that can be easily exchanged, adapted or integrated. Micro-frontends – like those provided in the open-source framework OLIVE [14] provide the flexibility to integrate the user interface into web-based applications. In order to avoid also the dependency on the OLIVE framework, we propose also the realization of a pure static generated web-frontend, which uses standard technologies like Markdown and state-of-the-art generators like Jekyll to generate micro frontends for each marketplace item. Based on the experience in previous projects – CaxMan, CloudSocket, GO0DMAN, DigiFoF[15] – BOC concluded that the use of Markdown files to store the content of an item and Jekyll transformation to create the according user interface, seems to be a flexible and powerful approach.
    - First, the Jekyll transformation rules is widely known in the software engineering industry as a basic technique in GitHub – the “de-facto standard” for open source projects.
    - Second, the transformation rules can be easily adapted to also meet the requirements for not only a web interface but potentially any interface than can support widgets like mobile apps, or many software applications. Hence, an integration and evolution is possible.

- Third, the static generation allows the usage of the marketplace also without Internet connection, so it can be run locally. This enables the storage of the whole marketplace on a mobile hard disk as well as reduces dependencies on the Internet connectivity that may be helpful in some settings e.g. security or networking issues.

- "Middleware": The middleware introduces the so-called business logic – in our case the shopping cart - of the marketplace like the awareness of which user selects which items and enables its deployment and usage. As our marketplace builds on the commercial license of CloudBroker, we use this component to rely on an established middleware, which need to be adapted in such a form that models from the marketplace models can be used to configure parts of the middleware. The interaction with the landing page and the underlying IT-Infrastructure shall be reduced to a minimum.
- "IT-Infrastructure": The operation of a marketplace is a complex task itself. The software packages need to be deployed on servers with
  - the appropriate hardware resources, e.g. number of processors, memory, storage size,
  - the appropriate software environment, e.g. operating system, database management system, application server, monitoring software, and
  - adequate throughput to scale in and out depending on how many users perform interactions on the marketplace.

  The complexity raises in case the provided marketplace items are software services that need also to be deployed and operated on the IT-Infrastructure. Hence, in addition to the usage of the marketplace, the IT-Infrastructure has to cope with the usage of the individual items like applications, simulations, visualizations, data handling, etc. These services vary extremely related to the requirements (e.g. computational performance, storage efficiency, throughput, etc) and resource consumption. Therefore, different forms of marketplace offerings need to be possible. We assume that approaches like Docker images / containers, monitoring of deployed container and container management are appropriate for such a marketplace.

3.2.2 Add-On Components

The consequence of a flexible and light weight approach of the core components is the necessity to enable add-on components in order to complement the available functional capabilities with user or domain-specific functionality.

We therefore expect add-on functionalities like but not limited to:

- User authentication, by re-using a standardized solution.
- Search functionality, that provides extended search functionality on the marketplace.
- Onboarding functionality, that supports the provision of item description and a systematic onboarding process of marketplace items.

- Payment service, that enables the payment of items.
- Smart assessment and mapping features, that enable smart mappings of offerings to user requests.
- Integration with assessment and consulting tools
- Searching in other marketplaces to extract a marketplace model from existing offerings from third party marketplaces.

The aforementioned list of add-on functional capabilities needs to be reflected in the design of the marketplace to allow extension, configuration or adaptation for the particular need of the marketplace provider.

3.2.3 Marketplace Model-Repository

The marketplace model acts as an orchestrator that moderates the ecosystem around a marketplace. The model representation has the following characteristics:

- A conceptual definition of the marketplace model – the so-called meta model – defines the necessary semantic primitives in order to describe the content, its meaning and the behaviour of the marketplace.
    - The content is described by a meta model about items.
    - The meaning is described by semantic models that specify the context of the items.
    - The behaviour is described by a meta model that enables configuration of the marketplace.
- A technical marketplace model repository, the so-called meta modelling platform, technically realizes the repository and enables the storage, access and manipulation of the marketplace and semantic model. Each set of marketplace and semantic models defines the ecosystem of one marketplace. The integration of several marketplace models results in a system of systems that enables the orchestration of several marketplaces and their corresponding ecosystems.
- A tool environment that enables
    - the generation and usage of marketplace models,
    - the access and integration of marketplace models in other tools,
    - the introduction of semantic for the application of smart algorithms,
    - the operation of the repository, and
    - the development environment that allows the adaptation and evolution of the marketplace model.

The aforementioned list of issues is based on model-based scenarios in knowledge management and innovation management.

3.3 REQUIREMENTS FROM CONCEPTUAL PERSPECTIVE

The conceptual perspective introduces the possibility to apply smart search and mapping algorithms. The aforementioned description of the items is extended by the so-called context of the items. The context may be expressed in semantic models like an ontology, which is elaborated also in D1.4. In the following we explain the different mechanisms on how to integrate the conceptual perspective in order to enable smart algorithms.

3.3.1 Introductory Sample of Smart Algorithms

The introduction of additional semantic into the marketplace model is an additional effort but enables the application of smart algorithms. There is a plethora of algorithms like but not limited to:

a) **Smart Search:** When searching for keywords, the search results are not only providing the items that are described by the exact keywords, but items that are described with keywords with a reasonably close "semantic distance" are also shown as a search results. Such a smart search may be used on the landing page.

b) **Smart Mapping**[1]: When describing a particular need of the customer by using keywords, the corresponding set of solutions – in our case items – can be mapped by using semantic matchers that match the requirement description with the offering description. Such a mapping may be used in modelling or assessment tools.

c) **Questionnaire-Based Interaction**[2]: Questions and answers are a common interaction for a user when searching for a solution concerning a concrete challenge. The basic principle is that the annotation of request is performed while answering the questions. Each answer describes the semantic context of the query. This can be realized in different forms, in the prototype mentioned here, the provided answers for a particular question are used to annotate the query while interacting with the users. The questions are selected in order to maximally reduce the result set. Hence, the questions are selected according to their contribution to reduce the resulting item set in order to optimize the interaction. This means, the user has to answer only a minimum number of questions to retrieve the set of solutions. This algorithm may be used for both, the landing page or within modelling- and assessment tools.

[1] An open-source prototype of a semantic matcher is provided in the Innovation Shop of CloudSocket: https://site.cloudsocket.eu/cloudsocket-innovation-shop/-/asset_publisher/47zCOQayhR2E/content/1-1-smart-business-and-it-alignment

[2] An open source prototype of a questionnaire based interaction is provided in the Innovation Shop of CloudSocket: https://site.cloudsocket.eu/cloudsocket-innovation-shop/-/asset_publisher/47zCOQayhR2E/content/1-2-questionnaire-based-annotation

d) **Chatbot**: Similar to the questionnaire-based interaction the communication with the user can also be realized in chatbots, where based on the underlying semantic enrichment, the questions and answers are provided in form of chat answers. This is often used to create and train a knowledge base, where first the question and answers interaction is performed by human workers which are then iteratively exchanged with automated chatbots.

e) **Recommender System**: When selecting one item, appropriate additional items are automatically recommended. Often those recommendations are intuitive like providing training or consulting for a selected digital twin technology. The recommending algorithm may identify also dependencies that are not extracted from conceptual inference but from the approximation of similarities. Such algorithms do not exactly identify a dependency but assume a dependency according to observations.

f) **Speech Recognition:** Although we consider a full-fledged speech recognition out of scope of the marketplace prototype, in Change2Twin, we want to enable such technology conceptually foreseen through extensions. The syntax and grammar of the speech interaction can be linked to the marketplace meta-model, in order to enable the usage of the marketplace terminology also for the speech interaction.

Although there is a plethora of similar algorithms, we do not want to provide a complete list of possible smart algorithms, but introduce requirements which we need to consider in the design of the marketplace model by using samples we are familiar with and hence can argue the added value.

3.3.2 Conceptual Integration Techniques

Conceptual Models – such as the proposed marketplace model – have the capability to provide a pre-defined semantic, which is defined in the meta model and instantiated while modelling in form of graphically represented modelling objects. There is a certain semantic expressiveness available that can be used for aforementioned or similar smart algorithms.

In order to benefit from additional semantic expressiveness, different knowledge-based approaches are applied or to cover flexibility requirements that are needed for self-learning approaches, the meta model can be enriched with additional semantic. This is known as “semantic lifting” of models, which has been researched in a series of projects and publications [16]–[19]. The semantic enrichment can be performed on the domain level, which means that the marketplace model is semantically lifted via a domain ontology describing the digital twin related aspects of each modelling object.

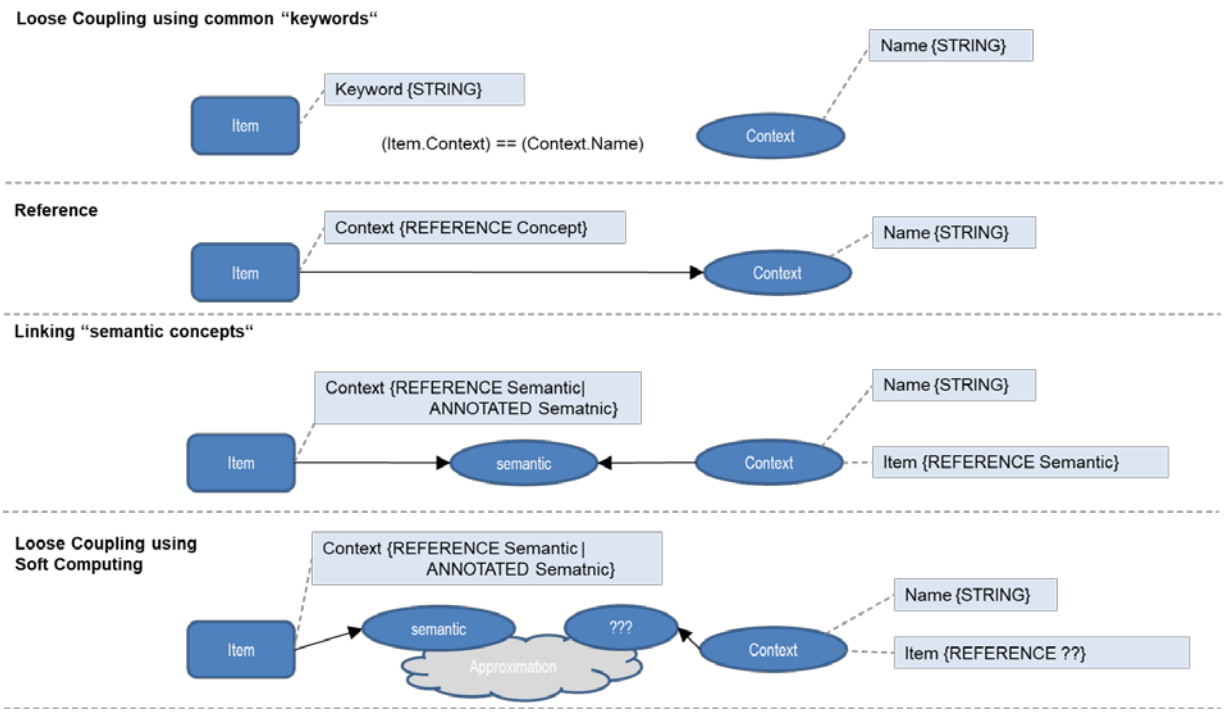


FIGURE 11 DIFFERENT FORMS OF SEMANTIC LIFTING

FIGURE 11 introduces some alternatives to semantically lift a modelling object. In our case we used the modelling object "item" that describes an item on the marketplace and elaborate the corresponding semantic enrichment introducing four different technologies.

1. **Loose Coupling using common Keywords:**

   This is the simplest form with minimum requirements on the market place model. Keywords can be entered in the provided text field. Those keywords must be the same keywords than in the used semantic model. This may be ensured by creating a copy of the keyword from the semantic model and paste it into the marketplace model. Although this technique requires only a textfield, it is quite error prone and not user friendly.

2. **Reference:**

   A more advanced form is the conceptual reference, which enables the linkage – using either a reference or a pointer – of two concepts, whereas one concept is the marketplace item and the other is the semantically expressed keyword. This enables a selection of the keyword without the need to copy or paste it. However, this approach requires that all semantic concepts are imported into the marketplace model in order to be available for reference. This causes redundancies and a time gap, as sematic models need continuously be imported.

3. **Linking "Semantic Concepts":**

   In order keep flexibility, the aforementioned approach is extended by introducing the linkage of semantic concepts. The reference approach is still used in order to enable the semantic description of a marketplace item in a user-friendly way. The reference concept is not directly the semantic

concept, but a so-called “transit objects”, which acts like an interface between the models within the marketplace modelling environment and the semantic description in the semantic environment. There are different techniques, like the invocation of services that enable the read and write between the two modelling environments. The semantic concepts are read from the semantic environment and written in the “transit concept”. Semantic inferences can now be executed on the semantic transit concepts.

4. **Loose Coupling using “Soft Computing”:**
   Current approaches focus mainly on the 2nd or 3rd approach or different variations. The current evolution of soft computing approaches requires that we also consider such approaches, even if we currently do not have concrete samples. With approximation we indicate, that neuronal networks, simulations or other forms of approximations are used for “recommending” a certain item for a specific context.

Although we assume that within the project not all smart algorithms will be realized, we consider such requirements as important to be reflected in the conceptual architecture of the marketplace model.

3.3.3 Requirements raised by Context

The goal of distinguishing the flexible item context and the fixed item description in the marketplace model is to be as independent as possible from the provided context. There is still a dependency that needs to be considered in the requirement analysis, which is the number of annotations for a marketplace item.

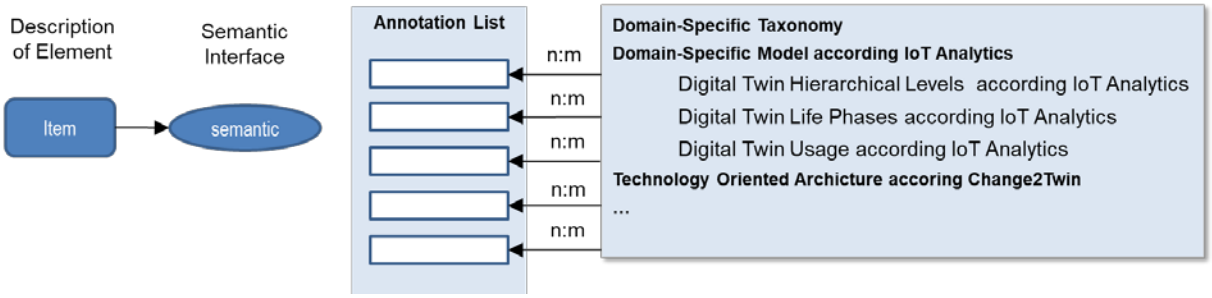


FIGURE 12 NUMBER OF ANNOATIONS FOR A MARKETPLACE ITEM

FIGURE 12 shows that the number of different context descriptions affects the way the semantic description is realized. In this sample we foresee (a) a taxonomy, (b) the hierarchical level of Digital Twin, (c) The life phase of Digital Twin, (d) the usage of Digital Twin and (e) the position in a reference architecture. Independent, if those dimensions are the correct one, there is the need to enable semantic lifting of several independent semantic ecosystems. Conceptually this can be handled in different ways like:

- The semantic ecosystem must combine all semantic models, hence a multi selection out of one semantic system is needed. The marketplace item has one reference to one semantic transit objects, which is selected multiple times by one semantic ecosystem.

- Each transit semantic concept accesses – via multi selection - only one semantic ecosystem, whereas each semantic ecosystem represents one context. The marketplace item has several references to semantic transit objects, each enabling multiple selection by one semantic ecosystem.
- Several semantic ecosystems can be used to not only enable multiple annotations but also several annotations. The marketplace item has one or several references to semantic transit objects, which enable multiple selection using a pre-defined number of annotation fields and hence can access many semantic ecosystems. The sample in Figure 12 provides five possibilities.

The introduction of different forms to introduce smart behaviour into the marketplace model finalises the current requirement elicitation.

4 SPECIFICATION OF MARKETPLACE META MODEL

This chapter addresses the specification of the metamodel for the marketplace model, more specifically the modelling method that provides the structure, interaction capabilities as well as functionality (as mechanisms and algorithms). Following metamodeling approach, the modelling language is defined based on the information model discussed in the previous chapter and elevated with concepts for model processing (e.g. transformation towards representation and technical interpretation). As such the metamodel is positioned as the vocabulary adequate for the definition of arbitrary marketplace models and defines initially the notation, syntax and semantics of the constructs as well as their purpose for realization of a modelling tool capable of interpreting the modelling method and provide tooling (introduced in chapter 5).

4.1 INTRODUCTION IN MODELLING METHOD ENGINEERING

The approach for defining the marketplace modelling method builds on the framework introduced by Karagiannis/Kühn in [1]. The framework acts as a structuring approach to discuss requirements according to the components of a modelling method in general.

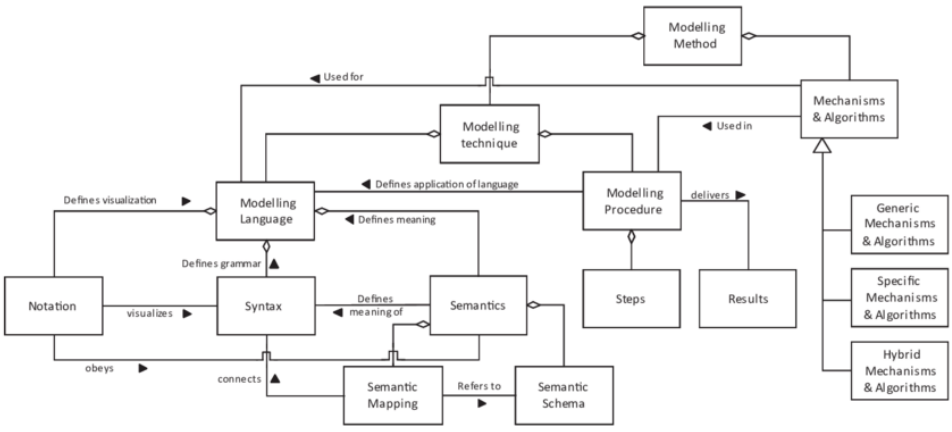


FIGURE 13 GENERIC MODELLING METHOD FRAMEWORK [1]

In the following, the components of the framework are introduced and mapped to the characteristics of the marketplace design methodology:

1. Modelling Language: the modelling language defines the notation, syntax and semantics of the modelling constructs as required for the domain-specific instantiation. The constructs and elements are defined in the form of a metamodel (following a language-oriented understanding of the modelling language as discussed by Strahringer in [20]. For the marketplace metamodel, the definition of constructs (classes and relations) and their attributes is required.
2. Mechanisms and Algorithms: building on the metamodel, mechanisms and algorithms provide processing capabilities of the modelling method. These capabilities utilize the metamodel (or even

the meta-meta model) as a semantic rich structure and provide value to the user. For the marketplace model in C2T, the processing capabilities include:

   a. Type generation: identify marketplace items according to defined axioms (maturity assessment),
   b. View generation: graphical view on the items and their interrelations,
   c. Model transformation: transformation of a marketplace model into a format that is understood by the landing page system, middleware components and infrastructure in a bidirectional manner,
   d. Composition design and assessment: defined as bundles of atomic services/offerings aggregated in a meaningful manner for the application in pilots and application scenarios.

3. Modelling Procedure: understood as the organizational process of how the modelling method is used (steps) and outcomes are produced (results). The procedure considers input/output relations whereas the steps are supported by functionalities as mechanisms and algorithms.

As outlined in the introductory section, we consider the conceptualization of the modelling method as an agile process that is informed by its application and use. In order to guide the process from concept [21], [22] design to implementation and deployment the "Agile Modelling Method Engineering (AMME)" approach is followed. AMME establishes the procedure and design outcomes required for each phase in the realization process, following an iterative and agile adaptation technique. This implies that the later phases in the process influence the design artefacts and therefore continuously adapt to the concretization of requirements. As design artefacts of this process, in the following the results are presented as

a) Conceptual Metamodel: identifying and mapping the requirements to an abstract representation using concepts, characteristics and connectors needed to support the intended purpose of the modelling approach,
b) Platform Independent Metamodel: applying the UML notation to derive a platform, programming language independent representation of the metamodel,
c) Platform Specific Metamodel: transforming the platform-independent view using the ADOxx metamodeling platform [23] for the implementation, applying inheritance and scripting techniques.

4.2 META MODEL OF MARKETPLACE MODEL

The conceptual metamodel is derived based on the observation and technical feasibility study performed during the initial release of the marketplace. The design approach selected is based on metamodel building blocks that constitute structural and functional bundles that operate on an abstract structure and are specialized during the design process.

The following high-level requirements are summarized below as input for the design of the marketplace metamodel.

a. Classification of “Marketplace items”: items are defined in a first hierarchical level as “Innovation items” and “Offerings”; this implies that common characteristics are assigned to the top-level element “Marketplace item” applicable for both concrete manifestation of innovation and offering constructs respectively.
b. Characteristics of marketplace items: targeting the presentation aspect of the concept based on a structured definition of properties and attributes. A typing technique is applied to classify items by their technical nature and provisioning scheme.
c. Ontological annotation: each item is annotated using the marketplace ontology developed in WP1. The design decision to use annotation is based on the evolutionary nature of the ontology and provides means to adapt the knowledge representation according to changing contextual information. The purpose of ontological annotation is related to user interaction (browsing/searching and advanced integration in assessment methods through discovery and retrieval mechanisms).
d. Context enrichment: the context of the item is established through its use and bridges the generic identification of the item with its instantiations. Similar as for ontological annotation, the context assigned (m:n relation between item and context), describes mechanisms to define environmental requirements and constraints in a generic manner.
e. Model processing: as mechanisms and algorithms to support the use of the model for analytical and representational purposes. The following functionalities are defined and operate on metamodel level:
    a. Querying/Filtering: basic querying mechanisms based on the metamodel definition (classification, properties/attributes); advanced query mechanisms based on matching techniques using ontological representation formats.
    b. View generation: generate model representation based on the model content (matrix views, dependency views, timelines) as a visualization of the model repository.
    c. Model transformation (Push): relevant for model-to-model transformation, whereas the marketplace metamodel is used to map rules sources for the transformation and apply graph-rewriting techniques. The push mechanisms is required to transform the model into a serialization form for visualization.
    d. Model retrieval (Pull): open interfaces to retrieve information from the model and its objects. The pull mechanism is required for the deployment of items in the marketplace middleware and configuring the pricing models.

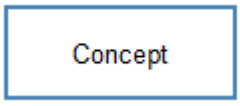
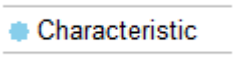
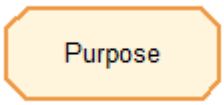
In the following the conceptual metamodel is represented using the CoChaCo approach. The resulting metamodel describes the concepts, characteristics and connectors of the marketplace model and is used as an input for the platform independent metamodel specification.

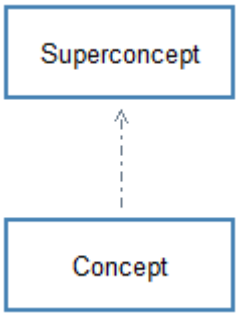
4.3 METAMODEL DESIGN USING COCHACO

The design process of the conceptual meta-model has been performed in an iterative manner, focusing on the structural elements initially (as observed in past projects) and elevating the recognized concepts with the above characteristics and their purpose.

The conceptual meta-model is discussed in the following using the CoChaCo (Concept-Characteristics-Connectors) metamodel design environment introduced in [22]. As proposed in [22], a metamodeling approach is applied to design the marketplace metamodel using generic constructs to identify the concepts (as structural elements), characteristics as well as relations between concepts. The extended view in CoChaCo is applied as the processing aspects are considered an important aspect of the marketplace metamodel. Required functionality is identified as the purpose for specific concepts and related to characteristics and their definition. During this phase of metamodel design, the scope is on conceptual level, which implies that technical aspects (such as realization techniques, data types, etc.) are not considered during the specification of the platform independent design. The used concepts from CoChaCo are identified and defined below (as a subset of the MM-DSL language). The design results are accessible as a graphical model in ADL (ADOxx Definition Language) format that can be imported into the MM-DSL toolkit openly available at https://www.omilab.org/activities/cochaco.html.

TABLE 1 COCHACO DESIGN ELEMENTS

CoChaCo Concept	Visualization/Notation	Definition
Concept		Identify concepts and their contextual specification
Characteristic		Identify characteristics of a concept or connector in a hierarchical manger
Connector		Identify relations between concepts
Purpose		Abstract definition of the purpose of a concept/characteristic/connector

specializes		Relation between concepts, connectors and characteristics Applicable between same type constructs Semantic: “is-a” Functionality: “inheritance”
has		Relation between Concepts/Characteristics and Connectors and Characteristics Semantic: syntax of construct
uses		Relation to define conceptual requirements for a model processing purpose Semantic: association

4.4 CONCEPTUAL DESIGN RESULTS: MARKETPLACE MODEL

In the following the conceptual design results for the marketplace model are discussed. The findings relate to the requirements in chapter 3, systematically analysed with respect to the purpose of specific concepts. A design based on decomposition has been selected to provide a high-level entry point and detailed representations for sub-elements.

Figure 14 presents the high-level design of the marketplace model.

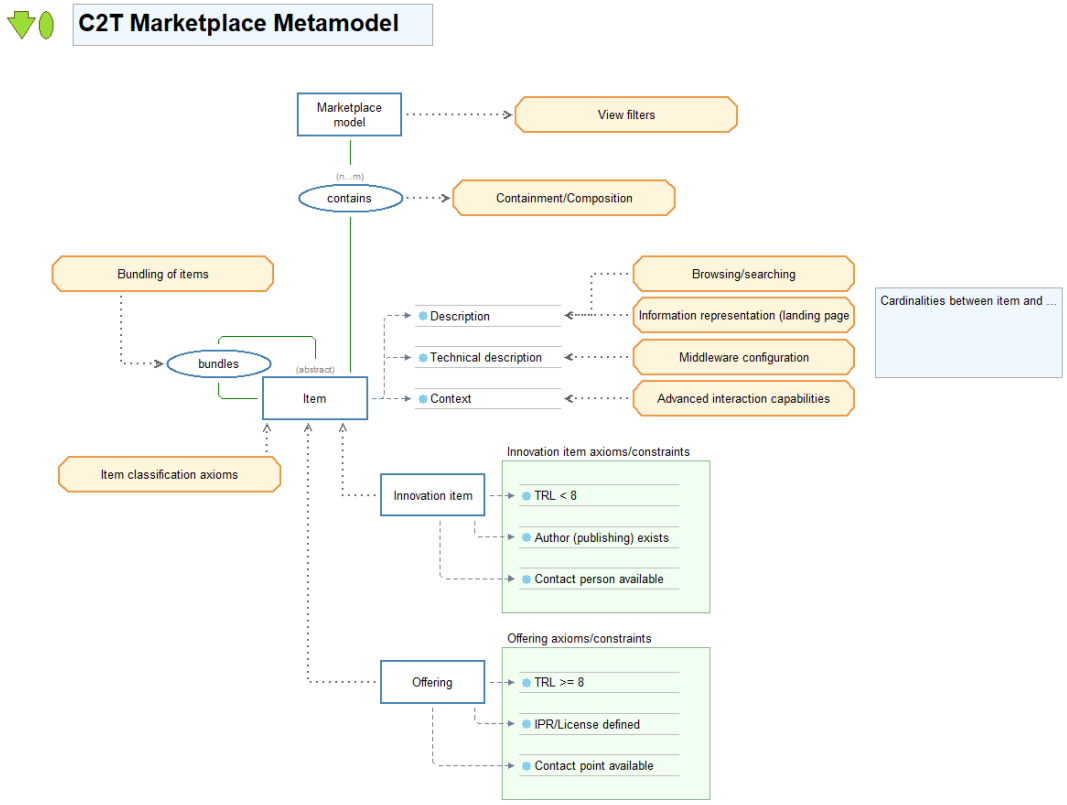


FIGURE 14 CONCEPTUAL DESIGN: MARKETPLACE MODEL (HIGH-LEVEL)

Marketplace Model: The concept "Marketplace model" is considered as a container that collects a sub-set of available "Items" ("contains" connector). The cardinality between the container and contained element is defined as "m:n, which implies that an item can be contained in multiple containers based on the modelling interaction of the marketplace modeler. The connector "contains" is required to provide means for containment analysis ("in which marketplaces is a specific item used/defined") and establishes the possibility to filter and assess a concrete marketplace model instance (filtering, searching within the model).

Item: An "Item" is considered as the building element/block of the marketplace model. It is defined in abstract terms, aggregating common characteristics of any item type. Following metamodeling techniques, concrete item types are specialization of the abstract concept. In the high-level design, an "Item" is characterized by three dimensions:

- Description: summarizes descriptive facets of an item that are used for visualization and representation. In addition, free form facets are applicable for full-text search and structure characteristics (such as selection fields) are used for categorization functionality (e.g. matrix view generation, filtering).
- Technical description: summarizes the technical facets per item required for deployment and operation of an item via the middleware. As the model acts as the configuration environment, the middleware

requests information from the model to configure e.g. the interaction with the user (checkout process), pricing model, or deployment options.

- Context: summarizes the contextual annotation of an item using ontological descriptions. These descriptions are defined as dynamic attributes that provide advanced querying and reasoning functionality using the concrete model as an input.

Items are specialized and concretized as sub-concepts, whereas a differentiation is proposed based on the technological readiness level (TRL) of an item.

1. "Innovation item": derived from the abstract "Item", an "Innovation item" is defined by a) a TRL lower than 8, b) availability of an Author, and c) Contact to the item owner can be established. As such, an innovation item is provided through direct interaction between a potential user and the item provider.
2. "Offering": derived from the abstract "Item", an "Offering" is defined by a) a TRL level higher or equal to 8, b) availability of license terms/IPR issues clarified, and c) contact point, rather than a contact person to provide an institutional interaction between user and provider.

The differentiation as discussed above is conceptualized as axioms dynamically determined as constraints and specialization of their characteristics. A detailed view on the descriptive characteristics is provided in Figure 15. Each descriptive facet is introduced conceptually and defined in the following table.

4.4.1 Marketplace Item: Descriptive Characteristics

Figure 15 provides a graphical view on the descriptive dimension of a marketplace item. These descriptive characteristics are applicable for both offerings and innovation items. Constraints developed classify the item.

Table 2 provide a definition of these characteristics and their specific relation to the purpose within the conceptual model.

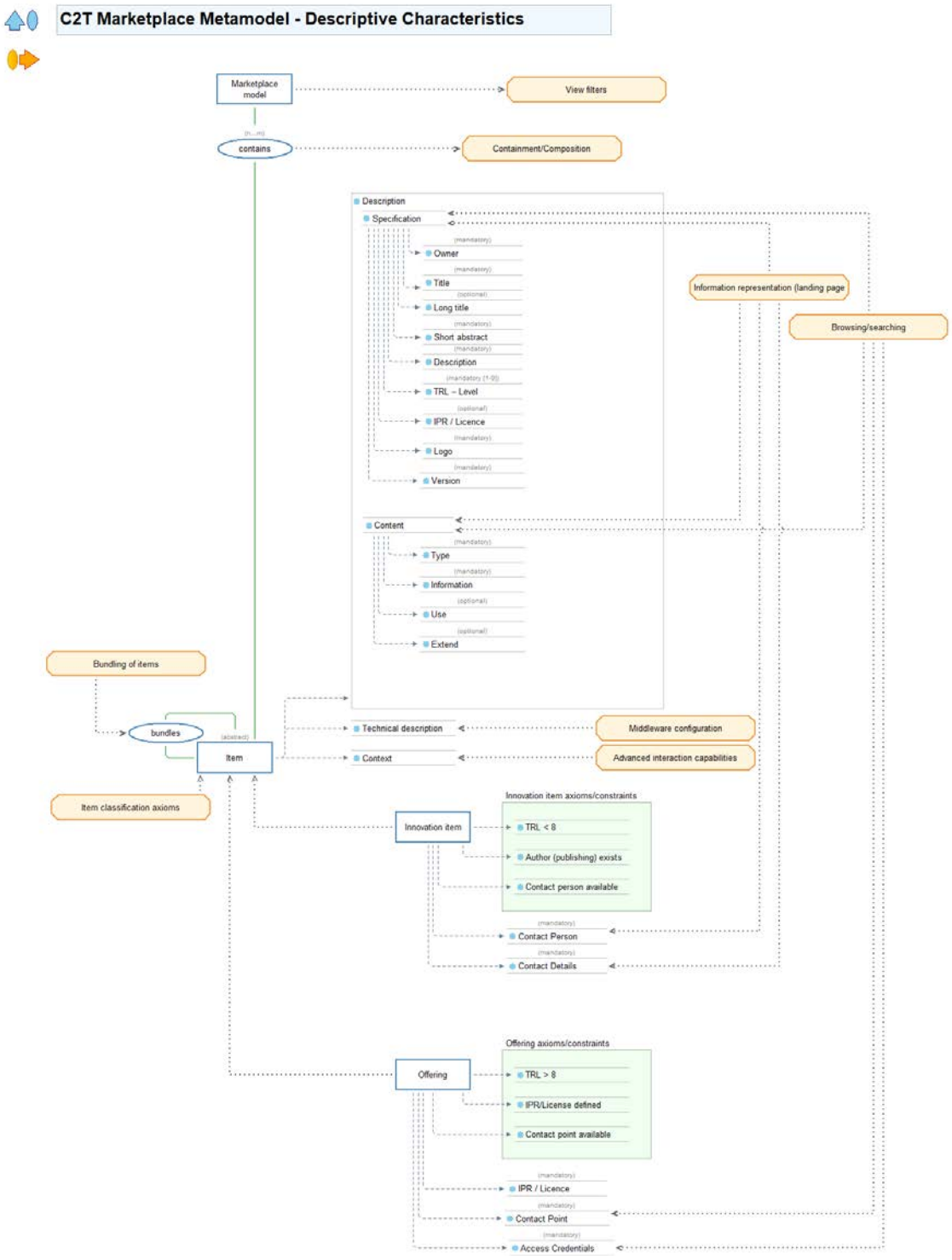


FIGURE 15 CONCEPTUAL DESIGN: MARKETPLACE MODEL, DESCRIPTIVE FACETS

In Table 2, the definition of descriptive characteristics for items is introduced. A distinction is made between specification and content representation.

- Specification: any descriptive facet required to identify the item and provide structure information about the item's representation,

- Content: any descriptive facets, in the form of free markup text to represent the items content (including multimedia elements).

Both categories are applicable for the representation of an item on the landing page (through model transformation), whereas the facets in the specification dimension provide input for assessment queries and view generation.

TABLE 2 ITEM: DESCRIPTIVE CHARACTERISTICS

Characteristic name	**Necessity**	**Definition**
Owner	mandatory	Responsibility (natural, legal person) Defines the organisational context
Title	mandatory	Header (for identification) Used as the header and title of the item, specifically on the landing page
Long title	optional	For mouse over, help text, second line title Used to describe the item in detail, specifically for user interaction
Short abstract	mandatory	Preview of item to gain awareness Short description of the item, as a teaser
Description	mandatory	Description to clarify what the user can expect Long description of the item to create insight in the offering
TRL – Level	mandatory [1-9]	Description of the technical readiness level Classification of the technical readiness, evaluated in constraints for item classification [offering, innovation item] [24]
IPR / Licence	optional mandatory for offerings.	License owner and license type (who has the right / licenses / is first author / first reference) License defined for the item to clarify how the item can be used
Contact Person (specialization for innovation items)	mandatory	Contact person Specifically for innovation items: organizational context who to contact
Contact Details (specialization for innovation items)	mandatory	Email address of contact person Specifically for innovation items: organizational context who to contact
Contact Point (specialization for offerings)	mandatory	Whom to contact if I want to buy / use it Specifically for offerings: organizational context who to contact in case a purchase is established
Access Credentials (specialization for offerings)	mandatory	How to contact if I want to buy / use it Specifically for offerings items: organizational context who to contact
Logo	mandatory	Icon and / or image to be displayed for the item Graphical, iconic representation of the item
Version	mandatory	Lifecycle of the item version, phases Version number, trajectory of releases
Type	mandatory	Type of the item: - Consulting Offering - Software Offering - Hardware Offering - Solution Offering

		Classification of the item type, impacts the deployment in the middleware
Information	mandatory	Collection of References to documents / website, webpages, videos, photos, data (sets) Hypertext including multimedia content to provide information and resources for the item
Use	optional	Reference of Demonstration, Trial / Demo Versions Hypertext to showcase the use of the item, e.g. demos or trials
Extend	optional	Reference to Open-Source Platforms {GitLab, GitHub}, Communities, Training and Academy Programmes Hypertext describing the extension capabilities (open-source items)

4.4.2 Marketplace Item: Technical Characteristics

This chapter defines the technical characteristics of an item. These characteristics have a relation to the type and influence the way the middleware provides the service to the user/consumer. Figure 16 provides a graphical representation of the dimension, detailed in Table 4.

The concept developed builds on the "Type" classification of an innovation item or offering. The type characteristic defines the technical viewpoint as input for the middleware to retrieve the configuration from the model and provide the deployable, digitized service.

The following types are investigated and detailed below:

TABLE 3 ITEM TYPES

Type	**Definition**	**Purpose**
Consulting	Required characteristic to digitize a consulting service. The contact point and contact person respectively from the descriptive dimension are repurposed.	The middleware retrieves and deploys such services to establish the contact and provide interaction capabilities between consumer and provider.
Information	Information (externalised knowledge) is provided through services. The artefacts are provided via the middleware, according to interaction streams supported (e.g. download link, email, access details provisioning).	The middleware provides access to the information artefact.
Software	Three types of software artefacts are currently considered: a. Download link: similar as for type "Information", the link to the software artefact is provided,	The middleware is, based in the classification and provided characteristic, responsible to establish the deployment (from providing a download link to configuring a tenant).

	b. Image deployment: using e.g. Docker images to deploy an own stack of SaaS on infrastructure level via the middleware, c. Tenant deployment: for systems that are not deployed in the infrastructure but an own tenant within a running deployment is provisioned.	
Hardware	Provision of hardware (and potentially) corresponding software solution.	Currently considered similar as a consulting service where contact information is exchanged using middleware capabilities.
Solution	As a combination of types to support a specific solution required. Realized as bundles.	Reflect bundles and provision them accordingly, also considering the context of the solution offering.

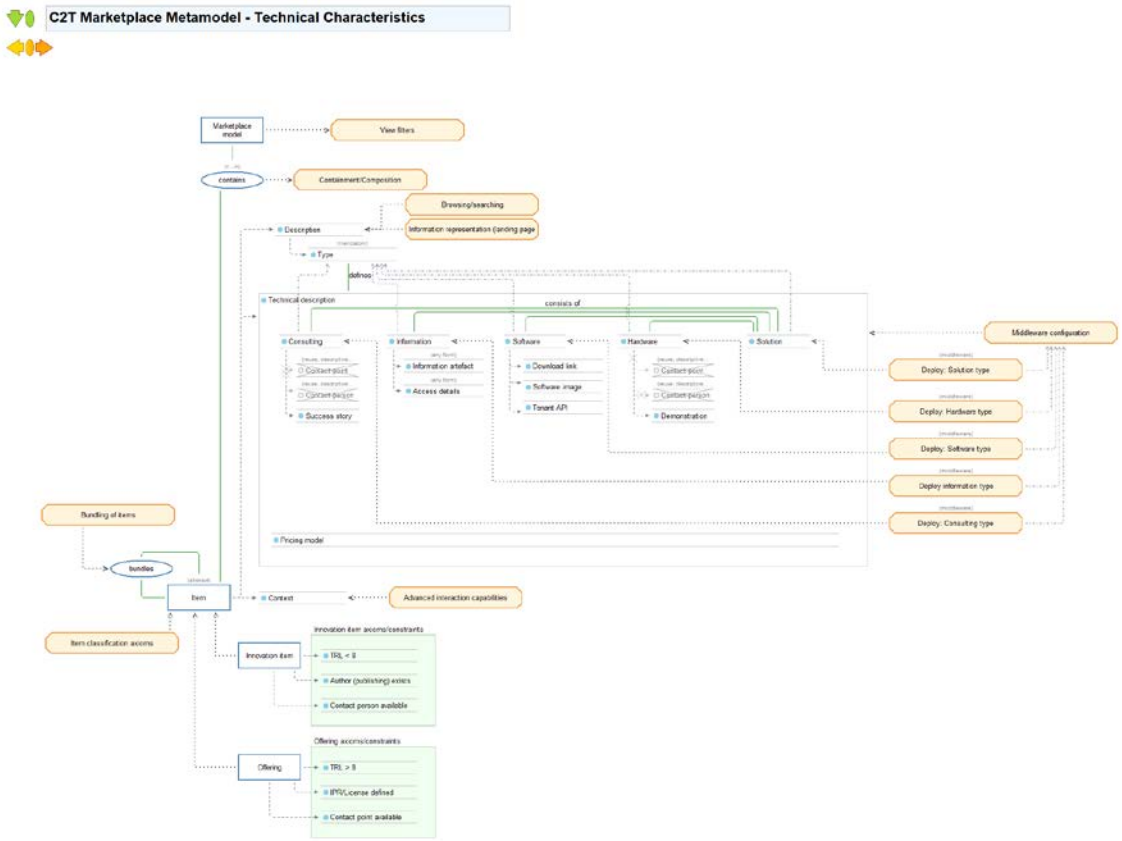


FIGURE 16 CONCEPTUAL DESIGN: MARKETPLACE MODEL, TECHNICAL FACETS

In the following table the type-specific technical characteristics are detailed. The technical characteristics are marked as mandatory, in relation and applicable for the specific type as introduced in Table 3.

TABLE 4 ITEM: TECHNICAL CHARACTERISTICS

Characteristic name	**Necessity**	**Definition**
Contact point Contact person (re-use from descriptive facet)	mandatory applicable for type “Consulting” and “Hardware”	Organisation context of the item, consisting of name and contact details. Required to establish the interaction
Information artefact	mandatory applicable for type “Information”	List of documents/resources (links, digital resources) Provided as after purchase
Access details	mandatory applicable for type “Information”	Can be none, details to access/open/use the information artefacts
Download link	mandatory applicable for type “Software”	Three alternative options are provided to retrieve a software artefact - Download: self-installation - Software image: container-based deployment by the middleware - Tenant: configuration of access via API Access to software artefact type
Software image	mandatory applicable for type “Software”	
Tenant API	mandatory applicable for type “Software”	
Pricing model	optional, no pricing model implies free of charge interaction	Definition of the pricing model applied for specific items/offerings Depends on the item characteristic and business model of the provider
Interaction model	optional	Definition of the interaction scheme the middleware supports Depends on capabilities of interaction flows supported by the middleware

4.4.3 Marketplace Item: Context Characteristics

The context defines the use of the item in an application. The characteristics is therefore defined as a many to many relation, meaning that one or more context definitions are potentially related to one or more item elements. The purpose of the context is to clarify the behaviour of an item in a concrete case and therefore elevate the semantic description of it. Modelling the context allows to retrieve further information on items and provide advanced querying, discovery and matching capabilities. This is specifically relevant for assessment methods to further extend the item description dynamically without changing the structural metamodel defined for the design.

An annotation approach is suggested utilizing semantic lifting techniques for conceptual artefacts as discussed in [25]. Technically, RDF is used for the annotation mechanism, whereas an item is understood as an RDF resource that is described using multiple RDF statements.

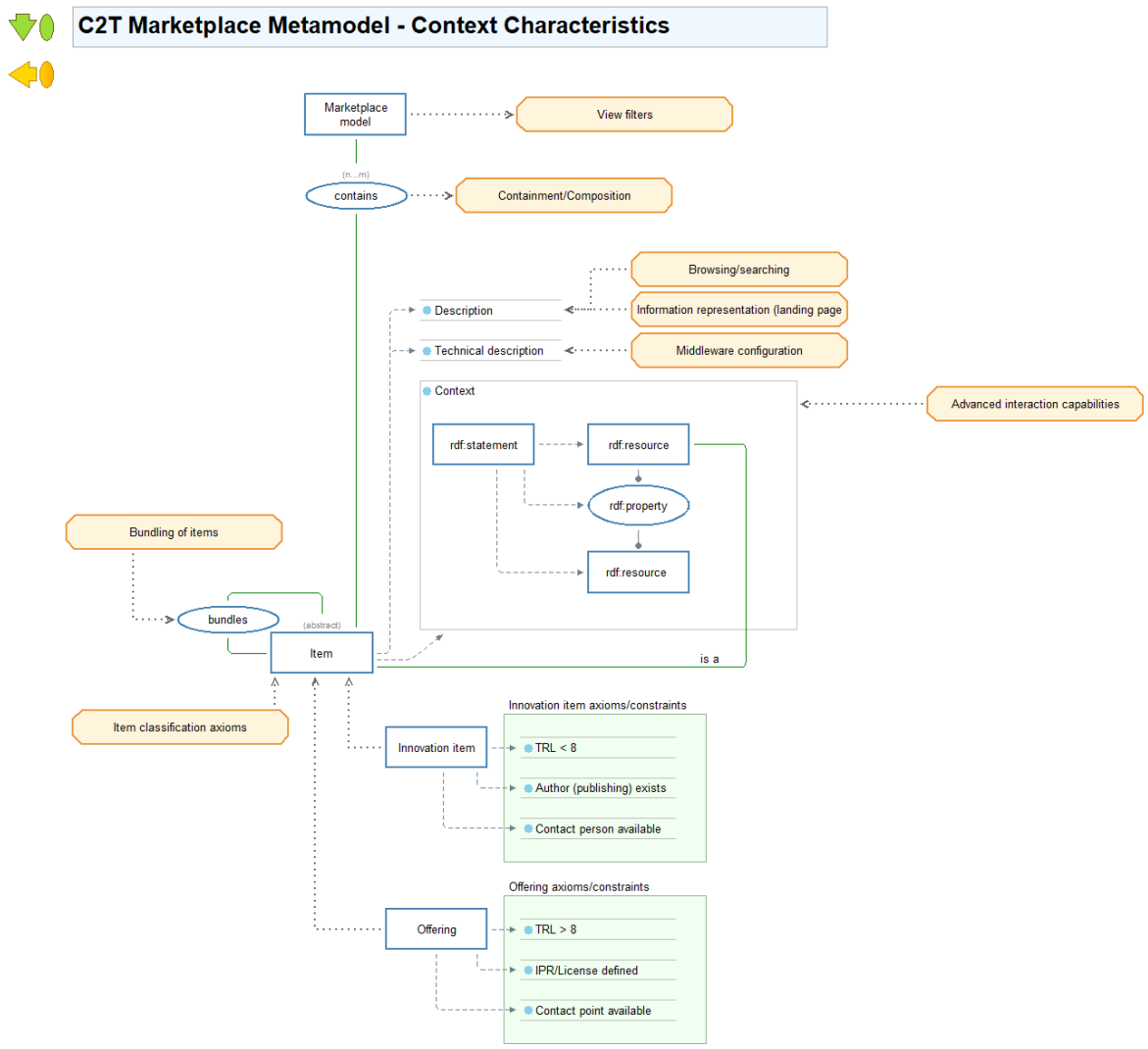


FIGURE 17 CONCEPTUAL DESIGN: MARKETPLACE MODEL, CONTEXT FACETS

Figure 17 shows the logic of context annotations using RDF as a technical format. The decision for RDF is based on the lightweight nature of the knowledge representation standard and its flexibility with respect to extensions required. Context is therefore defined as “use” within a specific “environment” for a given “purpose”.

TABLE 5 ITEM: CONTEXT CHARACTERISTICS

Name	Necessity	Definition
RDF statement	optional	An RDF statement is annotating the item whereas the RDF resource is defined as the item itself and the RDF property defines the relation to the annotation context as an RDF resource.
RDF Property	mandatory, for the definition of an RDF statement	Meaning assigned to the property. Established by the vocabulary applied (for the initial version), RDF and RDFS vocabulary is proposed (rdf:type, rdfs:label, rdfs:comment rdfs:range, rdfs:domain, rdfs:subClassOf, rdfs:subPropertyOf).
RDF Resource	mandatory, for the definition of an RDF statement	Annotation value used within the statement.

The above conceptual metamodel is transformed into a platform independent representation as a blueprint for implementation.

4.5 PLATFORM INDEPENDENT META MODEL OF MARKETPLACE MODEL

The platform independent model is described using the UML class diagram [26] notation as an architectural blueprint for implementation. The approach has been evaluated in [27] for its applicability during the design of metamodels.

The aggregating element in the architecture is the marketplace model that provides derived attributes based on the aggregation relation. As such the marketplace model acts as a container and provides the required operations to query and transform the contained elements. The following operations are specified:

- transformToMarkdown: as the model transformation functionality to re-write the conceptual model to the serialization format Markdown. The outcome is a set of files representing the contained items in Markdown syntax. The transformation rules are defined in a transformation mapping.
- retrieveItemByType: query capabilities to retrieve items by type (dynamic type recognition), internally building on the “queryByStructure” operation.
- retrieveItemByTRL: query capabilities to retrieve items by TRL, acts as a filter on the contained items, internally building on the “queryByStructure” operation.
- queryByStructure: generic query operation to retrieve items based on description and technical facets (evaluating the facets), considers the metamodel structure.

- queryBySemantic: query operation to retrieve items based on the contextual annotation, building on the semantics of an item annotation.

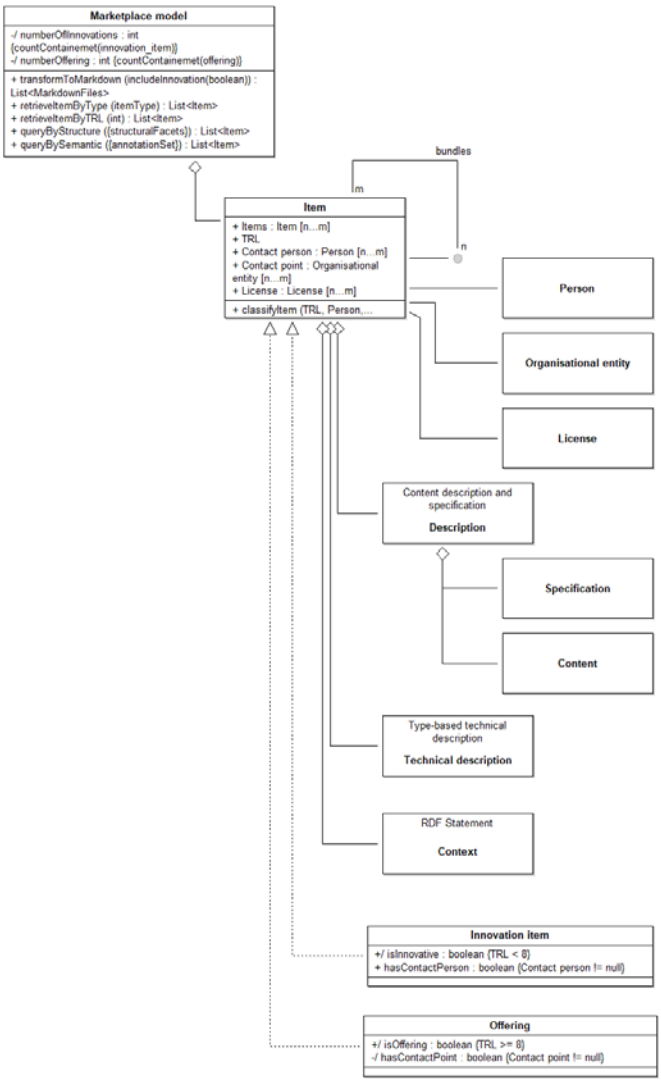


FIGURE 18 PLATFORM INDEPENDENT METAMODEL: MARKETPLACE MODEL

The item represents an offering or innovation item and follows a common structure for definition. The constraints defined act on the type recognition functionality of an item. An item aggregates the description, technical and context properties. Based on the design, this aggregation is considered as a many to many association.

5 IMPLEMENTATION OF THE MARKETPLACE META MODEL

The blueprint presented above has been implemented as a proof-of-concept implementation. The implementation has been performed using the ADOxx metamodeling platform, as the environment provides possibilities to re-use and configure existing platform functionalities. The platform specific metamodel is realized using

a) Inheritance techniques: building on the abstract metamodel in ADOxx, the concrete classes are derived and inherit the behaviour and functionalities of the abstract metamodel, the outcome is a metamodel in the domain-specific language ALL (ADOxx Library Language),
b) Scripting: the processing logic is implemented using the AdoScript language based on the ALL implementation of the metamodel. This means that functionality is either derived (from existing platform capabilities), configured (as mappings to the abstract metamodel and inheritance) or scripted individually.

In the following section, the platform specific metamodel is presented and discussed, followed by a presentation of the prototype results.

5.1 PLATFORM SPECIFIC META MODEL FOR ADOXX

The relevant aspects of the implementation are presented in the following, focusing on the class hierarchy of modelling construct and relation classes initially, then presenting code snippets for the introduced processing functionality. This approach is informed by the work in [28] and [29].

The following design decisions have been taken for the initial prototype of the tool:

- Condensed metamodel: the type logic is expressed through enumerations and specific enumeration domains,
- Attribute assignment: the item represents the full set of properties and can be specialized,
- Containment and composition: the composition logic controls the query and export functionalities,
- Processing capabilities are established through embedded scripts (using expressions e.g. type recognition based on properties) and scripts that are triggered on interaction (event-based processing).

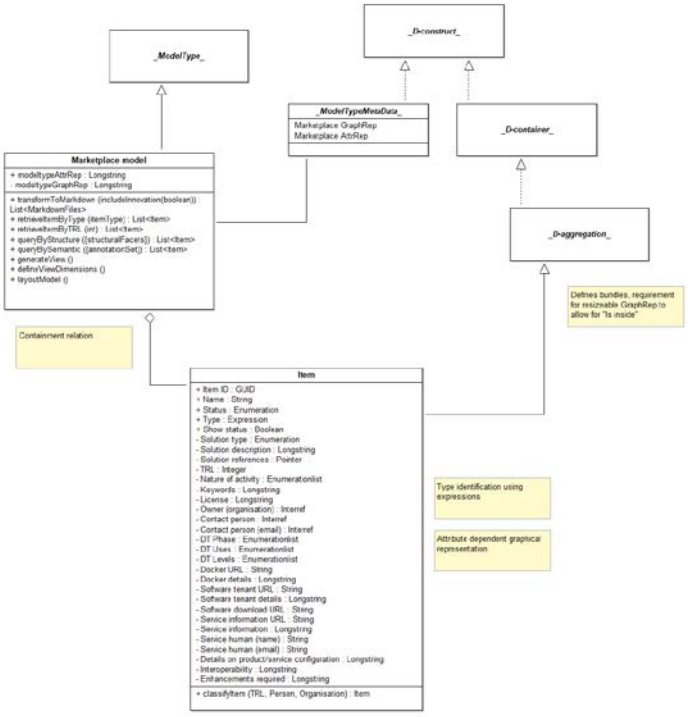


FIGURE 19 ADOXX PLATFORM SPECIFIC METAMODEL

Figure 19 shows graphically the platform specific metamodel including the abstract metamodel elements used during implementation.

5.2 DOCUMENTATION OF MODELLING TOOL IMPLEMENTATION

In the following subsection the implementation results of the above platform-specific metamodel are presented. An example view (of collected items during the project) and extended with results from previous project results is shown in the below figure in FIGURE 20.

The characteristics of the tool prototype are:

- Graphical modelling tool: graphical notation is applied to represent marketplace models and items, whereas the attributes of an item influence their appearance.
- Formal representation: the model is established formally (based on the metamodel) and allows to apply processing functionality (analysis, view update, transformation), supported by the platform and functionality through the serialisation to a knowledge graph (RDF representation)
- Model repository: different model variants and versions can be maintained in a single database and the user can create various marketplace structures to support subsetting during assessment workshops
- Advanced queries: through context definition and discovery mechanisms

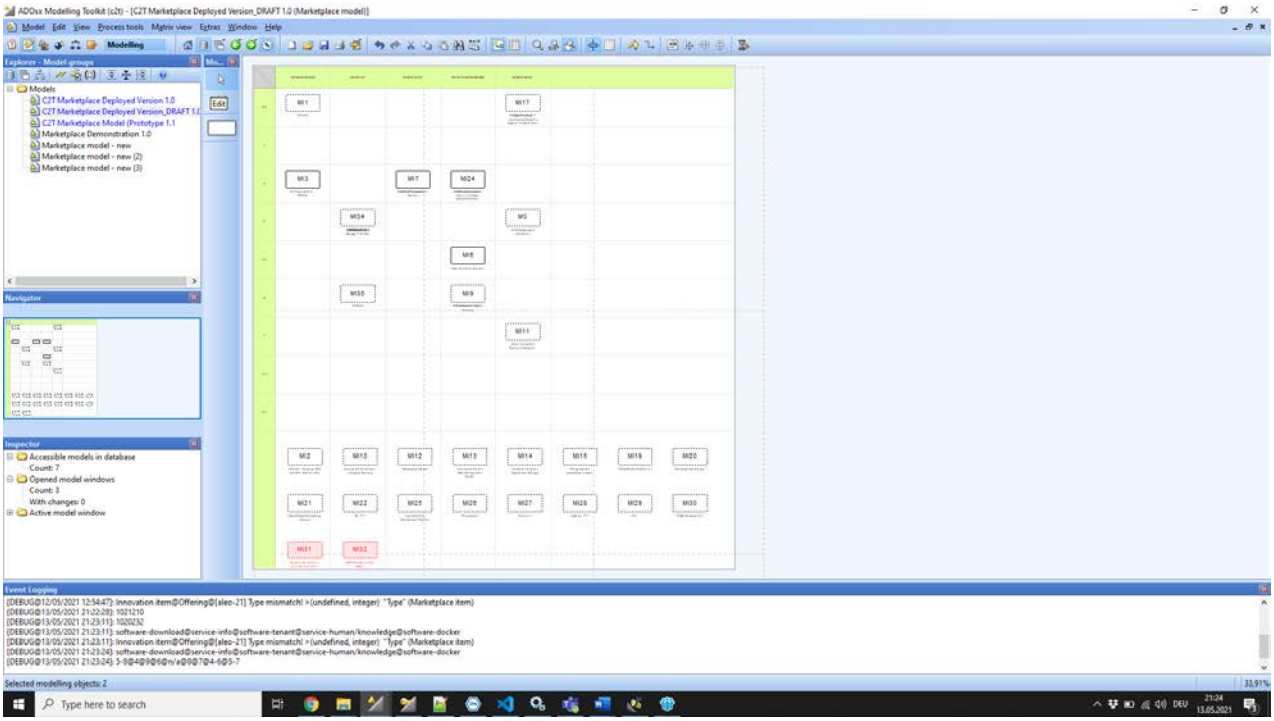


FIGURE 20 CHANGE2TWIN MARKETPLACE MODELLING TOOL

From a technical viewpoint, the implementation results are presented in the following subsections.

5.2.1 Marketplace Model

Derived from ADOxx construct: MODELTYPE, realizing the following requirements:

- Containment: platform functionality
- Graphical representation/modelling: platform functionality, and extension for a modeltype attribute notation and graphical representation (containment statistics and view generation)
- Print, search, table editor: platform functionality
- Model persistence: platform functionality
- Model query: platform functionality

Configuration result:

```
GENERAL
MODELTYPE "Marketplace model" plural:"Marketplace models" from:none attrrep:"Marketplace AttrRep" gra
phrep:"Marketplace GraphRep"
INCL "Marketplace item"
```

CODE 1 ADOXX IMPLEMENTATION: MODELTYPE CONFIGURATION

Implementation result modeltype: the graphical result of the modeltype is shown in FIGURE 21.

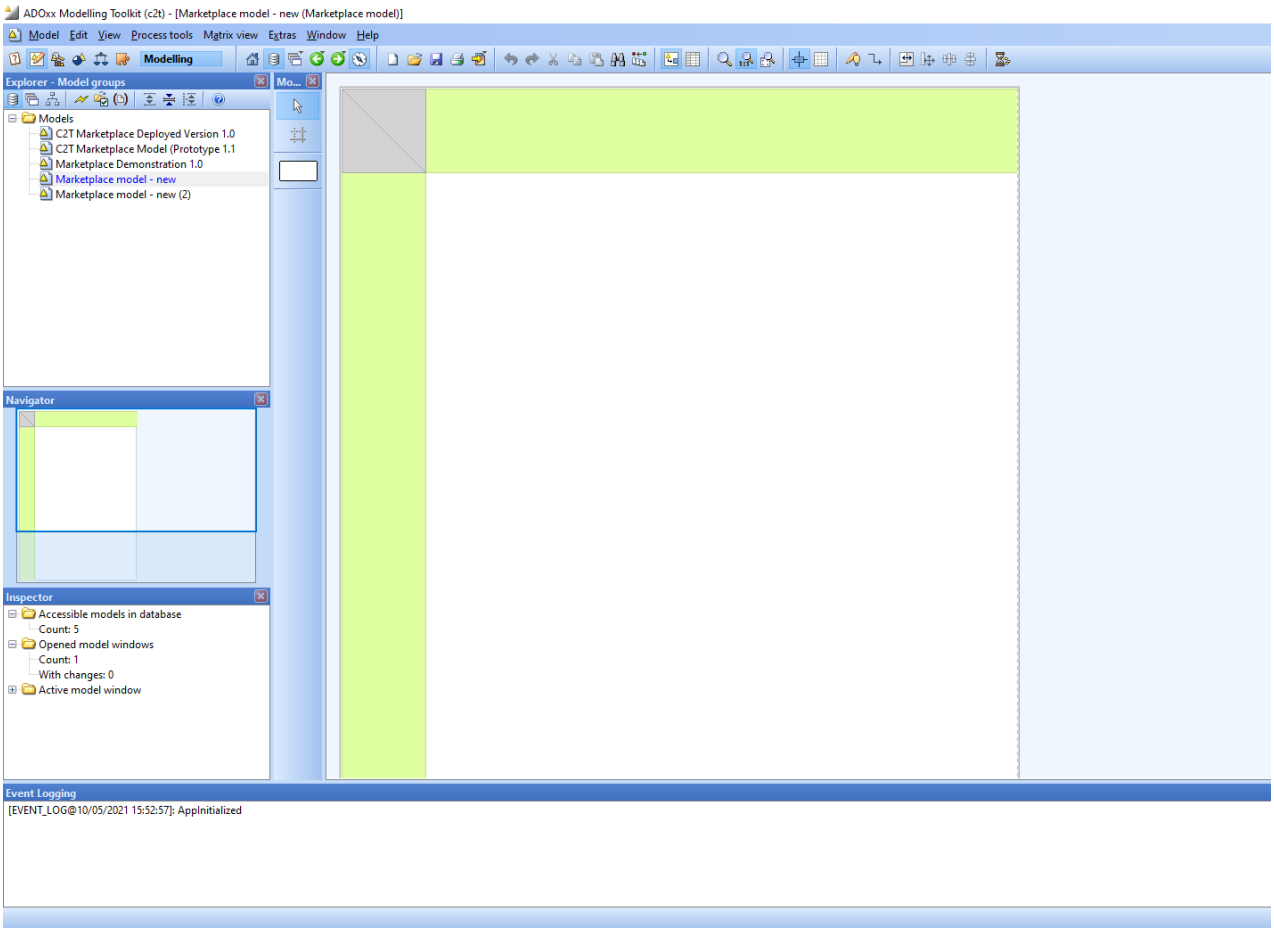


FIGURE 21 ADOXX IMPLEMENTATION: MARKETPLACE MODEL

Implementation result – view generation: The vertical and horizontal graphical borders enable a dynamic sorting of items according to user-defined criteria from the static metamodel. The functionality is triggered on the event “SaveModel”, where the user has the possibility to enter the vertical and horizontal dimensions and the items contained are automatically sorted and put into a 2-dimensional layout. (documented in the Annex as fragment Code 3).

Additional scripts are provided to reset the initial selection of view dimensions, based on user interaction (within the menu of the tool see FIGURE 22). These include:

- Definition of dimensions (vertical/horizontal)
- Retrieval of all possible dimensions (based on completeness of item description)
- Positioning: manual trigger of above-mentioned script to re-layout

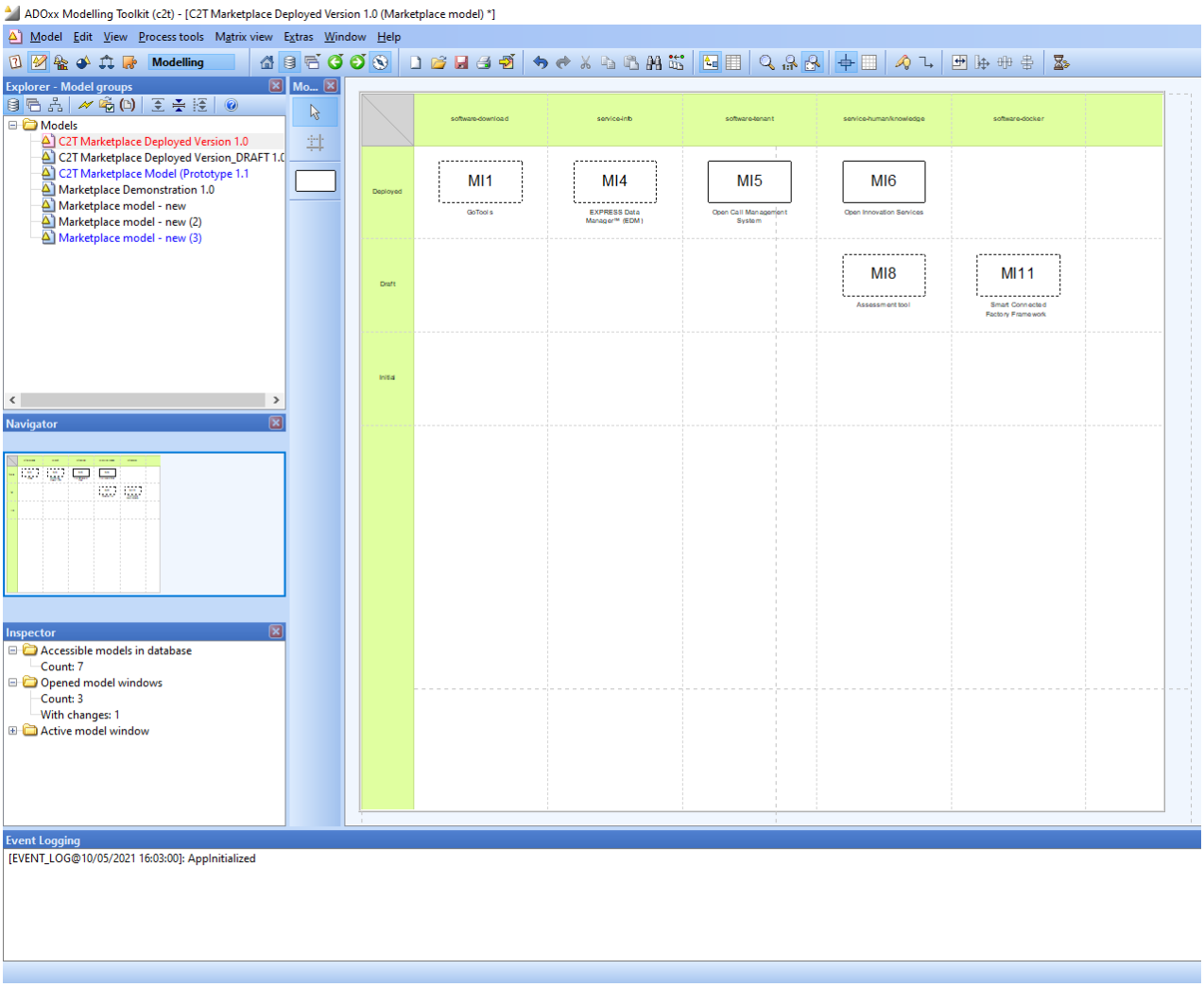


FIGURE 22 MARKETPLACE ITEM CLASSIFICATION

Implementation result import/export: the base functionality of ADOxx to generate arbitrary model representations/serializations is used to realize import and export mechanisms. The mechanism of model transformation to markdown is exemplified in fragment CODE 4 provided in the Annex.

This code snippet is triggered by the user and enables the export of Markdown files to a provided location. Filters can be selected and applied during the export.

5.2.2 Marketplace Item

Derived from ADOxx construct: _D-aggregation_, realizing the following requirements:

- Containment: bundle logic derived from _D-aggregation_ superclass (platform functionality to support “Is inside” relation between contained element), relevant for bundles

- Graphical representation: attribute-dependent visualization, the basic representation as shown below adapts to attribute changes
- Attributes/Properties: defines for the class, in accordance with the specification
- Automatic type recognition: using expression logic to implement the type constraints on item level

The type of an item is dynamically recognized using an ADOxx Expression. The basic expression below assesses the TRL level (CODE 2)

```
EXPR type:string expr:fixed:(
        cond(VAL aval("TRL") > 8,
           "Offering",
           "Innovation item"
        )
)
```

CODE 2 ADOXX IMPLEMENTATION: TYPE RECOGNITION BASED ON TRL

The graphical representation is attribute-dependent and updates according to the attributes and their values for instance (see the detailed fragment in CODE 5.

An example view of two items and their adaptation based on the type is shown in FIGURE 23.

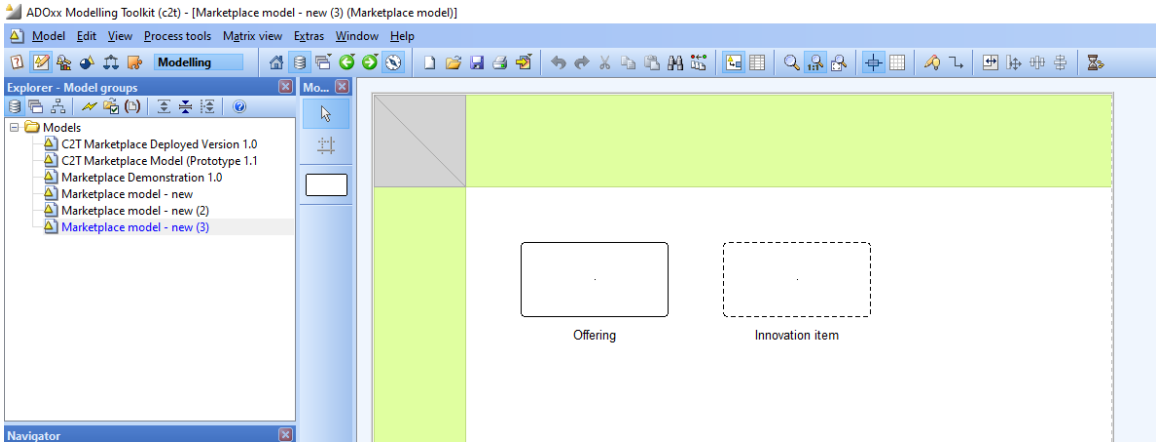


FIGURE 23 MARKETPLACE ITEM VISUALISATION

5.3 MARKETPLACE MODELLING TOOL BASED ON ADOXX

The prototype of the modelling tool is available as an open artefact via the ADOxx.org community page, more specifically the Change2Twin development space.

https://adoxx.org/live/web/change2twin/overview

The modelling tool implementation is provided as a standalone full installer, which means that after download, the tool can be installed on PCs and is usable for evaluation by any interesting stakeholder.

6 EVALUATION: MARKETPLACE AS AN ITEM

6.1 MARKETPLACE AS AN OFFERING

We consider the software package of the marketplace also as an interesting marketplace item, which we aim to distribute to Digital Innovation Hubs and Associations with the goal to achieve sustainability. Therefore we propose three different offerings with respect to the marketplace itself: ready-to-use, ready-to-install and ready-to-adapt.

6.1.1 "Ready to Use" Marketplace Offering

A "ready to use" marketplace, which is mainly provided by PSNC, by offering the IT infrastructure and the corresponding software packages – the middleware and the landing page – in such a form that the user can select, deploy and use the marketplace. The Digital Innovation Hub would be the customer of the marketplace item – in order to become a broker of marketplace items, which we do not know, but which are entirely in the atmosphere of the consuming Digital Innovation Hub.

In this model, the marketplace is offered as-a-service, working out-of-the-box, where potential customers, e.g. broker DIHs, use already available solution without bothering with software package installation or deployment. DIHs have the possibility to monitor the hardware resource usage, but also to administer the content of the marketplace, i.e. the offerings, and registered users. Still, the marketplace look & feel can be adapted by the owner of this service.

The underlying IT infrastructure is primarily used for running the ready-to-use marketplace. However, it may be used for hosting deployable services offered via this marketplace as well, making the administration and monitoring of their usage simpler.

In order to utilize full features, such as automatic deployment of the deployable offerings, support full monitoring and billing capabilities, this ready-to-use marketplace may be coupled with the CB middleware. In this case, PSNC infrastructure acts additionally as a hosting environment for deployable services, providing extended monitoring capabilities to the middleware for billing purposes. However, other infrastructure can be still used for automatic deployment of the items being offered through the marketplace.

6.1.2 "Ready to Install" Marketplace Offering

A "ready to install" marketplace, which is mainly provided by CB by offering the various software packages, which can be downloaded as a deployable Docker image. In that case our customer – e.g. a Digital Innovation Hub – is not using the IT-infrastructure of PSNC, but is taking responsible to find an appropriate infrastructure and deploy the Docker image with the corresponding parameters. Similar to the above scenario,

the Digital Innovation Hub would be a customer and the actual items on that marketplace are unknown to us, who only provide the sw-packages to operate such a marketplace.

6.1.3 "Ready to Adapt" Marketplace Offering

A "*ready to adapt*" marketplace, which is mainly provided as open-source package and the corresponding documentation that explain installation, running and enables the extension, improvement and adaption of the provided software package for the particular need of the customer. The landing page and the marketplace model are provided in open formats, so that potentially any middleware can be added to run on potentially any IT infrastructure. We consider this package as utmost important to enable sustainability, hence beside the provision of the software as open-source, we dedicated a task at the end of the project that deals with the hand-over of this item to a sustainable community. Hence, this item may change during the project duration, when the hand-over procedure is more concrete at the end of the project.

The marketplace ready-to-adapt flavour, also called marketplace delivery package, will provide (beyond required software components) documentation, training materials, certification workshops etc. It is a continuous development effort to address DIH/customers/providers comments and requirements. This solution is based on open-source solutions to avoid any vendor lock-in. However, possible integrations with commercial tools (e.g. CB middleware) and deployment infrastructure (e.g. PSNC) will be explained to ease marketplace adaptation. Moreover, the package will list other open-source and commercial tools/add-ons that may be used with the marketplace. Some of the potential add-ons, planned to be investigated/developed within the project, are discussed in next subsection.

The delivery package will be available for download via different platforms, including Zenodo, GitHub or PSNC FIWARE lab node.

6.2 POTENTIAL MARKETPLACE ADD-ONS

Add-ons to the C2T Marketplace result in additional items in the marketplace and elevate the functionalities and user interaction capabilities. Some examples of this approach are presented in the following.

6.2.1 Search Engine

The marketplace can be extended by advanced search mechanisms. The default behaviour is a hierarchical structuring of items and allows for browsing based on specific tags. It is possible (due to the openness of the technology selected), that the generator of the frontends utilizes open source or commercial search mechanisms. These mechanisms are embedded upon the generation result and provide either local (e.g. LUNR - https://lunrjs.com/ or SOLR https://solr.apache.org/) indices for full-text search or cloud-based mechanisms (e.g. Algolia – http://algolia.com/). Independent of the selected technology, the examples

mentioned above provide pre-configured techniques to extend the transformation of items into Markdown (landing page) representation and update the indices on the fly.

6.2.2 Onboarding Process

The onboarding is part of the governance process to add a new item to the marketplace. Our experience in adding internal project offerings and still increasing number of external enquiries proves that this is not a trivial task. The first issue to tackle is to well describe required information in order the item to be onboarded. A handholding meetings with providers are a possible solution to go through the offline questionnaire, explain and discuss required content, and get onboarded immediately (if that is the case). It can be realized in a form of consultancy item in the marketplace.

Another approach would be to provide an online questionnaire and then initiate discussion with interested party, if required. In either case, we envision a database of items waiting for registration is available to follow the current status of the onboarding (e.g. in process, waiting for reply, accepted or rejected, just to name few). Such add-on could be based on ticketing system like Trello or Jira, which automatically registers new item to be onboarded and keeps track on its current status. Once accepted, it may trigger automatic update of the marketplace offerings.

We believe that such add-on will be very helpful for operating the marketplace, although the governance process needs to be shaped separately on case-by-case basis.

6.2.3 Payment Service

Billing component will be responsible for generating invoices. Invoices will include the total amount, but the user will have access to detailed billing items. It is assumed that the invoices will be available in the user's profile. It should be possible to generate the .pdf version of the invoice. Invoices will be generated on a monthly basis. Invoices will be provided for organizations and for individual customers that do not belong to any organization.

The billing component will receive usage data from other components. The billing component will not be responsible for stopping the deployment if the user runs out of money or to forbid the user to run some items.

The billing component will provide different price models for different item types and their combinations. There will be a possibility to purchase items in a bundle. Deployments will presumably be charged per core runtime second or runtime second. All price models and ways to interact with payment would be carefully assess with project partners who are responsible for commercialization and business model aspects.

The billing component should track all the items the customer used and should include the following information in the billing items of the invoice: unit cost per one item that was used, the quantity of the units

used, the total amount per item, item name and/or short description, links to the marketplace items or to the generated results, starting and ending time (if applicable). The item units will depend on the item type. For example, the unit for consultation might be 1 hour, etc.

6.2.4 Cross-Layer Monitoring

Utilizing monitoring techniques in all layers of the marketplace, the perception of items, their use and context can be retrieved dynamically. This specifically targets the “user” concept that is an essential part of the “cart” provided by the middleware and influencing the infrastructure deployment. Nevertheless, information on the actual use, adequacy of offerings and user satisfaction builds upon the technical monitoring capabilities and merges these aspects with the model characteristics.

6.2.5 Modelling Environment

An extension of the marketplace item relates to the support of designing marketplace items (following the methodology described above). This includes a modelling toolkit that is capable to perform views based on structural and semantic queries as input for a semi-automatic model transformation (conceptual model/item to Markdown representation and static HTML content as frontends). The structural (descriptive and content) characteristics are considered by transformation rules. In addition, the transformation considers tags (based on enumeration domains defined) and provides filters dynamically.

As an outcome of this extension, the user has the chance to create marketplaces following a design-driven approach and provide (potentially unlimited) subsets of items for a specific purpose and need. This approach considers the domain-specific specialisation (according to the contextual analysis performed).

An extension of the marketplace item relates to the support of designing marketplace items (following the methodology described above. This includes a modelling toolkit that is capable to perform views based on structural and semantic queries as input for a semi-automatic model transformation (conceptual model/item to markdown representation and static HTML content as frontends). The structural (descriptive and content) characteristics are considered by the transformation rules. In addition, the transformation considers tags (based on enumeration domains defined) and provides filters dynamically.

7 SUMMARY AND OUTLOOK

Within this deliverable the design methodologies for the Change2Twin marketplace have been defined based on observed requirements and the vision to apply a knowledge representation technique in the form of conceptual models to support user interactions on business, technical and conceptual/contextual level.

The initial specification of the marketplace model has been performed from a conceptual viewpoint (concepts, characteristics and connectors) as input for the specification of a platform-independent metamodel. This metamodel has been used as input for the prototyping phase, resulting in a modelling tool that enables the conceptual design and specification of marketplace models.

7.1 USAGE SCENARIOS

The marketplace model establishes, in the initial version discussed in this deliverable, the base interaction functionalities to

a) Design items according to a common metamodel,
b) Structure and refine item descriptions,
c) Transform items in arbitrary visualisation and frontends, and
d) Provide an API (based in the model defined) for advanced, semantically rich queries.

Advanced usage scenarios are supported (through the formal representation, e.g. context-based search, similarity/discovery), but require a harmonization of the knowledge base for semantic lifting. The information on context and assessment method design is guiding the update process.

7.2 CURRENT PROTOTYPES

The current status of prototypes realized in the context of this deliverable are:

a) Modelling Toolkit: two iterations have been implemented demonstrating the visualisation and semantic search capabilities of a model-based design approach.
b) Model Transformation Mechanisms: using mapping rules defined in AdoScript based on the ADOxx platform, the graphical models are transformed into Markdown representation.
c) Landing Page Generator: the Markdown content is transformed into static HTML content, following the design guidelines of the C2T website.
d) Browsing and Searching Plugins: browse and search plugins that operate on annotation of enumeration values and full text are integrated in the transformation logic.
e) Deployment prototypes: following continuous integration and deployment techniques, the prototype is sensitive to changes of the model. This means that in case of a model change and release (trigger

of transformation), a re-build is triggered locally. As soon as the transformation results are confirmed by the modeller and committed, the marketplace deployment is rebuilt. As the baseline technology for deployment is standardized and open, it can be run a) locally or b) on minimum server requirements.

Based on the learnings and observations with the above prototypes, in the following the challenges for upcoming work are defined.

7.3 FUTURE CHALLENGES

As challenges for upcoming, iterative adaptation of the marketplace model, the following aspects are deemed important:

- **Onboarding Process:** as the C2T marketplace is considered as an internal and external view on the offerings of the project but also beyond, mechanisms for onboarding are required. These mechanisms are currently considered in the form of organizational governance rules and require an extension with respect to axioms defined for innovation items and offerings. As such, the governance process needs to reflect the contextual information of item use, potentially success stories and challenges observed per item and provide feedback to item owners.
- **Integration with Assessment Methods:** the current API is defined based on the query results of the model. Alignment is required to provide a harmonized interaction with the assessment method development stream and lessons-learned from pilot use.
- **Integration with other Marketplaces:** as a bidirectional interface, integration capabilities are required to a) discover, digest, and import items from other marketplaces and b) provide content of the C2T marketplace model as input for external sources. The idea for this challenge is to maintain the model as an item repository, abstracting upon external sources. Extensions to the metamodel are foreseen to accommodate variants in items and their axioms. The selected approach of applying metamodels provides this flexibility.
- **Smart Behaviour:** including concepts of usage elevates the contextual representation of items and might lead to a learning environment, defined as smart behaviour of the marketplace model. This included functionality to suggest and propose potential usage streams/trajectories of items (temporary logics), logical bundles or support interactions with users.

8 REFERENCE

[1] D. Karagiannis and H. Kühn, *Metamodelling Platforms*, vol. 2455. 2002.

[2] D. Karagiannis, H. C. Mayr, and J. Mylopoulos, Eds., *Domain-Specific Conceptual Modeling*. Cham: Springer International Publishing, 2016.

[3] R. Woitsch and W. Utz, *Business process as a service (BPaaS): Model based business and it cloud alignment as a cloud offering*, vol. 9373. 2015.

[4] "Digital Transformation Initiative In collaboration with Accenture." Accessed: May 10, 2021. [Online]. Available: http://reports.weforum.org/digital-transformation.

[5] W. Utz and D. Falcioni, "Data Assets for Decision Support in Multi -Stage Production Systems Industrial Business Process Management using ADOxx," 2018, doi: 10.1109/INDIN.2018.8472033.

[6] X. Zhang *et al.*, "SkyNet: a Hardware-Efficient Method for Object Detection and Tracking on Embedded Systems," *arXiv*, Sep. 2019, Accessed: May 10, 2021. [Online]. Available: http://arxiv.org/abs/1909.09709.

[7] M. Chein, M. Mugnier, S. Editors, L. Jain, and X. Wu, *Graph-based knowledge representation: computational foundations of conceptual graphs*. 2009.

[8] I. Nonaka and H. Takeuchi, "The knowledge-creating company: How Japanese companies create the dynamics of innovation," 1995. Accessed: May 10, 2021. [Online]. Available: https://books.google.nl/books?hl=en&lr=&id=tmziBwAAQBAJ&oi=fnd&pg=PA3&dq=nonaka+%26+takeuchi+1995&ots=pT5aIN_JDx&sig=fToIKrykPa4rw59eQlAZrGGTkRQ.

[9] R. Maier and A. Schmidt, "Explaining organizational knowledge creation with a knowledge maturing model," *Knowledge Management Research and Practice*, vol. 13, no. 4. Palgrave Macmillan Ltd., pp. 361–381, Nov. 01, 2015, doi: 10.1057/kmrp.2013.56.

[10] A. Schmidt, K. Hinkelmann, T. Ley, S. Lindstaedt, R. Maier, and U. Riss, "Conceptual foundations for a service-oriented knowledge and learning architecture: Supporting content, process and ontology maturing," *Studies in Computational Intelligence*, vol. 221, pp. 79–94, 2009, doi: 10.1007/978-3-642-02184-8_6.

[11] "Computer Aided Technologies for Additive Manufacturing CAxMan." https://www.caxman.eu/ (accessed May 10, 2021).

[12] "Home - CLOUDSOCKET.eu." https://site.cloudsocket.eu/ (accessed May 10, 2021).

[13] D. Karagiannis, R. A. Buchmann, P. Burzynski, U. Reimer, and M. Walch, "Fundamental Conceptual Modeling Languages in OMiLAB," in *Domain-Specific Conceptual Modeling*, Cham, 2016, pp. 3–30.

[14] "OLIVE - ADOxx.org." https://www.adoxx.org/live/olive (accessed May 10, 2021).

[15] "DIGIFOF - The FoF Designer: Digital Design Skills for Factories of the Future." https://www.digifof.eu/ (accessed May 10, 2021).

[16] V. Hrgovcic, D. Karagiannis, and R. Woitsch, "Conceptual modeling of the organisational aspects for distributed applications: The semantic lifting approach," in *Proceedings - International Computer Software and Applications Conference*, 2013, pp. 145–150, doi: 10.1109/COMPSACW.2013.17.

[17] R. Woitsch, “BPaaS modelling: business and IT-cloud alignment based on adoxx,” 2016, Accessed: May 13, 2021. [Online]. Available: https://dl.gi.de/20.500.12116/848.

[18] R. Woitsch and W. Utz, “Business process as a service (BPaaS): Model based business and it cloud alignment as a cloud offering,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9373, pp. 435–440, doi: 10.1007/978-3-319-25013-7_35.

[19] K. Hinkelmann, E. Laurenzi, B. Lammel, S. Kurjakovic, and R. Woitsch, “A Semantically-Enhanced Modelling Environment for Business Process as a Service,” in *Proceedings - 4th International Conference on Enterprise Systems: Advances in Enterprise Systems, ES 2016*, Mar. 2017, pp. 143–152, doi: 10.1109/ES.2016.25.

[20] S. Strahringer, *Zum Begriff des Metamodells*, vol. 38, no. 5. Techn. Hochsch., Inst. für Betriebswirtschaftslehre, p. 545.

[21] D. Karagiannis, “Agile modeling method engineering,” 2015, doi: 10.1145/2801948.2802040.

[22] D. Karagiannis, P. Burzynski, W. Utz, and R. A. Buchmann, “A metamodeling approach to support the engineering of modeling method requirements,” in *Proceedings of the IEEE International Conference on Requirements Engineering*, 2019, vol. 2019-Septe, doi: 10.1109/RE.2019.00030.

[23] “ADOxx Metamodelling Platform.” https://www.adoxx.org/live/home (accessed Feb. 20, 2019).

[24] European Commission, “Technology Readiness Levels,” 2014. https://ec.europa.eu/research/participants/data/ref/h2020/wp/2014_2015/annexes/h2020-wp1415-annex-g-trl_en.pdf (accessed May 10, 2021).

[25] R. Woitsch, W. Utz, and D. Karagiannis, “The IT-socket: Model-based business and IT alignment,” 2009, doi: 10.1109/ComputationWorld.2009.106.

[26] OMG - Object Management Group, “UML 2.5,” 2017. http://www.omg.org/spec/UML/2.5/ (accessed May 17, 2017).

[27] N. Efendioglu, R. Woitsch, W. Utz, and D. Falcioni, “ADOxx Modelling Method Conceptualization Environment,” *Advances in Science, Technology and Engineering Systems Journal*, 2017, doi: 10.25046/aj020317.

[28] “MM-DSL Specification - Grammar v1.0.pdf.” 2013, [Online]. Available: http://www.omilab.org/static/download/MM-DSL_Specification_-_Grammar_v1.0.pdf.

[29] W. Utz, “Design metamodels for domain-specific modelling methods using conceptual structures,” in *CEUR Workshop Proceedings*, 2018, vol. 2234.

ANNEX A: CODE FRAGEMENTS

Type-based sorting algorithms for view generation.

```
# get all itmes and find all dimensions, add to the respective table in the model attribute
CC "Modeling" GET_ACT_MODEL
SET nCurrentModelID:(modelid)
SET lVerticalDimensions:""
SET lHorizontalDimensions:""

CC "Core" GET_ATTR_ID classid:bp-model attrname:"Columns"
SET nColumnsAttrID:(attrid)
CC "Core" GET_ATTR_ID classid:bp-model attrname:"Rows"
SET nRowssAttrID:(attrid)


# position elements
CC "Core" GET_ALL_REC_ATTR_ROW_IDS objid:(nCurrentModelID) attrid:(nColumnsAttrID)
SET lColumnItems:(rowids)
CC "Core" GET_ALL_REC_ATTR_ROW_IDS objid:(nCurrentModelID) attrid:(nRowssAttrID)
SET lRowItems:(rowids)

SETL nColumnWidth:(6.5cm)
SETL nRowHeight:(4.5cm)
SET x:(2.5cm + nColumnWidth/2)
SET y:(2.5cm + nRowHeight/2 - 0.5cm)
SET y_max:(y)
SET x_max:(x)
CC "AQL" EVAL_AQL_EXPRESSION expr:("(<\"Marketplace item\">)") modelid:(nCurrentModelID)
SET lAllItems:(objids)
#EVENT_LOG msgType:"DEBUG" message:(lAllItems)
FOR sColumnItem in:(lColumnItems) sep:(" ") {

 CC "Core" GET_ATTR_VAL objid:(VAL sColumnItem) attrname:"Item"
 SET sColumnName:(val)
 FOR sRowItem in:(lRowItems) sep:(" ") {

  CC "Core" GET_ATTR_VAL objid:(VAL sRowItem) attrname:"Item"
  SET sRowName:(val)
  #EVENT_LOG msgType:"DEBUG" message: (sColumnName + " x " + sRowName + " " + STR x + " x " +
STR y)
  CC "AQL"  EVAL_AQL_EXPRESSION expr:("(<\"Marketplace item\">[?\""+sUserHorizontal+"\" = \""+sC
olumnName+"\"]) AND (<\"Marketplace item\">[?\""+sUserVertical+"\" = \""+sRowName+"\"])") modelid:(nC
urrentModelID)
  SET lFoundItems:(objids)
  SET lAllItems:(tokdiff(lAllItems, lFoundItems))
  #EVENT_LOG msgType:"DEBUG" message:(lAllItems)
  FOR sFoundItem in:(lFoundItems) sep:(" ") {
   CC "Modeling" SET_OBJ_POS objid:(VAL sFoundItem) x:(x) y:(y)
```

```
    }
    SET y:(y +nRowHeight)
    SET y_max:(y)
  }
  SET x:(x + nColumnWidth)
  SET x_max:(x)
  SET y:(2.5cm + nRowHeight/2 - 0.5cm)
}

SET x:(x_max)
SET y:(y_max)
FOR sRemainingObjID in:(lAllItems) sep:(" ") {
  CC "Modeling" SET_OBJ_POS objid:(VAL sRemainingObjID) x:(x) y:(y)
  SET x:(x + nColumnWidth)
}
```

CODE 3 ADOXX IMPLEMENTATION: ITEM SORT AND LAYOUT

Model transformation (ADOxx to Markdown)

```
#CC "AdoScript" INFOBOX "Starting item export ..."
SETG sExportPath:("D:\\RESEARCH PROJECTS\\CHANGE2TWIN\\WP2 Marketplace\\marketplace\\_items\
\")
CC "Modeling" GET_ACT_MODEL
SET nModelID:(modelid)
IF (nModelID = -1) {
   CC "AdoScript" ERRORBOX "No model that contains items is open! Open a model and re-run!"
   EXIT
}

CC "Core" GET_ALL_OBJS_OF_CLASSNAME modelid:(nModelID) classname:"Marketplace item"
SET lItems:(objids)
FOR sItemID in:(lItems) sep:(" ") {
   SET nItemID:(VAL sItemID)
   CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Status"
   SET sStatus:(val)
   IF (sStatus = "Ready for deployment") {
     CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Item ID"
     SET sMarketplaceItemID:(val)
     CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Name"
     SET sItemName:(replall(toutf8(val), "\"", ""))
     SET sItemFileName:(replall(replall(replall(replall(replall(toutf8(val), "\"", """), "™", ""), "\'", ""), """, ""), " "
, "_").trim())

     # get values for headers
     CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"TRL"
     SET sTRL:(val)
     IF (sTRL = "") {
        SET sTRL:"n/a"
     }
```

```
CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Solution description"
SET sDescription:(toutf8(val))
CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Owner (organisation)"
SET sOrganisation:(toutf8(val))
CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Contact person"
SET sContactPerson:(toutf8(val))
CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Contact person (email)"
SET sContactPersonEmail:(toutf8(val))

CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"DT Phase"
SET sDTPhase:(replall(val, ";", ", "))
CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"DT Uses"
SET sDTUse:(replall(val, ";", ", "))
CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"DT Levels"
SET sDTLevel:(replall(val, ";", ", "))

SET sTag:(array(0))
IF (sTRL != "") {
 SETL dummy:(aappend(sTag, "trl" + sTRL))
}
FOR sDTPhaseIT in:(sDTPhase) sep:(", ") {
   SETL dummy:(aappend(sTag, sDTPhaseIT))
}
FOR sDTUseIT in:(sDTUse) sep:(", ") {
   SETL dummy:(aappend(sTag, sDTUseIT))
}
FOR sDTLevelIT in:(sDTLevel) sep:(", ") {
   SETL dummy:(aappend(sTag, sDTLevelIT))
}
CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Solution references"
SET lSolutionReferences:(val)
CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Nature of activity"
SET sNatureOfActivity:(toutf8(val))
CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Keywords"
SET sKeywords:(toutf8(val))
CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"License"
SET sLicense:(toutf8(val))

CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Solution type"
SET sType:(val)

CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Software download URL"
SET sDownloadURL:(val.trim())
CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Software tenant URL"
SET sTenantURL:(val.trim())
CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Docker URL"
SET sDockerURL:(val.trim())
CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Service information URL"
SET sInformationURL:(toutf8(val.trim()))
```

```
    CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Service human (email)"
    SET sServiceHumanEmail:(toutf8(val.trim()))


    CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Details on product/service configuration"
    SET sConfig:(toutf8(val))
    CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Interoperability"
    SET sInterop:(toutf8(val))
    CC "Core" GET_ATTR_VAL objid:(nItemID) attrname:"Enhancements required"
    SET sExtend:(toutf8(val))

    SET sFilename:(sExportPath + sMarketplaceItemID + "-" + sItemFileName + ".md")
    # write header markdown
    CC "AdoScript" FWRITE file:(sFilename) text:"---\n"
    CC "AdoScript" FWRITE file:(sFilename) text:("title: \"" + sItemName + "\"\n") append:1
    CC "AdoScript" FWRITE file:(sFilename) text:("layout: layout_c2t_item\n") append:1
    CC "AdoScript" FWRITE file:(sFilename) text:("trl: \"trl" + sTRL + "\"\n") append:1
    CC "AdoScript" FWRITE file:(sFilename) text:("excerpt: \"" + sDescription + "\"\n") append:1
    IF (LEN sType != 0) {
      CC "AdoScript" FWRITE file:(sFilename) text:("item_type: \"" + sType + "\"\n") append:1
    }
    CC "AdoScript" FWRITE file:(sFilename) text:("download_url: \"" + sDownloadURL + "\"\n") append:1
    CC "AdoScript" FWRITE file:(sFilename) text:("info_url: \"" + sInformationURL + "\"\n") append:1
    CC "AdoScript" FWRITE file:(sFilename) text:("human_email: \"" + sServiceHumanEmail + "\"\n") append
:1
    CC "AdoScript" FWRITE file:(sFilename) text:("tenant_url: \"" + sTenantURL + "\"\n") append:1


    # organisational information
    CC "AdoScript" FWRITE file:(sFilename) text:("owner: \"" + sContactPerson + "\"\n") append:1
    CC "AdoScript" FWRITE file:(sFilename) text:("owner_mail: \"" + sContactPersonEmail + "\"\n") append:1
    CC "AdoScript" FWRITE file:(sFilename) text:("owner_organisation: \"" + sOrganisation + "\"\n") append:1
    # write dt tags
    CC "AdoScript" FWRITE file:(sFilename) text:("environment: [" + sDTPhase + "]\n") append:1
    CC "AdoScript" FWRITE file:(sFilename) text:("purpose: [" + sDTUse + "]\n") append:1
    CC "AdoScript" FWRITE file:(sFilename) text:("category: [" + sDTLevel + "]\n") append:1
    # write all tags in a common field for generation only
    CC "AdoScript" FWRITE file:(sFilename) text:("tags: [" + replall(replall(STR sTag, "{", ""), "}", "") +"]\n")
 append:1

    CC "AdoScript" FWRITE file:(sFilename) text:"---\n" append:1


    # write content
    CC "AdoScript" FWRITE file:(sFilename) text:("\n\n### Information\n\n") append:1
    CC "AdoScript" FWRITE file:(sFilename) text:(sDescription + "\n\n") append:1

    FOR sSolutionReference in:(lSolutionReferences) sep:("\n") {
```

```
    CC "AdoScript" FWRITE file:(sFilename) text:("["+sSolutionReference+"]("+sSolutionReference+")" + "
\n\n") append:1
  }



  CC "AdoScript" FWRITE file:(sFilename) text:("\n\n### Usage information\n\n") append:1
  IF (sType ="sofware-download" AND LEN sDownloadURL != 0) {
   CC "AdoScript" FWRITE file:(sFilename) text:("[Download]("+sDownloadURL+")" + "\n") append:1
  }
  IF (sType = "software-tenant" AND LEN sTenantURL != 0) {
    CC "AdoScript" FWRITE file:(sFilename) text:("[Access]("+sTenantURL+")" + "\n") append:1
  }

  IF (sType = "software-docker"  AND LEN sDockerURL != 0) {
    CC "AdoScript" FWRITE file:(sFilename) text:("[Deploy]("+sDockerURL+")" + "\n") append:1
  }

  IF (sType = "service-human/knowledge") {
    CC "AdoScript" FWRITE file:(sFilename) text:("[Contact](mailto:"+sServiceHumanEmail+")" + "\n") ap
pend:1
  }

  IF (sType = "service-info") {
    CC "AdoScript" FWRITE file:(sFilename) text:("[Retrieve]("+sInformationURL+")" + "\n") append:1
  }

  # CC "AdoScript" FWRITE file:(sFilename) text:(sInformation + "\n") append:1
  CC "AdoScript" FWRITE file:(sFilename) text:("\n\n#### Nature of activity\n\n") append:1
  CC "AdoScript" FWRITE file:(sFilename) text:(sNatureOfActivity + "\n") append:1

  IF (LEN sKeywords != 0) {
    CC "AdoScript" FWRITE file:(sFilename) text:("\n\n#### Keywords\n\n") append:1
    CC "AdoScript" FWRITE file:(sFilename) text:(sKeywords + "\n") append:1
  }

  IF (LEN sLicense != 0) {
   CC "AdoScript" FWRITE file:(sFilename) text:("\n\n#### License\n\n") append:1
   CC "AdoScript" FWRITE file:(sFilename) text:(sLicense + "\n") append:1
  }

  IF (LEN sType != 0) {
   CC "AdoScript" FWRITE file:(sFilename) text:("\n\n#### Item type\n\n") append:1
   CC "AdoScript" FWRITE file:(sFilename) text:(sType + "\n") append:1
  }



  IF (LEN sConfig != 0) {
   CC "AdoScript" FWRITE file:(sFilename) text:("\n\n### Use/Configuration\n\n") append:1
```

```
      CC "AdoScript" FWRITE file:(sFilename) text:(sConfig + "\n") append:1
    }

    IF (LEN sInterop != 0) {
      CC "AdoScript" FWRITE file:(sFilename) text:("\n\n### Interoperability\n\n") append:1
      CC "AdoScript" FWRITE file:(sFilename) text:(sInterop + "\n") append:1
    }
    IF (LEN sExtend != 0) {
      CC "AdoScript" FWRITE file:(sFilename) text:("\n\n### Extend information\n\n") append:1
      CC "AdoScript" FWRITE file:(sFilename) text:(sExtend + "\n") append:1
    }
  }
}
```

CODE 4 ADOXX IMPLEMENTATION: TRANSFORMATION AND EXPORT TO MARKDOWN

Item Graphrep

```
GRAPHREP
SHADOW off
FILL color: white
AVAL set-default:"Offering" type:"Type"
IF (type = "Innovation item") {
  PEN style:dash
}
ROUNDRECT x:-2cm y:-1cm w:4cm h:2cm rx:0.1cm ry:0.1cm
PEN
FONT h:22pt
ATTR "Item ID" w:c h:c
FONT
ATTR "Name" w:c:4cm h:t y:1.3cm line-break:rigorous
FILL color:gray
AVAL set-
default:"_not_init_" status:"Status" # Initial@Draft@Ready for deployment@Deployed@Decomissioned
AVAL set-default:"0" show_state:"Show status"

IF (status != "_not_init_" AND show_state = "1") {
  IF (status = "Initial") {
    FILL color:white
  }
  IF (status = "Draft") {
    FILL color:red
  }
  IF (status = "Ready for deployment") {
    FILL color:orange
  }
  IF (status = "Deployed") {
    FILL color:green
  }
  IF (status = "Decomissioned") {
```

```
  FILL color:gray
 }
 ELLIPSE rx:0.3cm ry:0.3cm x:-2.5cm y:-0.7cm
}
```

CODE 5 ADOXX IMPLEMENTATION: ITEM GRAPHREP

ANNEX B: ITEM DESCRIPTION TEMPLATE

The initial item list (presented in section 2.5) has been described using a common template according to the metamodel design presented in this deliverable. The template is available in TABLE 6 below.

TABLE 6 ITEM DESCRIPTION TEMPLATE

{ID}: {Long Title}	
DESCRIPTION OF MARKETPLACE ITEM	
Owner:	who is responsible {natural person, legal person}
Title:	header which is first identification point (max. 25 char.)
Long Title {optional}:	optional for mouse over, help text, second line title (max. 60 char.)
Short Abstract:	preview of item to gain awareness (max. 200 char.)
Description:	description to clarify what the user can expect (max. 2 pages)
TRL – Level:	definition of the technical readiness level
IPR / Licence {optional}:	who has the right / licences / is first author / first reference
ACCESSING THE MARKETPLACE ITEM	
Contact Person {innovation}:	whom to contact if I want to know more
Contact Details {innovation}:	Email of contact person
Contact Point {offering}:	whom to contact if I want to buy / use it {e.g. Firefox, Open Source can be TRL 9}
Access Credentials {offerings}:	how to contact if I want to buy / use it
Logo – Icon:	Icon and / or image to be displayed for
DESCRIPTION OF ITEM CONTENT	
Item Type / Nature:	Select one of the following • ***Consulting Offering:*** Consulting, Offering knowledge, experience and interpretation, training, …

- ***Information Offering:*** Offering of data (sets), reference material / models, Pointer / References to Communities, Studies, Success Stories, …
- ***Software Offering:***
  - „Library“ for use {SW that needs compilation}
  - „SW Packages“ for download {App, Browser Plug-In,…}
  - „SaaS Offering“ for Multi-Tenant operation {Simulation, Repository, Analytics, …}
  - „Bundle Offering“ for individual deployment {Simulation, Repository, Analytics, …}
- ***Hardware Offering:*** Digital Infrastructure and enabling devices {Edge devices, RFID, …}
- **Solution Offering:** Complete set of parts that work for a purpose

Information:	List of links to get detailed information about the item
Use:	List of links to use the item
Extend:	List of links to extend the item