Decentralized Registries: Taxonomy of decentralized registries and an architectural overview

April 2023

Discussion paper

Imprint

Publisher
Industrial Digital Twin Association
Lyoner Strasse 18
60528 Frankfurt am Main
Germany
https://www.industrialdigitaltwin.org/

Status
April 2023

Layout
Publik. Agentur für Kommunikation GmbH

Contents

Table of Figures

1 Motivation – Decentralized Registries for I4.0 Data Spaces

Future digital I4.0 ecosystems will be cross-company, open, dynamic, and collaborative. The requirements arising from that demand highly scalable solutions without single points of failure or concentration of power. Participants will form exchange channels on the fly, established in the order of seconds, and demand that the underlying infrastructure keep up with their needs while also adhering to flexibility, security, regulation compliance, and other functional and non-functional requirements that will empower future business ecosystems. Decentralized and scalable infrastructure components are therefore seen as necessary building blocks for the evolving I4.0 Data Spaces [1].

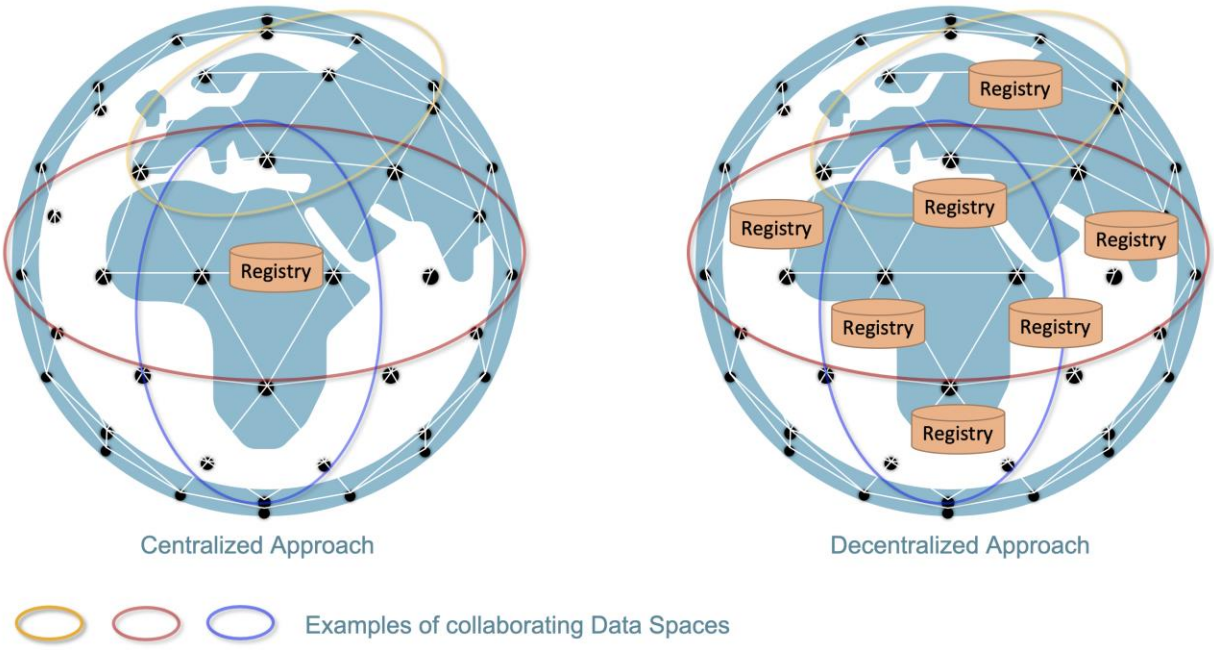

Figure 1: Multitudes of registries at different company premises

Regarding the Asset Administration Shell (AAS), the AAS Registry is one of the core enablers for searching for and locating AAS or Submodels. Without reliable, scalable, and trustworthy registries, data consumers cannot find the related AAS for an asset, in particular when the AAS is hosted by another company. AAS providers face the problem of how to announce new objects in their data ecosystem. Even though a globally known and accessible registry appears to be a feasible solution, the monopoly position of global platforms introduces characteristics in the industry environment that are unacceptable for most industry players and would represent a no-go criterion regarding their participation in the global platform. Operators of any such registry model automatically gain information advantages and become critical, and thereby powerful. Therefore, they become actors with the undesired ability to discriminate against other companies. As both such kinds of gatekeepers as well as single point of failures need to be prevented, decentralized AAS Registries become a strict requirement (Figure 1).

The high importance of decentralized registries requires an adequate definition of what 'decentralization' means in this context and which criteria determine when it has been reached.

The current discussion is characterized by different interpretations of decentralization and how it can be established in the context of I4.0. Often, private and public deployments are mixed up with traditional synchronized data storages and emerging distributed ledger technologies. The resulting confusion complicates the alignment processes and hampers the establishment of reliable decentralized registries in cross-company ecosystems.

This document introduces several views on decentralized architectures for AAS Registries, discusses their implications, and derives different reference patterns. Based on this aligned understanding, the document presents advantages and disadvantages of the identified architecture patterns.

It can not be expected, that there will be any one-size-fits-all or “best” approach that is capable of fully accommodating all envisioned I4.0 scenarios. The prioritization and weighting of advantages and disadvantages strongly depends on the use case and the corresponding requirements.

This document is intended to serve as support for the reader by providing an overview of the approaches, relevant comparison criteria, and characteristic features rather than arguing in favor of one specific pattern.

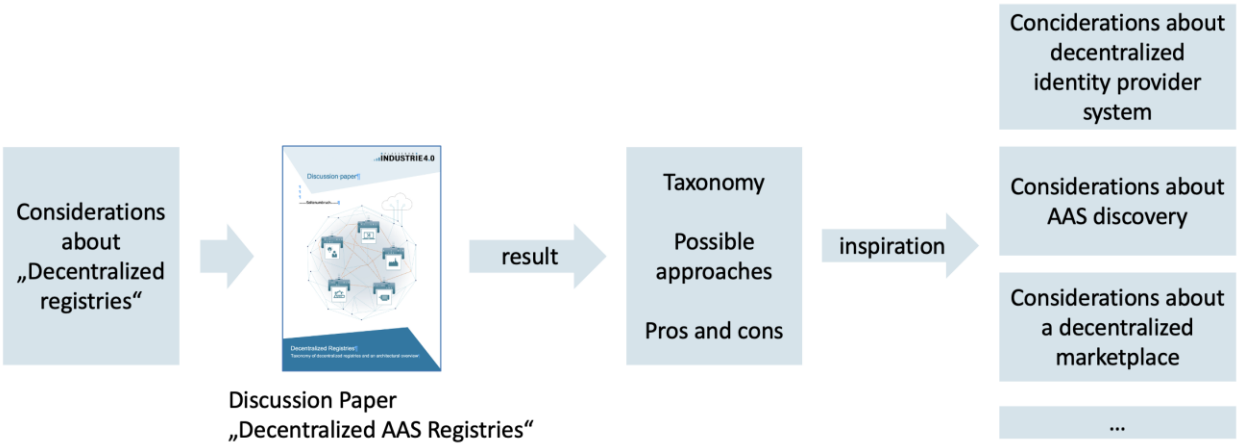

Figure 2: Scope of the document

Additionally, this document separates the registry from concepts such as discovery, repository, marketplace, shared production platform, querying, and others (Figure 4). These concepts face similar challenges to the ones mentioned above if implemented in a decentralized manner. The design principles provided in this document can therefore be relevant to them as well and can be considered in the development of respective decentralized solutions.

Chapter 1	Motivation
Chapter 2	Functionality of an AAS registry
Chapter 3	Approaches to building a decentralized registry
	Characterization of approaches
	Recommendation and possible roadmap

Appendix 1	Overview of established technologies for implementing the various approaches to building a decentralized registry
Appendix 2	Current implementations of AAS registries

Figure 3: Principal approach of the document

The remainder of this discussion paper is structured as follows. This chapter describes the current and future challenges of AAS Registries. It introduces the methodology of the applied examination and explains its limitations.

Chapter 2 briefly summarizes the specified functionalities and capabilities of the AAS Registry concept, whether centralized or decentralized.

Chapter 3 continues with a discussion on how the registry’s scope can be extended and lists possible approaches along with their advantages and disadvantages. Chapter 3 concludes with a short summary and a possible roadmap for the establishment of a decentralized registry in I4.0 Data Spaces.

Appendix 1 provides an overview of existing approaches and relevant technologies that enable decentralized registries. This overview is intended to serve as the basis for further discussions, concepts, and specification development of decentralized AAS Registries.

Appendix 2 presents and classifies approaches for implementing decentralized AAS Registries that are currently being discussed and developed in the I4.0 community.

2 AAS Registry – Definition and Requirements

2.1 What is an AAS Registry?

The AAS Registry closes the gap between an AAS Discovery Service, which resolves Asset identifiers to AAS identifiers, and the Repository Service at which the AAS are accessible. The first-class citizens of the AAS Registry, the Descriptor objects, contain the information to the AAS endpoints. They enable the AAS clients to find the correct server. No further interface functions are currently defined, even though these might be added in the future. Examples might include additional query parameters to filter the set of returned Descriptors if the actual AAS identifier is not known, or the provisioning of company identifiers to limit the number of potential target repositories.

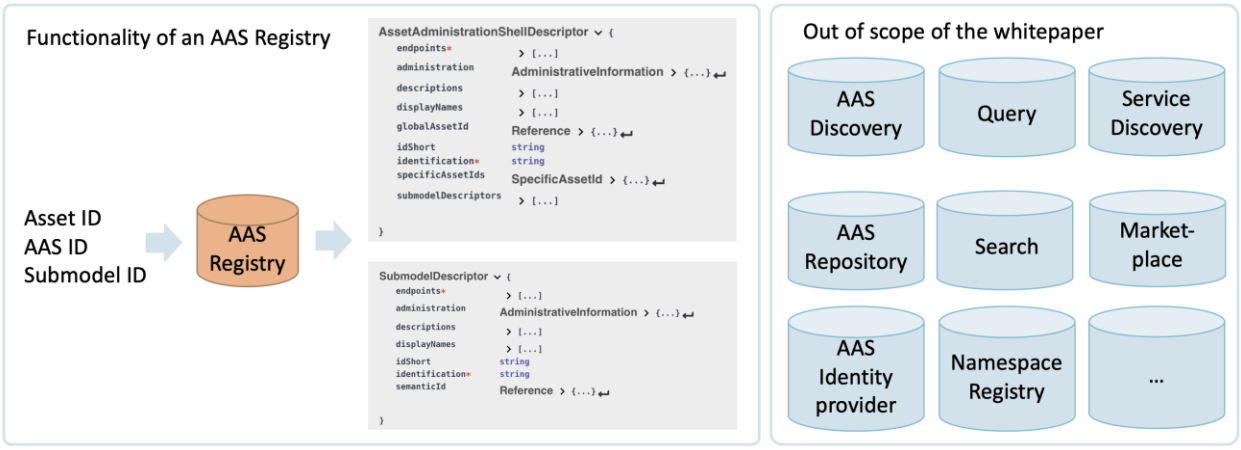

Figure 4: Differentiation of the AAS Registry Service from other concepts and infrastructural components

In the most generic case, the client is not aware of any additional information about the desired AAS, making the AAS Registry its single source of information.

2.2 Perspectives on Requirements for AAS Registries

A detailed listing of requirements for an industry-ready AAS Registry is a task for product managers of software vendors who intend to make a commercial offering, in close collaboration with e.g. Enterprise Architects and IT project managers on the user side.

In the context of this paper, it is just intended to provide a discussion of the different viewpoints that influence the specification of such an offering.

Security

As discussed above, an AAS Registry needs to be seen as a critical building block of future I4.0 Data Space architectures. Hence, the requirements for the security of a solution are correspondingly high. They include, for example:

- Preventing malfunction of the system due to external and internal attacks is mandatory.
- Protection of the data (i.e., registry content) as such – e.g., measures to avoid malicious data records (e.g., unsafe URLs) should be self-evident.
- Meta-data like data traffic, data amount, logfiles, and the like need to be regarded as highly sensitive information that needs to be protected.
- Guaranteeing trustworthiness of the information provided, secure interactions of stakeholders with the registry, and overall secure operation and management across the lifecycle of the registry and its processes.

Identity & Access Management (IAM)

It is common that Creating, Reading, Updating and Deleting (CRUD) data in the AAS Registry requires a solid and state-of-the-art security concept that adequately covers all requirements, e.g., for authentication, authorization, and accounting. For instance, role-based (RBAC) or attribute–based (ABAC) access control has to be designed taking into account the special challenge of assigning the appropriate roles and/or attributes to members of external companies as well.

Quantity structure/Performance/Scalability

The assumed mission criticality of the AAS results in correspondingly high requirements regarding, e.g.,

- the supported number of Descriptors,
- the response times (including technical latency but also non-technical times, e.g., time for administration/granting of authorizations), and
- the number of concurrent users.

Utilization of state-of-the-art technologies for (cloud) deployment, mirroring, and zero-downtime maintenance are also a logical conclusion of the registry's fundamental significance.

Ownership/Operating Company

A major motivation for the idea of decentralized AAS Registries is the sovereignty aspect.

- Companies will retain full control over IAM processes but also over the lifecycle of the Registry content.
- The market will need decentralized registry solutions that are operated as a service.
- A decentralized registry that is operated by a service provider on behalf of the actual AAS provider will have to adhere to requirements regarding multi-tenancy, billing/accounting, convenience, transparency, and the like.
- In addition, all features that enforce trust between the client and the operating company should be carefully designed and implemented.
- Archiving, client-specific reporting and logging, and audit readiness are noteworthy topics in this context.

APIs

- API Operations

The already specified and well-defined API Operations [1] for registering AAS in registries (in the first instance, the PUT, POST, DELETE services for the AAS Descriptors and Submodel Descriptors) are independent of the debate about centralized or decentralized registries.

- Access control configuration

Further evaluation might be needed regarding the question of (meta-)data required to configure access control on the right level of granularity.
On challenge might be, that a submodel, which is used in by different AASs (ID: 1,2 and 3), can be discovered by an external party Y, but not by an external party Z. Company Z may only discover the same Submodel for other AASs (ID: 4,5,6 and 7) – due to the fact that a certain property of the Submodel has a customer-specific value.

- API definition for the decentralized registry

An additional topic for further evaluation is the API definition for the decentralized registry acting as a client in order to interact with other registries (see options discussed in Chapter 3).

- Interaction between registries

One decentralized registry instance is an element of many in a registry network and may need to take care of informing other registries about changes, and/or of synchronizing itself with others.

- Create and update

The main information managed by an AAS Registry, the AAS Descriptor, and the Submodel Descriptor are rather simple data objects. However, the registry must validate (check the syntax of) new and updated records.

Minimal functional requirements

Many mandatory requirements result from a deep-dive examination of the above-mentioned perspectives Security, IAM, Performance, and API Operations.

Depending on future discussions regarding additional meta-data for access control and content-related discovery of AAS, the validation and potential verification of such information is self-evident but may be more challenging due to expected customer-specific definitions and compositions of such meta-data.

Some example touchpoints for non-mandatory requirements and perspectives are:

- Differentiation by more advanced, smart, sophisticated discovery mechanisms
- Offering of notification functions to stakeholders in the sense of “Have you heard that new AAS have become available”
- Recommendations like “Customers who searched for such AAS also requested the following AAS.”
- IAM-related workflows like “User or company xyz requested access to AAS_123; which Submodel shall be visible to this requestor?”
- Reporting and analytics like “Last month, the most popular AAS were …” or “Mid-sized companies last month triggered 1234 queries to our registry, whereas 5678 queries came from Fortune 500 companies.”
- Additional add-ons for registries to accommodate company- or consortia-specific requirements, e.g., utilization of a specific technology.

However, any such requirements are seen as part of registry extensions, and not as part of the core registry aspects discussed in this work. This implies a working model similar to, e.g., modern software (e.g., Internet browsers), where apart from the core functionality, customized extensions can be realized via add-ons (extensions).

3 Principal Approaches for Decentralized Registries

In the previous chapters, it was mentioned that the AAS Registry is one of the core enablers for the establishment of cross-company I4.0 Data Spaces.

In this chapter, possible architectures of cross-company registry networks will be described. In order to accomplish this, a model is introduced first, where the discussion about Data Spaces is divided into three levels (Figure 5).

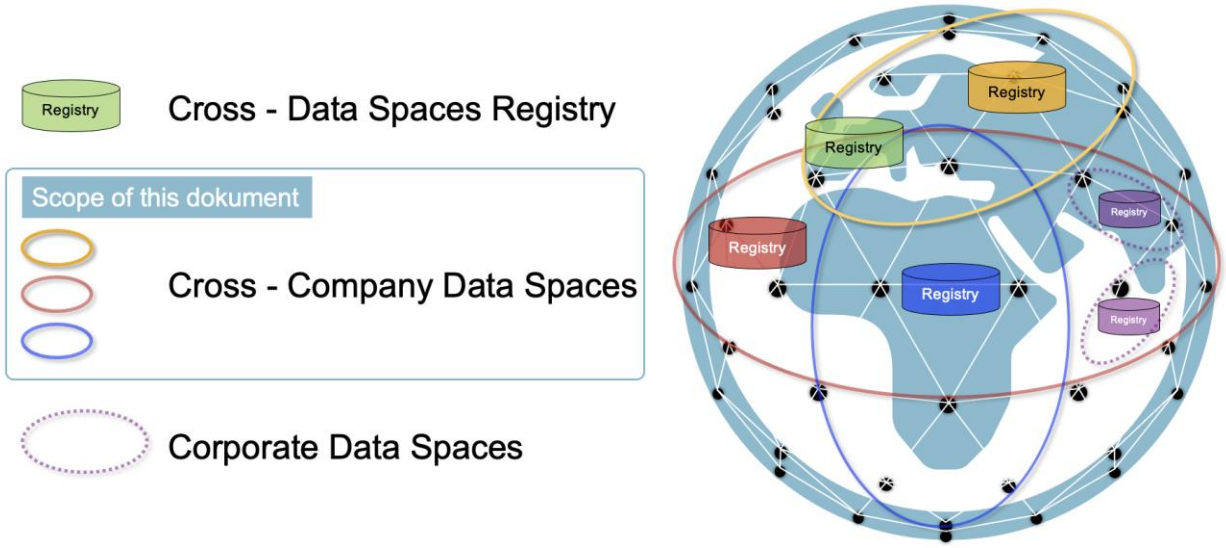

Figure 5: Classification of data spaces into three focus levels

The bottom level is the corporate Data Space. Company-internal registries can exist at this level in different architectures, just as they can in cross-company environments. However, further details and possible interactions of internal company registries with cross-company registries will not be discussed here as these may require separate considerations. The aim is also to focus on interoperable registries that enable interaction and collaboration at global scale.

The next level up follows the cross-company Data Space, which enables interaction between independent companies.

The third and highest level, which will not be discussed further in this discussion paper, is the interaction across Data Spaces. The definition of this level is motivated by the need for possibilities to exchange information between dedicated Data Spaces, for instance an automotive Data Space and an energy Data Space. This document provides no specifications and declarations regarding the architecture and functionality of cross-company registries. Only two possible options for implementing the necessary registry function are identified:

- A Data Space-wide registry (or registries) exists. The aspects discussed in this document may also be relevant for the decentralized design of such a cross-Data Space Registry.
- Each Data Space-internal registry can provide statements about which other Data Spaces it is aware of.

Overall, for the sake of interoperability, easy interaction, and low development integration effort, registries are expected to adhere to the standardized APIs when it comes to interactions with external stakeholders, while no assumptions are made about their internal architecture and operation. Overall, however, it is expected that similar technologies and architectures will be used to realize both internal (corporate) and external (public) / cross-company / cross-data-space registries.

This chapter describes the design principles and possible architectures of cross-company registry networks within a single Data Space. To arrange different possible architectures, a taxonomy of these design principles is shown in Figure 6.

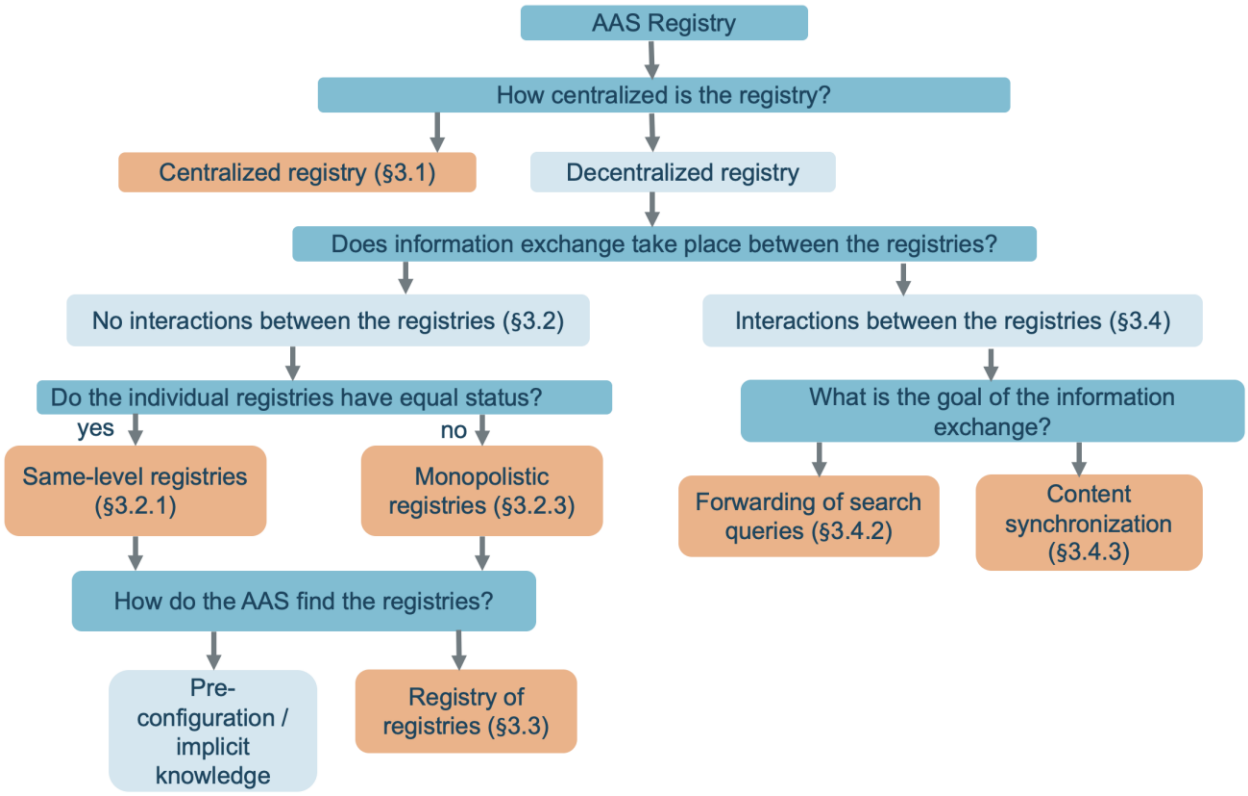

Figure 6: Taxonomy of approaches for decentralized registries [2]

To make this arrangement, the question of how centralized the registry function is will be used to divide it into a centralized approach and a decentralized approach. The architectures that will be explained in the following are marked in orange, for instance the centralized approach, which is discussed in chapter 3.1.

As shown in the taxonomy, the decentralized approaches are subdivided again with the help of different questions, marked in blue.

3.1 Centralized Registry

Digital B2C platforms such as eBay, Amazon, Google, and Uber have long been part of many people's everyday lives. Many of these virtual platforms have demonstrated their disruptive potential and have revolutionized entire industries [3]. This leads to the question of how the design principles of these well-known e-commerce platforms can be used in the context of Industry 4.0 and whether they can be integrated into the development of an I4.0 Data Space. The centralized approach, shown in Figure 7, consists of multiple companies representing AAS users or AAS providers and a central registry located outside these companies.

In the context of I4.0 Data Spaces, this means that there exists only one dedicated, globally unique AAS Registry, which is known to interested stakeholders and operated by only one (authorized) provider. This registry might be decentralized in terms of having a cluster operation mode or regionally distributed instances being deployed to reduce response times and increase resilience. However, from a business perspective, there is one single business entity that operates it.

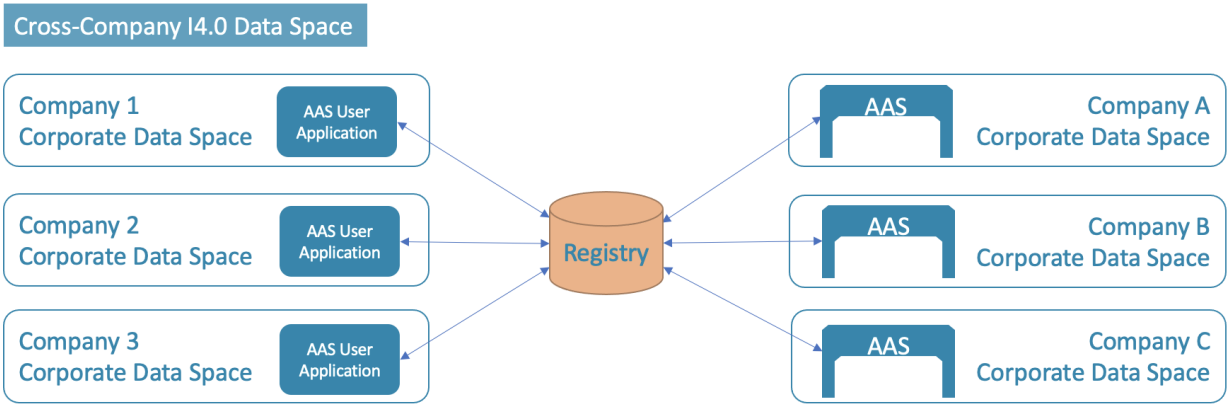

Figure 7: Centralized registry [2]

Advantages	Disadvantages

• Only one registry entry needs to be maintained.
• No need for a discovery step to find the right registry as there is only one.
• All information is in one location (good overview).
• Availability and up-to-date AAS Registry entries for all lifecycle phases and for all participants can be ensured.
• Simple architecture, thus relatively low maintenance effort and cost of operation.
• Fewer interoperability problems between AAS Services and AAS Registry.
	• Introduces a single point of failure.
	  ○ Failure of the central registry breaks all cross-company interactions.
	  ○ High traffic for one single instance (possible bottleneck).
	  ○ High number of data objects need to be stored and indexed.
	• Power concentration: Registry operator becomes a gatekeeper.
	• The AAS Registry operator has access to all AAS and Submodel Descriptors, which can be used to disclose or infer business secrets.
	• By monitoring client activities, conclusions can be drawn about the business activities and behavior of individual companies.
	• Registry operators may abuse their control over the platform infrastructure and manipulate registry activity for their own benefit, e.g., by giving preference to some participants in search results.

Use case relation – killer argument against this approach:

It might not be acceptable for business, geopolitical, or other reasons that one central registry enforces the storage of all AAS Descriptors of all participants worldwide.

Outcome:
The centralized registry approach is not recommended for an AAS ecosystem and will not be discussed further in the remainder of this document. The development of a centralized registry could be a valid first step towards implementing I4.0 scenarios and use cases in the short term. However, it should be taken into account that this approach cannot meet the requirements of collaborative I40 ecosystems.

Note: this could be a viable approach if it is operated by a trustworthy non-profit organization (e.g., similar to ICANN for DNS) and its actual implementation relies on failovers, e.g., a distributed system, load balancers, etc. However, considering the business relevance of the hosted information, it is highly unlikely that a single registry will be sufficient or successful for the various use cases and domains that are envisioned to benefit from AAS.

3.2 Non-Interacting AAS Registries

One possible approach for building a more decentralized system is the constellation of multiple, equivalent registries that are independent of each other. This approach could be seen as an evolution of centralized approaches. It is based on the assumption that industrial companies will try to build up their own registries and establish themselves as platform operators (Figure 8). Examples from everyday life are purchasing portals for suppliers of large companies, social media platforms (e.g., Twitter, TikTok, Meta), or shopping portals (e.g., online marketplaces of H&M, Zara, Mango). These are also systems that are used by several parties for communication (buyer – seller, member – member) and act independent of each other.

3.2.1 Equivalent but Independent AAS Registries

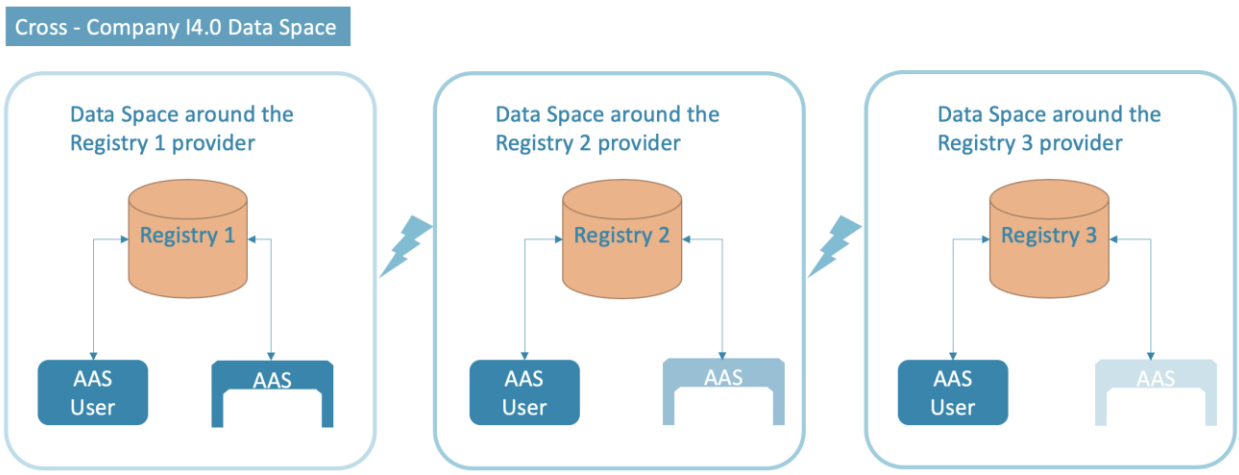

Figure 8: Several registries of the same level and independent of each other in one Data Space [2]

In this pattern, no information exchange takes place between the individual registries. This means that AAS Descriptors can only be found if the ‘correct’ registry is queried or if a client makes the effort and asks for all of them. At this point, the question arises of how to find these individual registries or how to know which registries exist at all in a Data Space?

It can be assumed that company-specific ecosystems will emerge around individual registries. This will put large companies at an advantage and SMEs at a disadvantage, as large companies will likely be able to build their own controlled ecosystems.

Advantages	Disadvantages
• Relatively low total cost of operation for each registry. • Simple architecture, therefore low maintenance effort for registry providers. • Full flexibility to deploy one's own registry. • Registries are equivalent and can be used to structure the ecosystem (domain-specific partitioning possible). • Each registry operator/consortium has clear control over their registry.	• Still single point of failure for parts of an I4.0 Data Space. • Failure of one registry may still break all cross-company communication in one domain. • Discovery of registries for the AAS client requires a priori knowledge: How to find the registries at runtime? • Limited content: An empty query result may merely indicate that the wrong registry was queried (exhaustive search would be needed if no other means are provided, e.g., a meta-registry). • If an AAS is registered in multiple registries, one has to make sure that they are all up to date (and in sync through other mechanisms). • In case an AAS needs to be removed, there might not be uniform ways to do that for each registry (maintenance complexity increases).

Use case relation – killer argument against this approach:

The approach does not add significant value to the overall system. While individual companies can build their own controlled ecosystems, this approach does not simplify or accelerate cross-company interactions in any way. In fact, it makes them potentially even harder, since the problem of finding AAS is transferred to the next level, and how to find individual registries still remains a challenge. In addition, it may be challenging for smaller stakeholders, e.g., SMEs, to operate their own registries, which could be a competitive disadvantage that has to be compensated via other ways, e.g., by forming consortia or outsourcing.

Outcome:
Independent registries add value to closed ecosystems such as private AAS networks. They may be relevant for large companies, as they can build their own controlled centralized network and cover the arising operation costs.

3.2.2 AAS Registered in Multiple AAS Registries

The long-term visions of I4.0 assume an open, highly dynamic, collaborative, and automated character of future cross-company digital ecosystems. All relevant assets must be interconnected and able to communicate and collaborate seamlessly.

To meet this requirement, an AAS must be registered in multiple (sometimes in all) registries (Figure 9). Accordingly, it must maintain its registration entries and communication relationships with multiple registries. This leads to increased organizational and implementation effort as well as redundant information that will need to be kept up to date.

Furthermore, the question arises of how the AAS operators and AAS users can find out which registries are available, and under which endpoints they can be found in the network.

Obviously, an implicit knowledge about the contents and addresses of registries is assumed. With an increasing number of AAS Registries, the result can be significant complexity and the need to manage it. The potential unawareness of the registry endpoints and their content existing in a Data Space and the further challenges mentioned above lead to the transformation of the decentralized system. As is typical for the platform economy and the winner-takes-all effect, development towards a de-facto centralized system caused by an exclusive or predominant market share of one of the registries can happen.

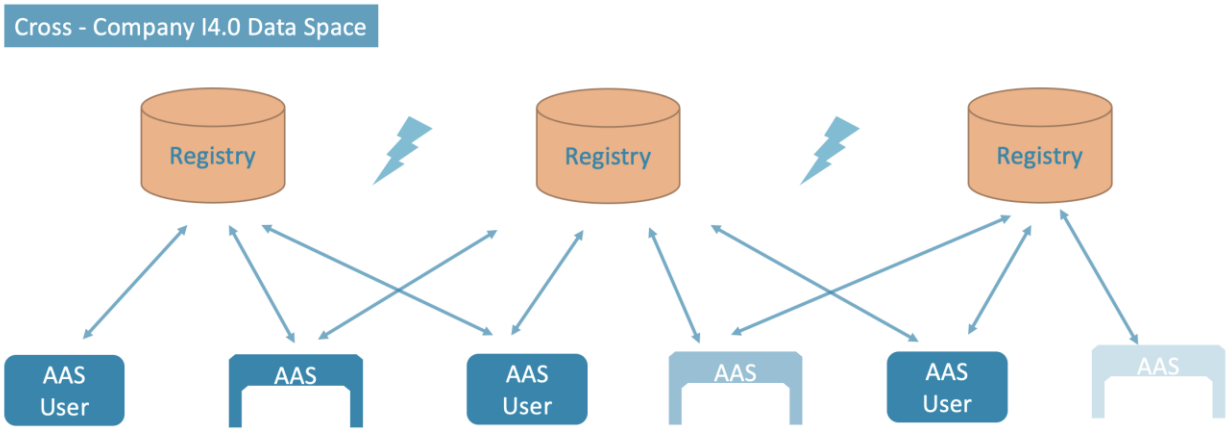

Figure 9: To ensure that the positive network effects take place, the AAS and the AAS users must be registered in several AAS Registries [2]

Advantages	Disadvantages
• Relatively low total cost of operation for each registry. • Simple architecture, thus low maintenance effort for registry providers. • No component with monopoly position. • Full flexibility to deploy one's own AAS Registry. • Registries are equivalent and can be used to structure the ecosystem (domain-specific partitioning possible).	• Increased organizational and implementation effort for each AAS provider. • Discovery of registries for the AAS provider requires a priori knowledge: How to find the registries at runtime? • Low scalability: Configuration effort for the AAS providers now also increases with the number of registries. • Potential concentration effects through winner-takes-all.

Use case relation – killer argument against this approach:

The costs for AAS providers grow significantly with the number of registries, as they have to interact with most of them to promote their AAS, introducing unnecessary hurdles to publish AAS. Even more importantly, the disclosure of business secrets disqualifies the approach for any real-world use case.

Outcome:
This is a possible development of a centralized approach that is not advisable. It leads to increased organizational obstacles and higher development effort and may create unnecessary redundance. In addition, there is a need for extensive coordination and clarification in advance, e.g., whether and under what conditions the AAS of one company can be registered in the registries of another company, whether it will have the same visibility when queried, etc.

3.2.3 Monopolistic Registries

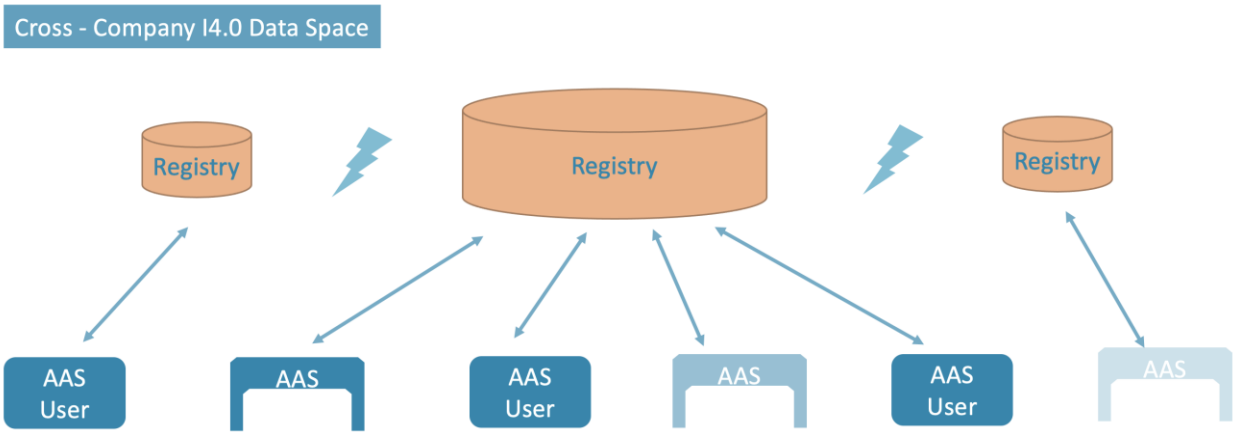

Figure 10: Through more frequent use or use of only one of the registries, the number of registered AAS and AAS users increases and the main registry does, de facto, establish a monopoly position [2]

This approach does not constitute a separate architectural approach but, as described in section 3.2.2, may be a possible consequence of the approach of equivalent registries. The following scenario is considered. During the operation of an I4.0 Data Space with several equivalent registries, a monopoly has been established by the fact that one or more registries are used more frequently or exclusively ('winner-takes-all'). The previously decentralized system does, de facto, become centralized, although de jure it remains decentralized. In the example of social media platforms already mentioned above, monopolies would emerge from economies of scale, such as Amazon or Meta today.

Advantages	Disadvantages

• Relatively low total cost of operation for each registry.
• Simple architecture, thus low maintenance effort for registry providers.
• No component with technically enforced monopolistic position.
• Full flexibility to deploy one's own AAS Registry.
• Registries are equivalent and can be used to structure the ecosystem (domain-specific partitioning possible).
	• Discovery of registries for the AAS provider requires a priori knowledge: How to find the registries at runtime?
	• Low scalability: Effort for the AAS providers now also increases with the number of registries.
	• The monopolistic registry introduces a single point of failure.
	  ○ Failure of the monopolistic registry breaks mostly cross-company interactions.
	  ○ High traffic for one single instance (possible bottleneck).
	  ○ High number of data objects need to be stored and indexed.
	• Power concentration: The monopolistic registry operator becomes a gatekeeper.
	• The monopolistic registry operator has access to most AAS and Submodel Descriptors, which can be used to disclose business secrets.
	• By monitoring client activities, conclusions can be drawn about the business activities and behavior of individual companies and their partners.
	• The monopolistic registry operators may abuse their control over the platform infrastructure and manipulate registry activity for their own benefit, e.g., by giving preference to some participants in search results.

Use case relation – killer argument against this approach:

The considerations relevant to the centralized approach also apply here. It has to be avoided that one registry does, de facto, enforce the storage of all AAS Descriptors of all participants – even though the technical requirements would allow competing services.

Outcome:
The risk of 'winner-takes-all' effects needs to be minimized by design, preventing the natural emerging of registries that are de facto monopolistic.

3.3 Registry of Registries

In the following architecture approach, there exists – in addition to the various independent registries, which may be company- or domain-specific – a registry of these registries, which we refer to as Registry of Registries (ROR). The ROR is aware of all other registries and serves to provide information for the overall system, and clients can use it, e.g., to direct their queries to the other registries. This approach thus presents a solution to the question of how to find the independently operating registries.

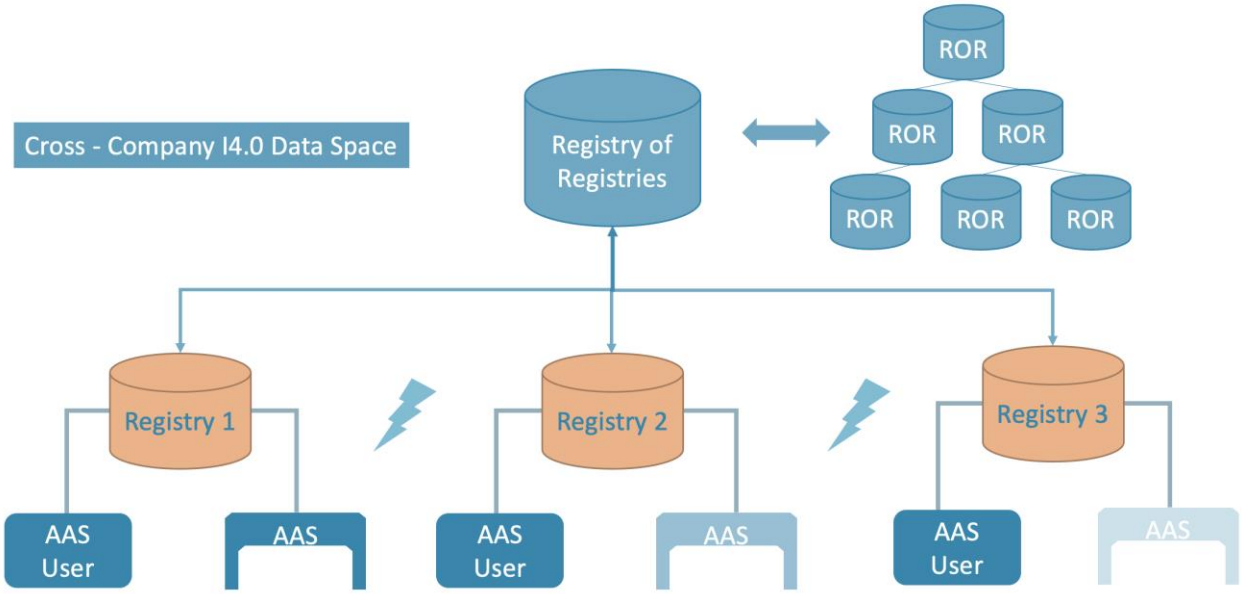

Figure 11: Registry of Registries [2]

The ROR is known to all AAS and other network participants, which means that it can be accessed by all of them. Several possibilities exist regarding what exactly is stored in the ROR and whether there is a search mechanism for its content. In the simplest case, the ROR contains only the endpoints of all registries and general information about what is stored in each registry; for example, what domain/company the registry represents.

The DNS operates on a similar principle, having a hierarchy of DNS servers with well-clarified resolving capabilities. Hence the functions of the ROR could also be fulfilled in a similar manner by a hierarchical structure of registries.

It is also possible for the ROR itself to be structured hierarchically, where the leaves would be the actual registries as shown in Figure 11. In this case, the lower-level registries would contain the Descriptors of an AAS, and the registries on higher hierarchy levels only the endpoints of the other registries. With this procedure, it is possible, for example, to divide the I4.0 Data Space into domains on the first level and then to subdivide these domains again on the following levels.

The advantage of this approach is the clean subdivision of the overall Data Space, which can be achieved by the individual registries knowing which other registries are "subordinate" to them.

The main disadvantage of this option is that the ROR leads to a concentration of power, which introduces potentially harmful dependencies. However, some of these concerns could be alleviated by having them operated by trusted third parties and at some level by the interested stakeholders within a domain.

Advantages	Disadvantages
• Relatively low total cost of operation for each registry • Flexibility to deploy one's own registry • Registry discovery problem solved • Increased resilience against single failures • Scalability: Additional registries do not impact the effort of the others or the AAS providers • Automated procedure for search queries • From a system point of view, AAS info is kept in dedicated registries and is up to date (without the need for replication) • Clear control of each registry that can be registered to RORs • An AAS can only be registered/maintained at one registry and be recursively found via RORs (low complexity, maintenance at the expense of more complex interaction). • Provides control over the registry by the owning company. The ROR only links to existing company registries.	• Single point of failure: If the ROR fails, the complete approach fails. However, backup mechanisms such as mirrors or load balancers can be built in. • ROR entries or APIs are not specified yet. • Criteria for deriving the structure of hierarchical RORs are not clear (e.g., similar to DNS, to consortia or country etc.) • Specific requirements for the type and structure of the AAS identifiers? • Additional implementation effort for the ROR instance • Unclear partitioning of AAS Descriptors: How to describe which registry to ask further? • Additional effort in operation and implementation • Increased communication (recursively in the tree) ROR provider may leverage "monopoly" to exclude registries.

Use case relation – killer argument against this approach:

The ROR forms a powerful component in the Data Space. As a result, potential negative aspects of the centralized approach, such as misuse of the position or manipulation of the requests, are reintroduced again. An unclear organizational structure would be problematic for operating RORs.

Outcome:
This approach represents an evolutionary development of the centralized system and solves the problem of registry discovery described in chapters 3.2 und 3.3. Also, an automated procedure for the search queries in the system can be created, which requires less tacit knowledge and can also be an effective mechanism to prevent the formation of monopolistic registries on the lower levels.

3.4 Interacting AAS Registries

In this section, the approach in which the registries of the AAS interact directly is considered (Figure 12). This is shown on the right branch of the taxonomy in Figure 6.

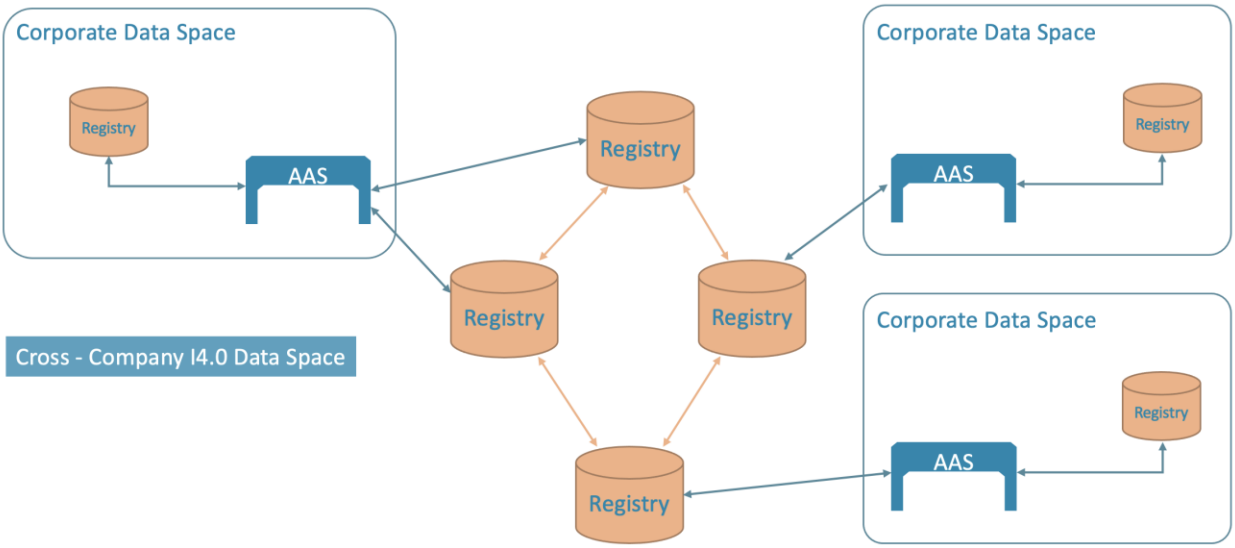

Figure 12: Communication of external registries (ring topology) [2]

The characteristics, advantages, and disadvantages of interacting registries are highly dependent on the underlying topology. Therefore, the characteristics of potential topologies shown in Figure 13 are typical for the structure of such network topologies and will be briefly explained below.

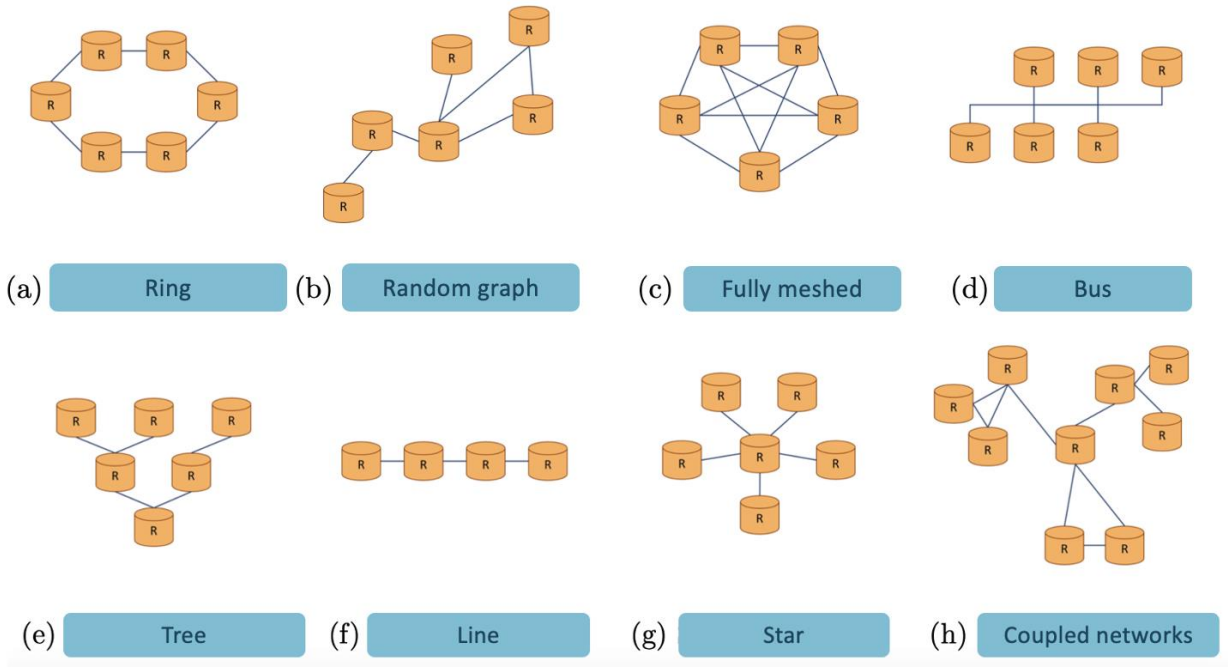

Figure 13: Possible topologies [4]

Network Topology	Some topology-dependent characteristics of networks of interacting registries
Ring topology	Ring topology is a simple implementation of the registry network that could be suitable for smaller systems. It is easy to expand, since only two new connections need to be assigned to a new participating registry. However, the failure of one single registry breaks the ring and can cause devastating communication problems.
Mesh/Random graph	The use of a random graph has the advantage that the connections or divisions of the segments can be adapted to the requirements or to the domain. It also has good resilience, since the failure of one component does not result in a complete system failure.
Fully meshed	The fully meshed topology can be very effective and fail-safe due to the high-density networking and is therefore well suited for smaller systems. However, adding new registries is time-consuming due to the large number of connections required.
Bus topology	The bus topology is a simple structure that is easy to extend. It is very fail-safe since the failure of one component does not affect the entire system. However, a connection break has several consequences, and the use of complex access methods is another disadvantage.
Tree topology	Due to its structure, the tree topology offers the option of splitting the registries. It forms a simple and easily expandable structure with a medium level of fail-safety and dense cabling.
Line topology	The line topology provides a simple and easily expandable structure. However, fail-safety is very poor here because the connection between registries would be interrupted if one component fails.
Star topology	The star topology implements very effective interconnection. It is easy to extend and offers good reliability, except if the middle component fails. It also represents a very centralized approach.
Coupled networks	Like the random graph, the use of coupled networks also offers the possibility to split the registries in a domain-specific way. Furthermore, this approach is easy to extend, but offers poor resilience due to the middle, centralized component.

3.4.1 Aim of the Information Exchange

The goal of exchanging information between interacting registries has not been explained in detail so far. In the following, possible alternatives for implementing this information exchange will be described. In the context of this document, the type of interaction is differentiated into two cases: forwarding of search queries or discovery requests, and synchronization of content between registries.

3.4.1.1 Forwarding of Search Requests

In this case, the search request of an AAS client is received by one of the registries. If this registry cannot provide suitable results, the request is forwarded to another registry. This procedure is repeated until the AAS client receives a satisfactory answer.

The advantages and disadvantages of this procedure are highly dependent on the topology.

Network topology	Some topology-dependent characteristics of registry networks forwarding their search queries
Ring topology	The ring topology can be recommended for this procedure since no dead ends can occur when forwarding search queries. Nevertheless, endless loops are possible if no suitable result is found. To prevent this, further mechanisms may have to be implemented.
Mesh/Random graph	Depending on the structure of the random graph, dead ends as well as infinite loops can occur. However, the structure offers the great advantage of dividing the entire system, meaning that search queries could be forwarded to the appropriate part of the system.
Fully meshed	The fully meshed topology results in a costly but resilient network.
Bus topology	The bus topology also provides a recommended setup, as search queries can be forwarded efficiently.
Line topology	The line topology can lead to search inquiries resulting in dead-ends, which is why in this connection, further mechanisms must be implemented.
Star topology	The star topology provides good forwarding of search queries, but always via the middle node, which again results in a central character.
Tree topology	Dead ends are possible with the tree topology, but an efficient mechanism can result from a sensible structure of the branches of the tree and thoughtful forwarding of the queries according to the division of the system.
Coupled networks	Analogous to the random graph, coupled networks can also forward search queries efficiently if the system has been divided sensibly and the queries are forwarded accordingly.

Forwarding the queries between the different registry instances allows answering queries by leveraging all available Descriptors while each individual registry only needs to store its own subset. In addition, the AAS user is freed of the challenge to find the correct registry instance.

Advantages	Open challenges
• No single point of failure • No concentration of power • Low level of implicit knowledge required for the AAS user • Minimal to no redundant Descriptors	• Discovery of registries: How can a registry identify which other registry is the right one to forward a query to? • How to implement efficient query forwarding? • How to ensure scalability in terms of the number of registries? • How to ensure industry-standard service-level agreements (response time, availability, coverage, security, etc.)?

Use case relation – killer argument against this approach:

No generally applicable disadvantages of this approach could be identified. The points listed above describe the identified research needs to build an industry-ready, secure, and robust Data Space based on registries that forward search queries. However, this is only possible if registries can operate in a trusted manner and all stakeholders involved behave well, e.g., within a specific context. Assuming responsibility for the forwarding delegates, however, the logic to the network and the client may need to wait until it is given a reply. This can be taken advantage of by malicious actors, who can overload the network with false/misbehaving requests that may, e.g., lead to DoS attacks. The alternative here is similar to DNS best practices, where results are not forwarded among the registries but instead control and results are given to the client that decides to query the next registry. This enables better load management and also delegates the point of action to the client who needs to act accordingly (e.g., authenticate in the next registry).

Outcome:
Highly decentralized approach, domain-specific partitioning possible, low demand for implicit knowledge necessary, no concentration of power. The nature of forwarding needs to be carefully considered as discussed.

3.4.1.2 Content Synchronization

It is also possible for the participating registries to synchronize content with each other on a permanent or event-driven basis. Accordingly, all registries have (partially) the same content and can answer queries of AAS users directly. The requirement here is that all registries need to have the same status of the entries, i.e., consistency of the entries must be given. The disadvantage is that a high number of participants and entries leads to increased synchronization effort. The various topologies are also suited differently for this scenario. The decisive criterion is the duration of the synchronization and its volume, which depends on the distance between the registries.

Advantages	Open challenges
• No components with monopoly position • Flexible • Highly automated • Low level of implicit knowledge	• How to ensure the synchronization of registries considering requirements for decentralization (without a central mediator), scalability, and security?

Use case relation – killer argument against this approach:

No generally applicable disadvantages of this approach could be identified apart from the synchronization effort, which is not trivial for (large-scale) distributed systems. The points listed above describe the identified research needs to build an industry-ready, secure, and robust Data Space based on registries that synchronize the search queries.

The only points that require clarification are:

- Data redundancy: AAS Descriptors need to be kept in sync in all registries by every AAS provider.
- Lacking sovereignty: The duty to expose the existence of all AAS to all registries discloses critical business secrets.

Outcome:
Truly decentralized, no monopoly/power position, no concentration of power, flexible, highly automated, low level of implicit knowledge necessary.

3.5 Conclusion – Bottom Line from an Architectural Viewpoint

In the previous chapters, various approaches for implementing I4.0 networks with and without registry functionality were identified and explained. Characteristics, open questions, advantages, and disadvantages of the individual approaches were discussed and briefly presented in corresponding tables.

The approaches to building a decentralized registry were arranged in a taxonomy based on different criteria (Figure 6), which is intended to illustrate and explain the interrelationships and differences between the individual approaches.

In conclusion, no single pattern can be identified as a clear favorite that could be described as “the best architecture” for all potential I4.0 use cases.

The individual evaluation of advantages and disadvantages, as well as the challenges associated with the different approaches described in this document, depend on the specific use cases and scenarios, as well as on the general context of the development and emergence of a specific I4.0 Data Space.

3.5.1 Relationship between Company-internal and Public AAS Registries

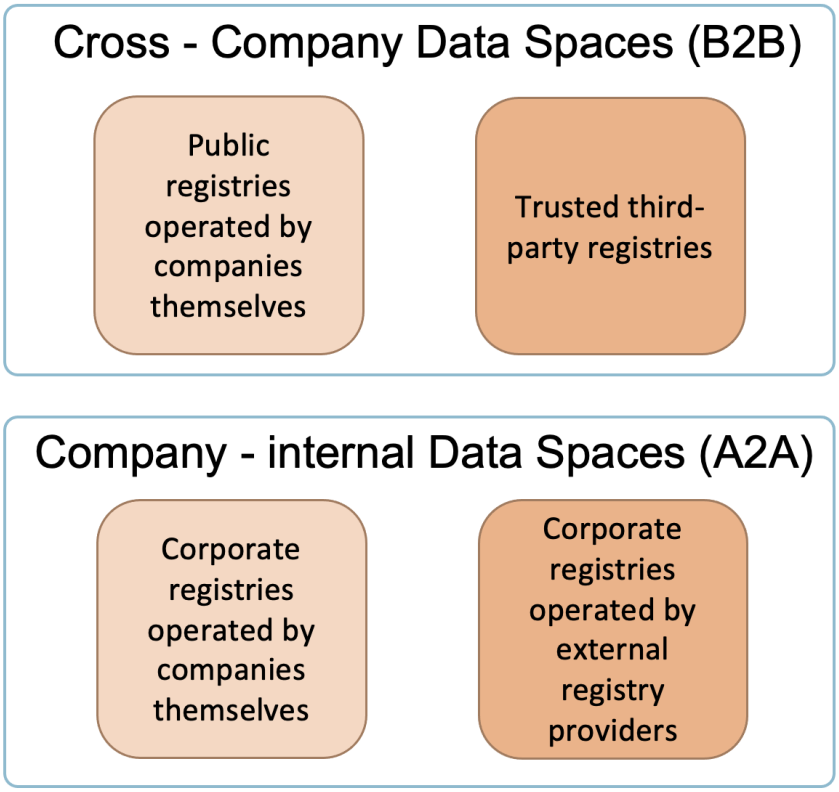

Figure 14: Relationship between company-internal and public AAS Registries

When implementing cross-company AAS Registry networks within I4.0 Data Spaces, the distinction between corporate and public AAS Registries plays a major role. Since their (non-)use can result in different characteristics, advantages, and disadvantages, they will be explained in the following using Figure 14.

“Corporate AAS Registries” are hosted internally by a company in restricted networks; examples include company-wide intranets or even smaller networks restricted to certain sites. Applications operated by other companies cannot access corporate AAS Registries if they are not deployed in the respective network. Therefore, internal AAS Registries are used for the registration of all AAS and Submodels, regardless of whether they contain confidential data or not.

“Public AAS Registries” operated by companies themselves are exposed externally to other parties through the Internet. They might also serve the purpose of corporate AAS Registries but require sufficient access control regimes beyond those of purely internal AAS Registries. Usually, public AAS Registries provide only a subset of their content to external consumers and other AAS Registries.

“Trusted third-party AAS Registries” further extend the concept of public AAS Registries. They act as neutral players in an I4.0 Data Space and do not differentiate as to whom they show their content. To the extent that specific I4.0 Data Space regulations allow this, public AAS Registries offer registration and query APIs to all I4.0 Data Space members.

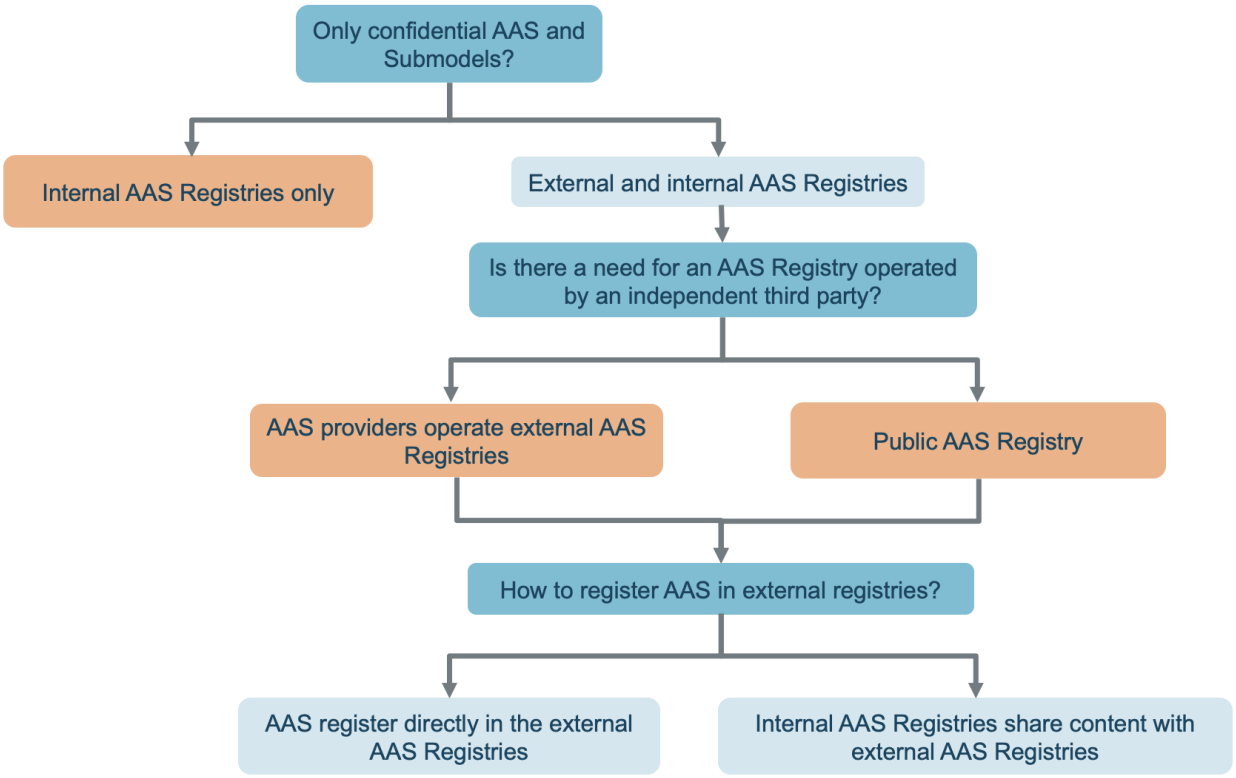

Figure 15: Decision tree for internal and external AAS Registries

The first decision towards a feasible architecture is whether the companies involved possess AAS or Submodels whose existence must not be exposed to others. If this is the case, the corresponding AAS and Submodels are only registered in the corporate AAS Registry. Company-internal AAS and AAS users can then only find each other via this registry and then communicate with each other.

If the existence and locations of (parts of) the AAS and Submodels are also to be shared with external partners, at least one public AAS Registry needs to be hosted. This AAS Registry might also appear as a corporate AAS Registry for company-internal application, or it might synchronize content with corporate AAS Registries. Companies might also decide to only operate one public AAS Registry (or several).

The disadvantage here is that in the case of a failure of this public AAS Registry, internal communication is also interrupted since the AAS are no longer able to find each other. However, common failback procedures to guarantee high availability of such critical services can be utilized to minimize such risks. Further advantages and disadvantages then result from the corresponding architectures.

If companies use their own internal, private (corporate) AAS Registries in addition to the public ones, this problem does not arise. In this case, it is possible to provide any number of AAS Registries in different architectures, just as in cross-company networks. In the case of a failure of the public AAS Registry, the corporate AAS Registries are still available. Furthermore, the question arises how the registration of the AAS in the public registries is done. On the one hand, there is the option that the AAS register directly in public AAS Registries in addition to registering in corporate registries, which means that they have to maintain several entries. On the other hand, corporate AAS Registries can share/synchronize their entries with public AAS Registries. The advantage here is that the AAS only have to maintain one entry and the registries take over the distribution task.

Therefore, corporate and public AAS Registries can also be operated as decentralized AAS Registries. It is, however, of critical importance that confidential information is treated accordingly, as especially decentralized registries with components controlled by other companies make it challenging or nearly impossible to subsequently recall information once it has been published. In addition, non-technical aspects come into play as well, including geo-politics, multi-country regulations, etc., which also need to be considered.

3.5.2 Recommended Concept for AAS Registries

Due to the scalability and distribution nature of Industry 4.0 applications, the existence of one global central registry, or several central registries disconnected from their environment, is not feasible (**Recommendation 1**). The number of envisioned AAS and Submodels, and thus their respective Descriptors, will further make it very challenging if not impossible for any client to efficiently query all or an adequate subset of all known registries at.

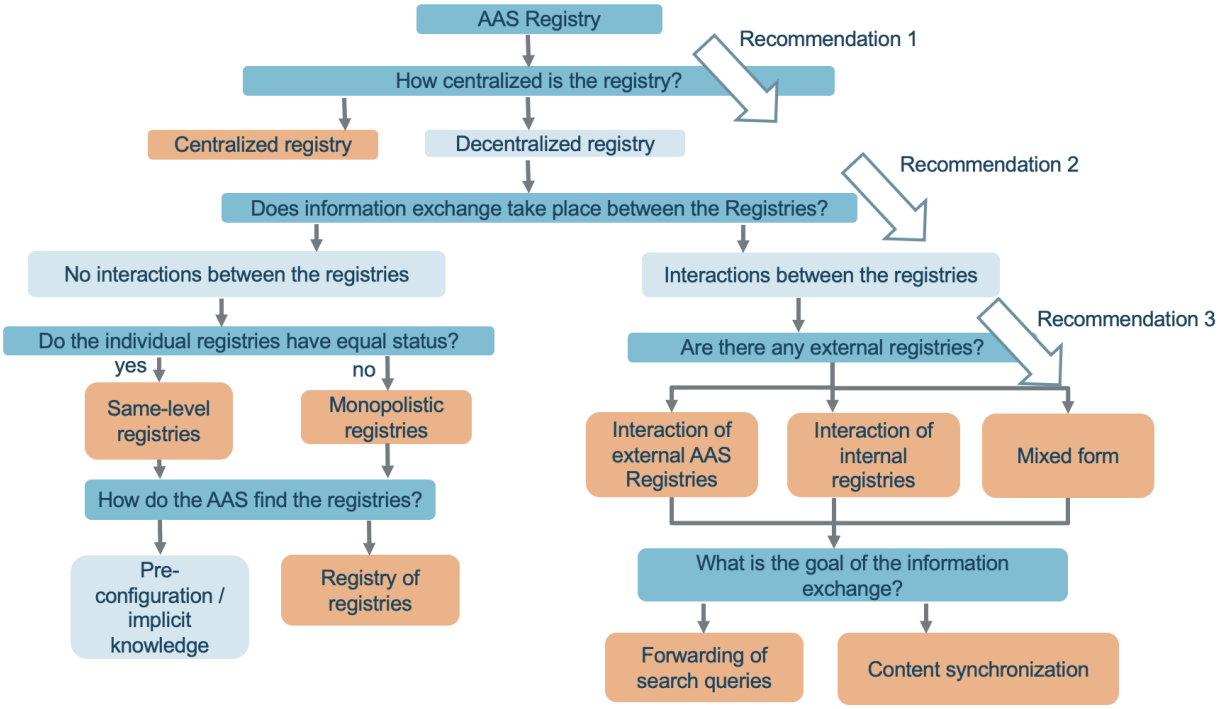

Figure 16: Recommended concept for registries based on architectural considerations

In some constellations, the decentralized registries in a logical ecosystem therefore have to exchange their set of Descriptors in advance and pre-process and index them efficiently (**Recommendation 2**).

It must be allowed to exchange data both between AAS Registry instances within the same I4.0 Data Space, IT landscape, or organization and across them, plus further include pure externally hosted AAS Registries (**Recommendation 3**). In any other case, the client will be unaware of relevant Descriptors, whose (non-)existence cannot be made transparent in such a setting.

The architecture decision regarding whether the AAS Registries shall forward the requests or their complete content depends mainly on the expected number and size of the Descriptors and the required response times for the clients.

It is to be expected that in high-volume setups, the continuous exchange of all Descriptors will allocate too many resources and become too expensive. In such cases, the generally longer response times of – synchronously – forwarded requests present the smaller issue.

If, however, short processing time is the critical factor, the answering registry instances need to prepare complete datasets and indexes thereof, utilizing read-optimized data structures possibly deployed in scalable clusters.

However, it can be expected that discovery requests are relatively rare events compared to, for instance, read or write operations to AAS themselves. This is because AAS and Submodels are read or updated significantly more often than they are created or moved to different repositories; a client can therefore cache their endpoint and only needs to check the Descriptors if a previously unknown AAS or Submodel is required.

Following this argumentation, the relevance of the response time will generally be lower than the issues caused by the high number of AAS and Descriptors. Nevertheless, future AAS implementation scenarios

might turn out to be different. Therefore, no appropriate recommendation is possible at this position in the graph.

3.5.3 Possible Roadmap to a Decentralized AAS Registry

The development and operation of a I4.0 Data Space-wide public AAS Registry is a valid first step towards implementing Industry 4.0 use cases in the short term. The long-term vision of Industry 4.0 assumes a decentralized character of the system. As described in chapter 3.2, the establishment of several company-specific and de facto centralized registries will not achieve any positive networking effects, since the AAS and AAS users located in an I4.0 Data Space will be separated through the registries. This can lead to the creation of a monopoly through the exclusive or preferential use of one of the registries. Alternatively, AAS and AAS users could register in many AAS Registries and manage multiple Descriptors. This would lead to increased organizational effort and is not recommended.

Therefore, the next step is the establishment of a Registry of Registries system. Each I4.0 Data Space operates or appoints one public AAS Registry that serves the Descriptors of the other external AAS Registries.

This approach allows an automated procedure for the search queries in the system. The client approaches the Registry of Registries first, discovers a suitable set of AAS Registries, and continues its search there. This pattern requires less implicit knowledge and can be an effective mechanism to prevent the formation of monopolistic AAS Registries.

As a side effect, the Registry of Registries forms a single point of failure and potential gatekeeper in the structure of the system and thus introduces unwanted risks. As a result, some negative aspects of the centralized approach, such as misuse of the position or manipulation of the requests, appear again.

A possible goal of the community for a later phase, which requires further research and specification work, could be the development of a concept of interacting AAS Registries that either exchange and forward search requests to other AAS Registries or synchronize their content (Figure 17).

Figure 17: Possible roadmap to a decentralized registry

4 Appendix 1 - Sources of Inspiration

This document **does not provide any novel approaches** to creating decentralized registries. In one form or another, these approaches can be found in a variety of existing and established technologies. This section outlines some of these technologies and arranges them into the developed taxonomy (Figure 18).

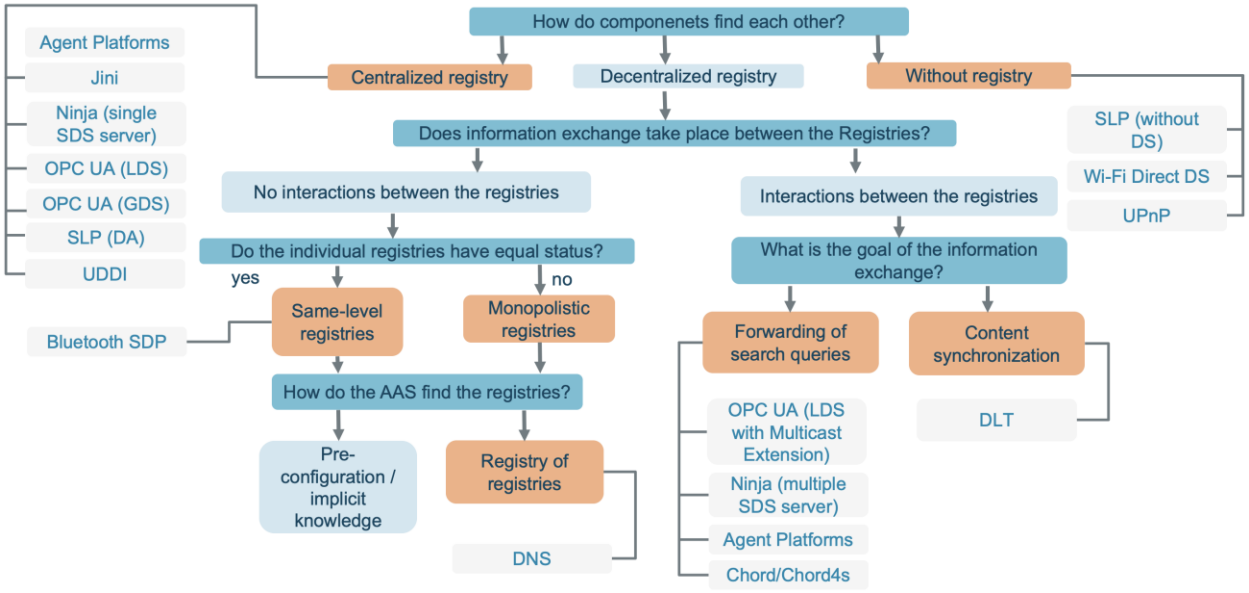

Figure 18: Classification of registry design principles from different technologies into the taxonomy of decentralized registries [4]

These technologies and the technology-specific design principles for the registry can serve as inspiration in the further course of the specification work for the AAS Registry and can possibly be adopted or adapted.

This overview has an impulse character and can serve as a basis for further discussions as well as for the concept and specification development of the decentralized AAS Registry.

4.1 Multi-Agent Systems

Agent platforms are designed to provide a physical infrastructure on which agents are distributed and to facilitate flexible agent or service discovery in the agent ecosystem. FIPA's preferred architecture for an agent platform includes a component called Directory Facilitator [5], which in the context of this document can be referred to as an agent registry or agent discovery. The FIPA specifications mention that it is possible to design decentralized networks of agent platforms, or directory facilitators, but not to specify exact concepts. This probably means that for smaller, manageable registry networks, a use-case specific solution would be sufficient. Since the agent doctrine does not explicitly offer decentralized kin concepts, the approach specified by FIPA is considered a centralized approach in the context of this document.

4.2 Bluetooth Discovery Service

The Bluetooth system specification provides a Service Discovery Protocol (SDP), which offers the possibility to discover available services and determine their properties [6]. The protocol defines interactions between an SDP client and an SDP server to discover services. The set of servers available to a client, changes dynamically based on the proximity of the server and client. The fact that there can be multiple servers that do not interact with each other allows this technology to be classified with the independent equal registry approaches.

4.3 Chord/Chord4S

Chord is a decentralized P2P approach to uniform data distribution and efficient query routing. The system consists of a set of distributed nodes that form a structured network. Service requests can be submitted to any node, which then forwards them if it cannot answer them. More information about Chord and the new, improved version Chord4S can be found in [7] and [8]. Since there are several forms of registries interacting with one another and forwarding information, this technology is classified as part of interacting registries.

4.4 Distributed Ledger Technology (DLT)

DLT is a decentralized transaction register in which network participants with equal rights manage exact copies of this transaction register. New transactions are distributed throughout the network and kept up to date. Since this is a type of association of equal registers with a synchronization mechanism, this technology is classified as an interacting synchronizing registry.

4.5 Domain Name System (DNS)

The Domain Name System (DNS) is used to convert symbolic names into an IP address, or vice versa, when communicating via the Internet. The address space is structured hierarchically, and the various sections (domains) have their own DNS servers, which are responsible for the assignment in the relevant address space section. A DNS server is a program that responds directly to queries regarding name resolution or forwards these queries up or down the hierarchically structured address tree [9]. In the context of this document, the DNS is thus to be understood as a hierarchically structured network of registries for registries.

4.6 Ninja Service Discovery Service

The Ninja project is about a software architecture for online support services. Its goal is to provide an infrastructure for the use of services in a scalable, fail-safe, and affordable way. The key component is the Secure Service Discovery Service (SDS), which makes it possible to display available and running services. To find specific services, clients perform a descriptive query to the SDS server, which contains descriptions of available services. The latter responds to the client queries [10]. If the service load exceeds a threshold, a subordinate server is started, to which a part of the load is then transferred. Thus, a kind of server hierarchy can be created.

4.7 OPC UA with Local/Global Discovery Server

OPC UA provides a platform-independent, service-oriented architecture for synchronous and asynchronous communication. It follows the client-server paradigm and enables clients to acquire information from servers through a set of discovery services. These discovery services are implemented by dedicated discovery servers [11]. These are applications that manage a list of OPC UA servers available on the network and provide mechanisms for clients to obtain this list.
The following discovery servers offer applications the possibility to discover previously registered OPC UA applications.

- The Local Discovery Server (LDS) manages discovery applications for all applications that have registered with it and are available on the same host
- The LDS may also include a Multicast Extension (ME), which allows it to share information about registered servers with other LDSs.
- With the help of the Global Discovery Server (GDS), the client can search for servers in the management domain.

4.8 Service Location Protocol (SLP)

The SLP is a decentralized service discovery protocol that allows devices to search for services on a local network without prior configuration. Three different roles exist for participating devices. They can assume

one of these roles: The User Agent (UA) represents devices searching for services; the Service Agent (SA) offers and announces one or more services; and the Discovery Agent (DA) contents a list of registered SA [12]. The SLP can be implemented in two variants. On the one hand, there is the possibility to implement the protocol without the Discovery Agent. In this case of direct communication, multicast is used to query which participant fulfills the requirements of the User Agent. This approach can be understood as an architecture without registries. When using the Discovery Agent, the approach can be placed among the centralized registries in the taxonomy.

4.9 Universal Description, Discovery, and Integration (UDDI)

The UDDI is a specification that aims to publish, discover, and promote business-to-business integration of Web services on the Internet [13]. The core element of this technology is the Universal Business Registry (UBR), which serves as the main directory for all available Web services. Information such as business information, service information, and the technical model are described and stored there using the Web Services Description Language (WSDL). So, after the service provider has announced its services to the UBR, the service user can discover and use them there. Since this involves the use of a single registry, this architecture can be placed in the centralized approaches section of the taxonomy.

4.10 Universal Plug and Play (UPnP)

UPnP is considered an extension of Plug and Play and implements a set of protocols and related technologies for peer-to-peer services that allow devices on the network to automatically discover each other [10]. Devices can dynamically join the network, obtain an IP address, communicate their capabilities on demand, and learn about the presence and capabilities of other devices. UPnP uses the Simple Service Discovery Protocol (SSDP) to report the presence of a device to others and discover other devices or services. For this purpose, no central registry is used; rather, devices send multicast messages to announce themselves and to search for matching devices or services. Because of these properties, UPnP can be understood as an approach without a registry.

4.11 Wi-Fi Direct Service Discovery

Wi-Fi Direct, or Wi-Fi P2P, is a technology that enables device-to-device communication in wireless local area networks [14]. Devices are connected directly without an access point, either as a one-to-one network or a one-to-many network. P2P discovery is intended to allow devices to quickly find each other and establish a connection. To do this, this process consists of two steps. The first step, called Device Discovery, allows devices to exchange device information. This is followed by Service Discovery, which is used to query the services offered and information about them. Since this is an architecture with direct communication without any kind of registry, this approach is classified among the approaches without a registry.

4.12 SOA Repository Artifact Model & Protocol (S-RAMP)

The SOA Repository Artifact Model & Protocol (S-RAMP) defines a common data model for SOA repositories and a protocol for common tooling and data sharing. It defines a hierarchical classification system based on OWL and a query language based on XPath 2.0 aimed at enhancing interoperability across service lifecycle phases. It stores artifacts such as document and logical model as well as derived and extended artifacts.

5 Appendix 2 - Current Implementations of AAS Registries

AAS Registries and their potential implementations are being discussed in multiple working groups and projects focusing on Digital Twins. As such different proposals are on the table, some of them have experimented with actual implementations of the concepts. In this appendix, an indicative list of such efforts in different states of maturity is provided in order to enable a more holistic discussion on registries that goes beyond concepts and involves actual experimentation.

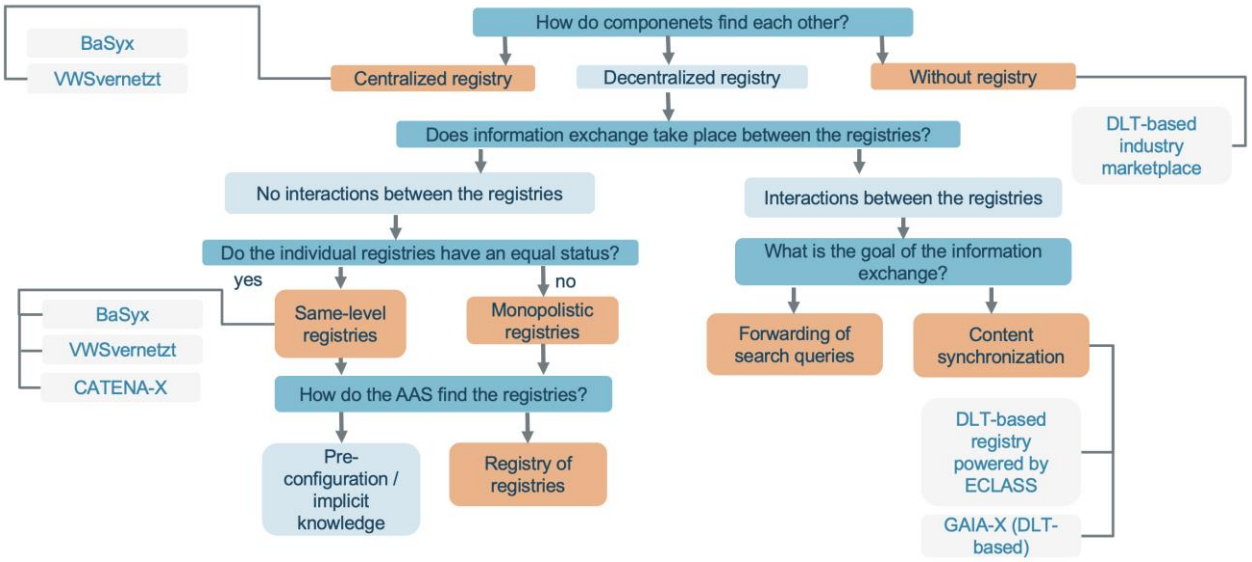

Figure 19: Arrangement of existing AAS Registries and AAS Registries under development in the taxonomy

5.1 BaSyx

The Eclipse open-source project Eclipse BaSyx [15] is being developed in the context of the BaSys 4 projects (BaSys 4.0, BaSys 4.2, BaSys ÜberProd, and BaSys4Transfer) which are funded by the German Federal Ministry of Research and Education (BMBF). BaSyx provides various SDKs and off-the-shelf components for the AAS infrastructure. It has implemented a centralized registry as a single containerized component that can be used with different types of backends [16]. Specifically, the SQL and MongoDB backends persist registry entries, while the InMemory backend stores the registry entries in RAM and hence is not persistent (considered for testing only). Additionally, the BaSyx AAS registry offers multiple features like MQTT eventing and authorization. Furthermore, BaSyx provides a TaggedDirectory, extending the registry API with the possibility to tag AAS and Submodels and retrieve them based on those tags. The implementations are available in multiple programming languages, for example Java or C#.

5.2 VWSvernetzt

The Working Group 1 "Reference Architectures, Standards and Norms" of the Plattform Industrie 4.0 intends to establish a Digital Twin-based Data Space. The AAS is the detailed specification of the Digital Twin of the Plattform Industrie 4.0. It requires an infrastructure, which, among other things, consists of an authentication server and a registry. For this purpose, the specification of the AAS defines an interface for a registry as well as a so-called descriptor, which contains the registration-relevant information [17]. The descriptor contains the endpoint of the AAS as well as the endpoints of the Submodels within this AAS.

The project "VWSvernetzt", which is funded by the German Federal Ministry for Economic Affairs and Climate Actions, supports the ongoing AAS specification work with testbeds and demonstrators. An implementation of the registry is one of the results used in demonstrators and in the testbed. The current implementation complies with the centralized registry pattern.

5.3 Gaia-X Federated Catalogue

The Gaia-X specifications introduce a technology-independent registry and discovery service for Gaia-X-compliant data assets and services called the Federated Catalogue, a component of the Gaia-X Federation Services GXFS). Defined in version 1.0 of the GXFS specification documents, the Federated Catalogues follow the pattern of ‘equally empowered registries’, which synchronize their content without a central mediator. Clients, both registering and searching ones, interact with the Federated Catalogue through a REST-inspired API and a graph-based query language for more expressive search requests.

The catalogue-to-catalogue data synchronization is intended to either leverage a peer-to-peer pattern or read/write the data content to **a Distributed Ledger**. However, as of November 2022, the final synchronization patterns still needed to be specified.

As the main data objects are participant and service self-descriptions instead of AAS Descriptors, AAS Registry entries cannot be managed natively by Federated Catalogues. Even though both AAS elements and Gaia-X self-descriptions can be serialized in JSON-LD, the respective API operations and schema checks focus on different structures and AAS Descriptors are therefore not accepted by Federated Catalogues without non-standardized nestings. For instance, a client may send a Gaia-X-compliant service description including AAS Descriptors as additional – optional – annotations. Other users of the Federated Catalogues might, however, not understand them and thus ignore them.

The reference implementation of the GXFS Federated Catalogue is available as an open-source GitLab project (available online: https://gitlab.com/gaia-x/data-infrastructure-federation-services/cat). The core server uses the SpringBoot framework, with community versions of PostgreSQL and Neo4J databases, as well as a keycloak instance for user authentication.

5.4 Catena-X

The Catena-X ecosystem proposes providing company-specific, internally hosted AAS Registries exposed through so-called connectors. These connectors act as the single-entry points to the Catena-X Data Space and manage control flows and negotiations. They also handle incoming communication from other Data Space participants and organize the outgoing communication. Company-internal AAS Registries manage and store AAS Descriptors, access to which is managed through the connector based on attribute-based access control contracts.

The company-internal architecture of the registries is the responsibility of each company. Hierarchical, peer-to-peer, or central architectures are possible. The Data Space-wide discovery of AAS leverages the connector discovery mechanism. Therefore, no Data Space-wide AAS Registry is required but the existence and location of those exposed by each company can be found through a multi-step process. First, the searching connector requests the endpoints of all connectors participating in the Data Space, goes through its locally indexed connector index, and extends it, if needed, through its crawling capabilities and filters for AAS Registries. Second, the connector requests the AAS Descriptors from all listed registries.

This means that the applied architecture pattern conforms to ‘equally-empowered registries’ without registry-to-registry data exchange. It is the responsibility of the clients to find all relevant registries, organize access, and merge the descriptor information. The Catena-X-wide lists of connectors acts as a Registry of Registries, even though they do not manage registry metadata but the metadata of the connectors. Therefore, it introduces one further discovery step. Federation of queries might be added in the future, as well as more efficient look-up methods.

5.5 DLT-based Decentralized Registry Powered by ECLASS

The DLT can serve as inspiration for the implementation of synchronizing registries. Each DLT has two essential components that should be considered in the context of this document. These are the distributed network itself (can be considered as an analogy to a network of registries) and the set of rules and the necessary interactions to ensure that the data newly added to one of the registries is distributed throughout

all registries within a defined time and in a tamper-proof manner (can be stated as a research demand as indicated in chapter 3.4.1.2).

These aspects are relevant for the context of this document since, if an attempt is made to define the synchronizing registries and the algorithms necessary for the synchronizing registries themselves, strictly speaking, such an attempt means that one is developing one's own DLT. The development of completely new solutions does not appear to be a feasible approach to the authors since this would require a great amount of effort and will not necessarily lead to success.

Instead, it is advisable to rely on the existing research that has already been developed and take inspiration from other DLT technologies or adopt the design principles.

A Proof of Concept (PoC) that implements synchronizing registries based on DLT has been developed in the project "Distributed Ledger-based Infrastructure for Industrial Digital Twins" supported by ECLASS and involving research institutions and industry players. The developed prototype (available online: https://aasRegistry.eclass.eu) implements the basic use case, namely registration and finding the AAS based on an Asset ID. The PoC also shows further use cases such as owner change and decentralized identity management [18].

5.6 DLT-based Decentralized Industry Marketplace

The concept of the decentralized DLT-based industry marketplace does not include any separate components that take over a registry function. The concept assumes that the network (Data Space) participants are interconnected via such a communication medium so that a service search query can directly reach the potentially interested service providers and they can then send their service proposals directly to the requester. Accordingly, there is no need for the intermediate step, namely the inquiry of the service registry about the list of registered service providers and the establishment of a connection with each of them. Thus, the separate logical component that performs the function of registering individual AAS (service provider) becomes obsolete. In this context, the DLT is considered as a technology that can serve as such a global communication medium.

An example from everyday life illustrates the concept: One could envision a decentralized Uber in which the search queries directly reach cab drivers or self-driving vehicles and where these can independently accept transportation orders or submit offers, without needing [9] the intermediary platform that would otherwise assign the orders to the drivers.

The concept was implemented as a PoC in the IOTA-based Industry Marketplace [19].

6 References

[1] P. I4.0, "Der Datenraum Industrie 4.0," BUNDESMINISTERIUM FÜR WIRTSCHAFT UND ENERGIE (BMWI), Berlin, 2021.

[2] A. Belyaev, J. Hasler, C. Urban and C. Diedrich, "Architektonische Gestaltungsprinzipien einer dezentralen Industrie 4.0 Infrastruktur," in EKA 2022 - Entwurf komplexer Automatisierungssysteme, Magdeburg, 2022.

[3] C. Lerch, Die volkswirtschaftliche Bedeutung von digitalen B2B-Plattformen im Verarbeitenden Gewerbe, Berlin: BMWiE, 2019.

[4] J. Hasler, "Masterarbeit: Taxonomie und architektonische Gestaltungsprinzipien von dezentralen Registries," Otto-von-Guericke-Universität Magdeburg, Magdeburg, 2023.

[5] Foundation for Intelligent Physical Agents (FIPA), "FIPA Agent Management Speci- fication," 2004.

[6] Bluetooth SIG, "Specification of the Bluetooth System," 2001.

[7] H. Qiang, "A Decentralized Service Discovery Approach on Peer-to-Peer Net- works," *IEEE Transactions on Services Computing 6.1,* 2013.

[8] T. Sivaramaprasad and M. Parameswar, "Chord4S: A Peer-to-Peer-Based Network Approach forDecentralized Service Discovery," *International Journal of Electro- nics Communication and Computer Engineering,* 2015.

[9] C. Meinel and H. Sack, Internetworking., Berlin : Springer , 2012.

[10] J. Moorman, J. Lockwood and S.-M. Kang, The State of Service Protocols, 2000.

[11] OPC Foundation, "OPC 10000-12: OPC Unified Architecture - Part 12: Discovery and Global Services," 2018.

[12] E. Guttman, Service Location Protocol, Version 2, 1999.

[13] OASIS, UDDI Specification 3.0.2, Luc Clement, 2004.

[14] A. K. Muhammad, "Wi-Fi Direct Research - Current Status and Future Perspectives," *Journal of Network and Computer Applications 93,* 2017.

[15] [Online]. Available: https://www.eclipse.org/basyx.

[16] [Online]. Available: https://wiki.eclipse.org/BaSyx_/_Documentation_/_Components_/_Registry.

[17] P. I4.0, "Details of the Asset Administration Shell - Part 2," BMWiK, Berlin, 2022.

[18] A. Dogan, "Distributed Ledger-based Infrastructure for Industrial Digital Twins," ECLASS, 2020.

[19] A. Belyaev, C. Diedrich, H. Köther and A. Dogan, "Dezentraler IOTA-basierter Industrie-Markplatz," *Industrie 4.0-Management,* vol. 2, no. 20, pp. 36-40, 2020.

[20] [Online]. Available: https://app.swaggerhub.com/apis-docs/Plattform_i40/Entire-API-Collection/V1.0RC03#/. [Accessed 8 12 2022].

www.industrialdigitaltwin.org