

# From IoT Servitization to IoT Assetization

Zakaria Maamar<sup>1</sup><sup>a</sup>, Amel Benna<sup>2</sup><sup>b</sup>, Vanilson Buregio<sup>1</sup><sup>c</sup> and David Alves<sup>3</sup><sup>d</sup>

<sup>1</sup>University of Doha for Science and Technology, Doha, Qatar

<sup>2</sup>Research Center for Scientific & Technical Information (CERIST), Algiers, Algeria

<sup>3</sup>Department of Computing, Federal Rural University of Pernambuco, Recife, Brazil

Keywords: Asset, IoT, Service, Management.

Abstract: This paper discusses the stages and techniques to guide the conversion of IoT-compliant things into assets and then, services. While existing initiatives focus on the conversion of things into services as part of the servitization process, there are not initiatives that focus on thing conversion into assets as part of the assetization process. Assetization permits to model things from a management perspective in terms of depreciation over time, transferability across locations, disposability after use, and convertibility across platforms. Thanks to assetization, things would provide economical, informational, operational, and regulatory benefits to their owners (whether moral or juridical). A system demonstrating the technical doability of thing assetization is also discussed in the paper.

## 1 INTRODUCTION


In today's ICT landscape, Internet-of-Things (IoT) perfectly demonstrates Weiser's definition of ubiquitous computing when he states in 1999 that "*the most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it*" (Weiser, 1999). IoT has become ubiquitous where 41 billion IoT devices are predicted to exist by 2027 and 70% of vehicles are predicted to be connected to the Internet by 2023<sup>1</sup>.


In line with IoT growing research interest, we presented in (Maamar et al., 2021b) an IoT-servitization framework exposing things as services to potential users. On top of this exposure, the framework addressed 3 concerns undermining IoT adoption: lack of cognitive things, lack of semantics between things, and thing confinement into silos. Servitization has been around for many years (Vandermerwe and Rada, 1988) where organizations like Rolls-Royce sells power-by-the-hour and salesforce.com offers pay-per-use software on the cloud. Nowadays, everything-as-a-service (XaaS) is a well-established operation


model (Duan et al., 2016).


Despite the role of servitization in sustaining IoT, the service model that underpins any form of servitization is tightly coupled to the request-response pattern; upon receiving requests, services direct them in our case to things and then, collect responses back, if necessary. Although this service model has proven itself over the years from an interaction perspective, aspects from a management perspective like thing depreciation over time, thing transferability across locations, thing disposability after a time-period/use, and thing convertibility across platforms are overlooked. To address the management-aspect overlook, we advocate in this paper for IoT assetization in order to encapsulate things into assets, a term commonly used in the finance world. The basis of our thing-assetization proposal is an asset model that would cater for the specificities of IoT like physical *versus* digital thing, online *versus* offline thing, etc.

An asset "*is something that provides a current, future, or potential economic benefit for an individual or other entity*"<sup>2</sup>. When things are assetized, they provide economical benefits (generates money like parking meter), informational benefits (produces data like sensor), operational benefits (performs operations like AC unit), and regulatory benefits (enforces policies like road radar) to their owners (whether moral

<sup>a</sup> <https://orcid.org/0000-0003-4462-8337>

<sup>b</sup> <https://orcid.org/0000-0002-9076-5001>

<sup>c</sup> <https://orcid.org/0000-0001-5122-6075>

<sup>d</sup> <https://orcid.org/0009-0008-2729-4068>

<sup>1</sup>[www.businessinsider.com/internet-of-things-report](http://www.businessinsider.com/internet-of-things-report).

<sup>2</sup>[www.investopedia.com/terms/a/asset.asp](http://www.investopedia.com/terms/a/asset.asp).

or judicial). To achieve these benefits, an asset cost-model could be used to track the “costs” of managing a thing as an asset. These costs could vary depending on things’ afore-mentioned specificities. In this paper, we present our IoT assetization framework in terms of motivations, guidelines, and demonstration. Section 2 defines asset. Section 3 details the assetization framework in terms of motivations, thing, asset, and service modeling, and guidelines. Section 4 implements the framework. Section 5 introduces future work. Finally, Section 6 concludes the paper.

## 2 ASSET OVERVIEW

The International Financial Reporting Standards framework states that “*an asset is a resource controlled by the enterprise as a result of past events and from which future economic benefits are expected to flow to the enterprise*” (Team, 2022a). Being a multi-disciplinary topic, we provide hereafter an overview of how asset is defined and perceived in 3 separate communities namely, ICT:security, digital-right management, and finance.

In the ICT:security community, Kuzminykh and Carlsson adopt assets to isolate IoT system risks (Kuzminykh and Carlsson, 2018). Recognizing IoT systems’ unique characteristics and requirements, the authors’ threat-risk modeling approach identifies security stakeholders (i.e., devices, services, and customers), security assets, possible attacks, and, finally, threats for the concerned IoT system. In the same community, Chehida et al. present a methodology to identify security risks, assess these risks’ impacts on IoT systems’ assets, and, finally, protect these assets from these risks. Along with assets upon which a security-risk assessment strategy could be built, services and business processes could help define a similar strategy. Finally, Matsumoto et al. discuss how important cyber-security countermeasures are for the Industrial Internet of Things (IIoT) and how different IIoT is from IoT requiring specialized means to manage assets. The authors’ assets correspond to edge devices being monitored by an asset configuration management agent.

In the digital-right management community, the objective is to ensure a proper and trusted use and protection of digital media assets and files in the age of social media and widespread file sharing. A well known W3C specification for this management is ODRL. It expresses that “*something is permitted, forbidden, or obliged, possibly limited by some constraints*” (W3C, 2018; Becker et al., 2013) and provides a flexible and interoperable information model,

vocabulary, and encoding mechanisms to represent statements about assets’ uses. An asset is an identifiable resource or a collection of resources such as data/information, content/media, applications, and services. On top of ODRL, Custers questions the appropriateness of existing digital-right management practices for today’s world due to the heavy reliance on social media and adoption of advanced technologies like big data and IoT (Custers, 2022). According to Custers, should not the ICT community define new fundamental rights for the digital era? Finally, in the same community of digital-right management, IBM defines digital asset management solution as “*a software and systems solution that provides a systematic approach to efficiently storing, organizing, managing, retrieving, and distributing an organization’s digital assets*”<sup>3</sup>.

In the finance community, asset is any resource that an organization owns or controls as part of its daily operations to produce positive economic value (O’Sullivan and Sheffrin, 2021). According to the Corporate Finance Institute (Team, 2022a), the properties of an asset include ownership (can be eventually turned into cash and cash equivalents), economic value (can be exchanged or sold), and resource (can be used to generate future economic benefits). And, in term of classification assets can be referred to as convertibility (how easy they are converted into cash), physical existence (tangible *versus* intangible), and usage (based on their business operation usage/purpose).

The paragraphs above offer a summary of the multiple uses of assets making them a cornerstone to any organization’s development strategy. At the end of the day, whether tangible or intangible, physical or digital, limited or unlimited, etc. we expect assets to provide benefits to their owners. In the remaining sections, we elevate things to assets enabling a process of identifying and then, describing the management aspects of these things.

## 3 ASSETIZATION FRAMEWORK

After motivating thing assetization, we model the assetization framework’s elements. Then, we present the guiding stages for a successful assetization.

### 3.1 Motivations

Today’s things suffer from several limitations like reduced size, restricted connectivity, limited energy,

<sup>3</sup>[www.ibm.com/ae-en/topics/digital-asset-management](http://www.ibm.com/ae-en/topics/digital-asset-management).

and constrained storage, which undermines their integration opportunities into mission-critical applications, for example. In addition, existing specifications, e.g., W3C Web of Things-Thing Description (WoT-TD, (W3C, 2020)), primarily focus on things' technical aspects like whether they are physical or digital along with what interaction and security protocols they support. Finally, thing servitization develops proxy services on top of things neglecting how to manage these things. With thing assetization, we tap into some asset management principles<sup>4</sup> like value added/level of service, lifecycle, and total cost of ownership to handle the following IoT-related management aspects:

- Depreciation is related to the (selling) value and/or (ongoing) performance of assets in the long run. In IoT, we foresee depreciation as a concern when things are used outdoor in harsh conditions, which would over time negatively impact their performance, for example. In addition, having things installed outdoor would likely expose them to physical tampering, which could accelerate their depreciation.
- Transferability is related to ownership change of assets from a legal perspective. In IoT, we foresee transferability as a concern when things shift from one location to another forcing them to comply with new data-privacy regulations, for example. The way we foresee transferability is not in line with Gunnarsson and Gehrman who focus on security requirements during ownership change (Gunnarsson and Gehrman, 2022). They note that backward and forward secrecy with respect to old and new owners should be preserved. An IoT system's new owner shall not deduce anything that the old owner did before the ownership transfer, and *vice versa*. Gunnarsson and Gehrman also mention that there should not be any time period where an IoT system would be under the simultaneous control of both owners.
- Disposability is related to the availability of assets. In IoT, we foresee disposability as a concern when things cannot be used after a date or a number of times, which could result in replacing them. To extend things' lifetimes, measures like regular servicing could be planned.
- Convertibility is related to the monetization of assets, i.e., how easy it is to turn an asset into cash. In IoT, we foresee convertibility as a concern when things cannot adjust/mutate because of changes of their environments. In (Maamar et al.,

<sup>4</sup>[www.nexgenam.com/blog/the-principles-of-asset-management](http://www.nexgenam.com/blog/the-principles-of-asset-management).

2021a), Maamar et al. discuss 3 arguments supporting thing mutation: performance, so things remain competitive; awareness, so things "know" their contextual surroundings; and survivability, so things remain in business.

It is worth noting that the management aspects above should offer a better return-on-investment on things based on the following benefits:

- Economical benefit is about money that organizations could make from things. Some things like parking meters generate direct moneys, while others like sensors generate indirect moneys since their owners could potentially sell the collected data to third parties.
- Information benefit is about data that organizations could obtain from things. Some things like sensors generate direct data, while others like actuators generate indirect data after processing the data that they would receive.
- Operational benefit is about the commands that organizations could exercise on things. Some things like electric switches respond to direct commands originating from users, while others like fire sprinklers respond to indirect commands originating from third parties like detectors.
- Regulatory benefit is about the use of things to enforce organizations' policies. Some things like road radars directly enforce speed limits by flashing violating vehicles, while others like automatic door locks indirectly enforce accesses based on signals received from face recognition programs embedded into cameras.

### 3.2 Modeling Things

For the sake of illustration, we use WoT-TD to model things. An excerpt of WoT-TD information-model<sup>5</sup> is presented in Fig. 1 where *Thing* representing a concrete thing is a core class supporting 3 types of interactions with end-users and/or peers. All classes associated with these interaction types are derived from the *InteractionAffordance* class and correspond, respectively, to *PropertyAffordance* that allows to sense and control parameters, *ActionAffordance* that refers to a thing's operations, and *EventAffordance* that allows to asynchronously push communications such as notifications, discrete events, and streams of values to receivers. Regarding a thing's security mechanism and resources, 2 relevant classes exist namely, *SecurityScheme* and *Link*. To describe how a thing's operations are exercised over *properties*, *actions*, *events*,

<sup>5</sup>[www.w3.org/TR/wot-thing-description](http://www.w3.org/TR/wot-thing-description).

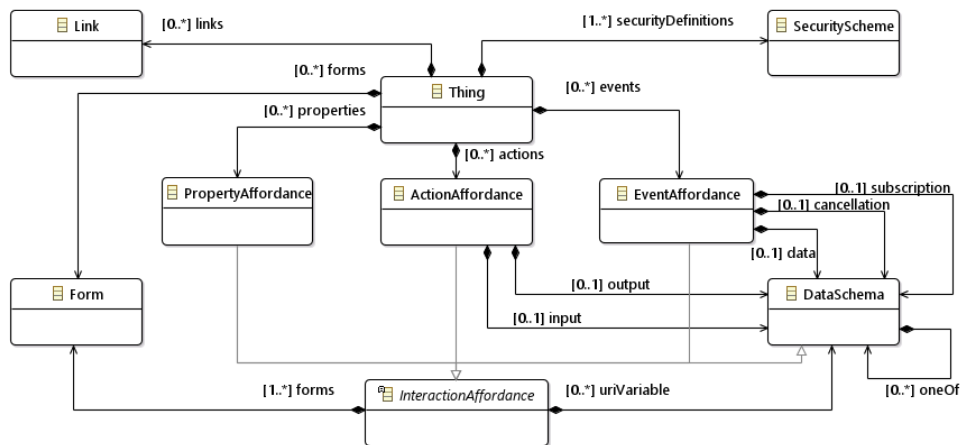


Figure 1: Excerpt of WoT-TD information-model.

and the thing itself, protocol bindings are serialized and represented with the *Form* class. Finally, the data format of an operation's inputs/outputs is described with the *DataSchema* class.

Listing 1 instantiates WoT-TD information-model for a thing denoted by lamp. It has an *id* (line 2), a title that is *MyLampThing* (line 3), and a basic security scheme (*basic\_sc*) included in *securityDefinitions* (line 4) requiring username and password (line 5). In term of interaction affordance, there exist one property affordance that is *status* (line 8) accessible via <https://mylamp.example.com/status> (line 10), one action affordance that is specified to *toggle* (line 12) the switch status available at <https://mylamp.example.com/toggle> as a resource (line 15), and, finally, one event affordance that is *overheating* (line 17) to allow *MyLampThing* to send asynchronous messages to concerned persons.

Listing 1: Excerpt of WoT-TD specification of MyLamp.

```

1 {"@context": "https://www.w3.org/2019/wot/td/v1"
2  "id": "urn:dev:ops:32473-WoTLamp-1234",
3  "title": "MyLampThing",
4  "securityDefinitions": {
5    "basic_sc": { "scheme": "basic", "in": "header"
6    },
7  "security": ["basic_sc"],
8  "properties": {
9    "status": {
10     "type": "string",
11     "forms": [{"href": "https://mylamp.example.com/status"}]},
12  "actions": {
13    "toggle": {
14     "op": "invokeaction",
15     "href": "https://mylamp.example.com/toggle"
16     },
17  "events": {

```

```

17   "overheating": {
18     "data": { "type": "string" },
19     "forms": [{
20       "href": "https://mylamp.example.com/oh",
21       "subprotocol": "longpoll"}]}]}

```

### 3.3 Modeling Assets

To assetize things, we design an asset static-model and an asset dynamic-model. The static model identifies constructs that capture the functional and non-functional characteristics of a future asset encapsulating a thing. And, the dynamic model identifies states that capture the lifecycle of the future asset when it becomes available for use. For design needs, UML notation is adopted to represent both models.

Fig. 2 is an asset's class-based static model where *asset* is a key class specialized into *physical* and *digital* sub-classes. For an organization, an asset provides benefits captured with the *benefit* class and specialized into informational, operational, regulatory, and economical. These benefits would depend on the asset's domain of use (e.g., hospitality) represented with the *domain* class. On top of representing the purpose and owner of an asset using *operation* and *owner* classes, respectively, how an asset is invoked and secured is regulated with the *protocol* class.

In Fig. 3 we represent an asset's state-based dynamic model where *activated* is the starting state having *put-to-use* and *depreciated* as 2 nested concurrent sub-states. When using an asset, it is automatically subject to depreciation. From: *put-to-use* state, the rest of states capture the following situations:

- To: *transferred* state corresponds to the situation where an asset is shifted from one place to another. This results in forcing the asset to comply

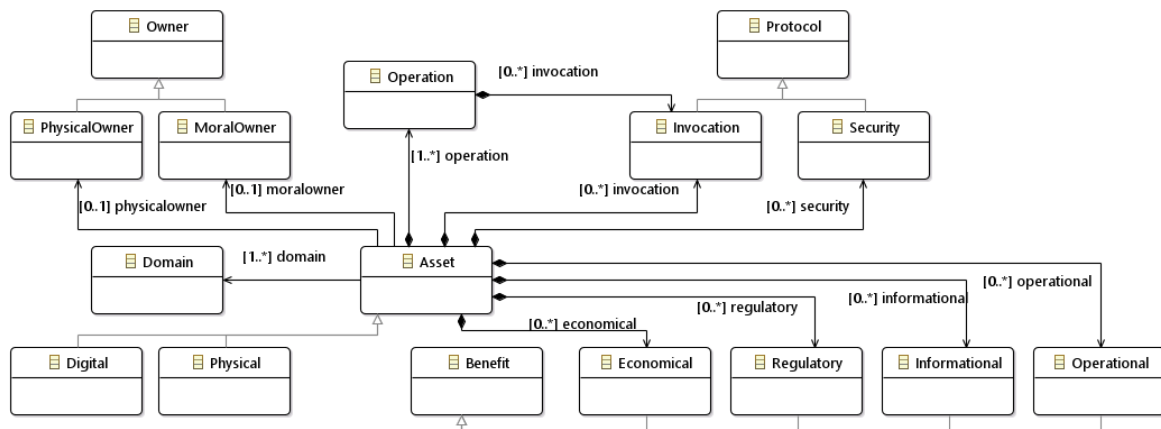


Figure 2: Asset's class-based static model in IoT.

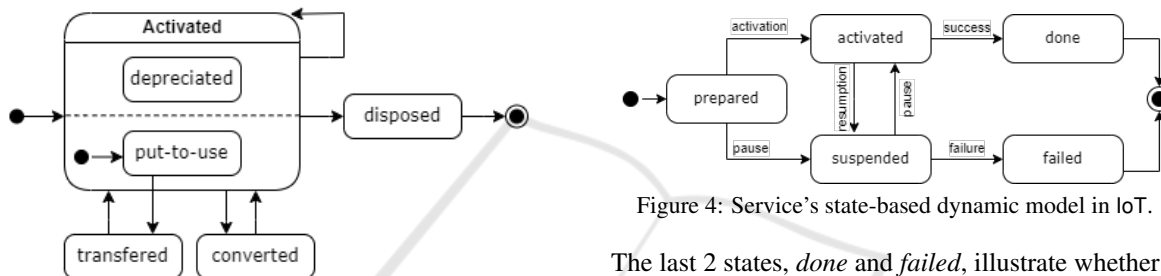


Figure 3: Asset's state-based dynamic model in IoT.

with the new place's regulations and/or to work under the new place's conditions.

- To:*converted* state corresponds to the situation where an asset is on-the-fly loaded with extra capabilities. This allows the asset to respond to changes in the environment without suspension.
- To:*disposed* state corresponds to the situation where an asset is withdrawn due to a fault.

From:*depreciated* state, the rest of states capture the following situations:

- To:*disposed* state corresponds to the situation where an asset is withdrawn due to out-of-existence triggered by excessive and/or long-time use.

Each situation listed above sheds light on a particular management aspect; e.g., from *depreciated* state to *disposed* state corresponds to the disposability management aspect, and from *activated* state to *converted* state corresponds to the convertibility management aspect.

### 3.4 Modeling Services

In Fig. 4 we represent a service's state-based dynamic model where *prepared* is the starting state having *activated* and *suspended* as 2 exclusive successor states.

Figure 4: Service's state-based dynamic model in IoT.

The last 2 states, *done* and *failed*, illustrate whether a service execution completed with success or failure, respectively.

### 3.5 Guiding Stages

Fig. 5 represents our 2 stage approach for thing assetization and asset servitization, respectively. These stages are in complete opposition to what we did earlier in thing servitization where things were directly encapsulated into services (Maamar et al., 2021b). Although Fig. 5 shows a bottom-up progress from the physical level (thing) to the digital level (asset) and finally, the abstract level (service) in support of thing assetization and asset servitization, this progress is also complemented by another top-down progress from the abstract level to the digital level and finally, the physical level demonstrating how both assetization and servitization are concretized at run-time. This happens by passing details from instantiating relevant constructs when specific details are passed on from one level to another. In the following, we decompose the discussions of the 2 stage approach into Bottom-Up (*BU*) and Top-Down (*TD*).

#### 3.5.1 Bottom-up Discussion

In the thing-assetization stage covering the physical and digital levels, the objective is to encapsulate a thing into an asset. This happens by identifying in

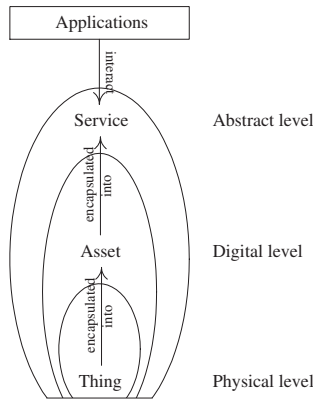


Figure 5: Thing assetization followed by asset servitization.

WoT-TD information-model the relevant constructs, i.e., classes and relations, that would semantically correspond to their counterpart constructs in Fig. 2’s asset static-model. This is summarized in Table 1 where  $c|r/x$  stands for class|relation/either *class’s name* or *relation’s name*( $class_i, class_j$ ). Should there be any missing correspondence (indicated by not-available in the table), then we will take corrective actions by adding new constructs to the asset static-model along with finalizing this addition, as we deem appropriate. Fig. 6 shows the new constructs in terms of classes (*event*, *state*, and *upgrade*) and relations (*event*, *state*, *upgrade*, and *invocation*) that become attached to Fig. 2’s asset static-model.

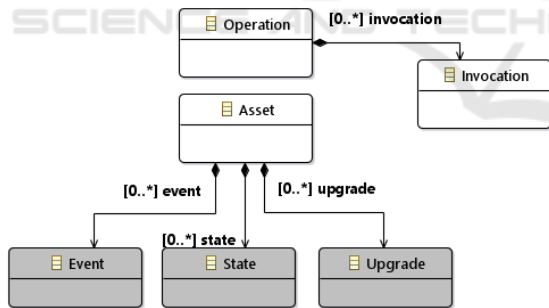


Figure 6: New classes/relations in the asset’s class-based static model.

We now continue with the asset-servitization stage covering this time the digital and abstract levels. The objective is to encapsulate an asset into a service that future applications would interact with. This happens by traversing the updated asset static-model (Fig. 2 and Fig. 6) to identify specific constructs that would semantically correspond to their counterpart constructs in Fig. 7’s service static-model. This identification is driven by the needs and requirements of the interface that would expose the service to applications for interaction. By analogy with Table 1, we proceed with the same as per Table 2.

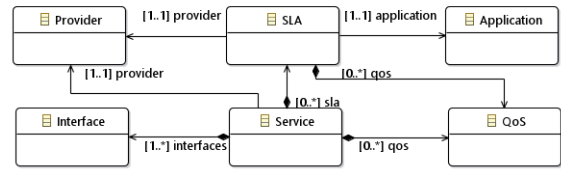


Figure 7: Service’s class-based static model in IoT.

### 3.5.2 Top-Down Discussion

Contrarily to the objective of the bottom-up asset-servitization stage that is to encapsulate an asset into a service, the objective of top-down is to cascade concrete details about applications interacting with services from the abstract level to the digital level. These details correspond to 2 types of values; those values that applications directly submit to interfaces when invoking them (e.g., interface’s *name*) and those values that are indirectly obtained based on what these applications submitted to these interfaces (e.g., service’s *id* is obtained based on interface’s *name*). In either way, cascading details requires identifying properties in the relevant constructs of both the service and the asset static-models that will act as value containers. This cascading is summarized with Table 3 where examples of properties in the respective service static-model include *id* to identify the service and *url* to interact with the interface of the service, and examples of corresponding properties in the asset static-model include *id* to identify the asset related to the service and *link* to associate the path of the operation to invoke with the *url* of the interface.

We now complete the thing-assetization stage where the objective now is to cascade concrete details from the digital level to the physical level allowing this time to effectively activate/trigger a particular thing based on specific constructs in the WoT-TD information model. These details refer to corresponding properties in the asset static model as per Table 3. Table 4 summarizes the detail cascading from the asset static-model to the WoT-TD information-model where examples of corresponding properties in this model include *href* to associate the submission path of the operation to perform with the invocation link and *input* to capture the operation’s input.

### 3.6 Handling Management Aspects

One of the objectives of the IoT-assetization framework is to manage things as assets. 4 management aspects along with this framework’s benefits are discussed in Section 3.1 and will hereafter be specified based on what has been achieved during bottom-up construct correspondences and top-down construct concretization.

Table 1:  $\mathcal{BU}$ :construct correspondences between WoT-TD and the asset static-model.

| WoT-TD information-model                      |                                       | Asset static-model's |   |
|---|---------------------------------------|----------------------|---|
| construct                                     | construct                             | construct            | corrective action                             |
| <i>c/Thing</i>                                | <i>c/Asset</i>                        |                      |   |
| <i>c/EventAffordance</i>                      | Not available                         |                      | Add <i>c/Event</i>                            |
| <i>c/PropertyAffordance</i>                   | Not available                         |                      | Add <i>c/State</i>                            |
| <i>c/ActionAffordance</i>                     | <i>c/Operation</i>                    |                      |   |
| <i>c/Form</i>                                 | <i>c/Invocation</i>                   |                      |   |
| <i>c/VersionInfo</i>                          | Not available                         |                      | Add <i>c/U pgrade</i>                         |
| <i>r/actions(Thing,ActionAddordance)</i>      | <i>r/operation(Asset,Operation)</i>   |                      |   |
| <i>r/events(Thing,EventsAddordance)</i>       | Not available                         |                      | Add <i>r/event(Asset,Event)</i>               |
| <i>r/properties(Thing,PropertyAddordance)</i> | Not available                         |                      | Add <i>r/state(Asset,State)</i>               |
| <i>r/forms(Thing,Form)</i>                    | <i>r/invocation(Asset,Invocation)</i> |                      |   |
| <i>r/forms(InteractionAffordance,Form)</i>    | Not available                         |                      | Add <i>r/invocation(Operation,Invocation)</i> |
| <i>r/versions(Thing,VersionInfo)</i>          | Not available                         |                      | Add <i>r/upgrade(Asset,U pgrade)</i>          |

Table 2:  $\mathcal{BU}$ :construct correspondences between the asset static-model and service static-model.

| Asset static-model                  |                                       | Service static-model's |                   |
|-------------------------------------|---------------------------------------|------------------------|-------------------|
| construct                           | construct                             | construct              | corrective action |
| <i>c/Asset</i>                      | <i>c/Service</i>                      |                        |                   |
| <i>c/Owner</i>                      | <i>c/Provider</i>                     |                        |                   |
| <i>c/Operation</i>                  | <i>c/Interface</i>                    |                        |                   |
| <i>c/Event</i>                      | <i>c/Interface</i>                    |                        |                   |
| <i>c/State</i>                      | <i>c/Interface</i>                    |                        |                   |
| <i>r/operation(Asset,Operation)</i> | <i>r/interface(Service,Interface)</i> |                        |                   |
| <i>r/event(Asset,Event)</i>         | <i>r/interface(Service,Interface)</i> |                        |                   |
| <i>r/state(Asset,State)</i>         | <i>r/interface(Service,Interface)</i> |                        |                   |
| <i>r/owner(Asset,Owner)</i>         | <i>r/provider(Service,Provider)</i>   |                        |                   |

Table 3:  $\mathcal{TD}$ :detail cascading from the service static-model to the asset static-model.

| Service static-model |               | Asset static-model  |                        |
|----------------------|---------------|---------------------|------------------------|
| construct            | property      | construct           | corresponding property |
| <i>c/Service</i>     | <i>p/id</i>   | <i>c/Asset</i>      | <i>p/id</i>            |
| <i>c/Interface</i>   | <i>p/name</i> | <i>c/Operation</i>  | <i>p/name</i>          |
| <i>c/Interface</i>   | <i>p/url</i>  | <i>c/Invocation</i> | <i>p/link</i>          |
| <i>c/Interface</i>   | <i>p/args</i> | <i>c/Operation</i>  | <i>p/input</i>         |

- Depreciation: in the literature (Department of the Treasury, 6pdf), many techniques to assess asset depreciation exist such as straight-line, declining-balance, and unit-of-production. Countries use these techniques, as they see fit; for instance, capital-cost-allowance in Canada<sup>6</sup> and straight-line and declining balance in the USA<sup>7</sup>. Despite

Table 4:  $\mathcal{TD}$ :detail cascading from the asset static-model to WoT-TD information-model.

| Asset static-model's |                | WoT-TD information-model  |                        |
|----------------------|----------------|---------------------------|------------------------|
| construct            | property       | construct                 | corresponding property |
| <i>c/Asset</i>       | <i>p/id</i>    | <i>c/Thing</i>            | <i>p/id</i>            |
| <i>c/Operation</i>   | <i>p/name</i>  | <i>c/ActionAffordance</i> | <i>p/properties</i>    |
| <i>c/Invocation</i>  | <i>p/link</i>  | <i>c/Form</i>             | <i>p/href</i>          |
| <i>c/Operation</i>   | <i>p/input</i> | <i>c/ActionAffordance</i> | <i>p/input</i>         |

<sup>6</sup>tinyurl.com/mr3sa4v9.

<sup>7</sup>tinyurl.com/dyhpv378.

the diversity of these techniques, they have some properties in common (Panicker, ions): original cost (asset's acquisition amount including sales tax, transportation, set-up, training, etc.), useful life (number of years one expects to have the asset in use depending on its type and is expected to be at least 1 year), and number of units produced before the asset is worn.

During thing assetization, depreciation as per Section 3.1 would require the following details about a thing: first-date-of-use, original-cost, level-of-use (e.g., number of invocation requests per day), number-of-years-of-use, and condition-of-use (indoor *versus* outdoor). These details are properties in the asset static-model and the values of these details are collected from WoT-TD-based thing specification. Table 5 illustrates potential correspondences between these properties and this specification's constructs. Should a correspondence be missing, i.e., reported as not-available in this table, then we would suggest corrective actions like done in Section 3.5.1.

- Transferability: in the literature (Heather et al., 2019), transferring assets is a multi-facet exercise that ranges from agreeing on what is eligible for transfer to preparing transfer agreements and identifying the necessary intervenants during the transfer. Both physical assets like lands (Patil and Shyamasundar, 2021) and virtual assets like cryptocurrency (Keundug and Heung-Youl, 2021) could be transferred requiring each a different handling mainly from legal and financial perspectives. Many details could be included in a transfer agreement with emphasis on price, assignee, rights and obligations relating to title, warranty, and indemnities<sup>8</sup>. An interesting discussion about asset transfer in IoT sheds light on shared own-

<sup>8</sup>www.contractsounsel.com/t/us/asset-transfer-agreement.

ership when both buyers and sellers retain control over the same assets, but in independent contexts (Raina and Palaniswami, 2021). Despite being sold to buyers, Tesla cars' software can be remotely altered over the Internet by a few keyboard-clicks at Tesla Inc.'s Palo Alto control center raising the question of who owns what.

During thing assetization, transferability as per Section 3.1 requires the following details about a thing: previous-location, current-location, and transfer-date. By analogy with depreciation, Table 6 illustrates potential correspondences between the asset class-model's properties and WoT-TD information-model's constructs. The same vocabularies and notation are adopted as earlier.

- **Disposability:** in the literature (FreshBooks, 2022; Team, 2022b), organizations proceed with disposing of their (often long-term) assets for multiple reasons, such as the asset's value has fully depreciated, the asset is no longer useful, unforeseen circumstances, asset replacement, and repair/service costs exceed the asset's value. Independently of these reasons, organizations need to accurately document any disposal-related transactions in their financial books. Asset disposal is either normal exemplified with transfer of ownership or trading exemplified with exchanging old goods with new ones (FreshBooks, 2022).

During thing assetization, disposability as per Section 3.1 would require the following details about a thing: salvage-cost and disposability-reason. By analogy with depreciation and transferability, Table 7 illustrates potential correspondences between the asset class-model's properties and WoT-TD information-model's constructs. The same vocabularies and notation are adopted as earlier.

- **Convertibility:** it is the exchange of a convertible type of asset into another type of asset usually at a previously agreed-upon price on or before a previously agreed-upon date. According to Chen (Chen, 2022), examples of assets that can undergo conversions are convertible bonds and preferred shares. Conversion is useful as it permits to categorize assets into current (those with a shorter life span and easily transferable into cash) and fixed (intended for long-term use and unlikely to convert quickly into cash).

During thing assetization, convertibility as per Section 3.1 would require the following details about a thing: upgrade-date and upgrade-type. By analogy with depreciation, transferability, and disposability, Table 8 illustrates potential cor-

respondences between the asset class-model's properties and WoT-TD information-model's constructs. The same vocabularies and notation are adopted as earlier.

## 4 SYSTEM IMPLEMENTATION

A system was developed to demonstrate the IoT-assetization framework's technical doability. It is described below in terms of architecture, IoT-assetization API, and usage scenarios. Documentation is available at [assetizationapp.fly.dev/docs](https://assetizationapp.fly.dev/docs).

### 4.1 Architecture

Fig. 8 illustrates the main components of the system's architecture. Key quality attributes like modularity as well as incorporation of best-in-class tools were adopted during the development to achieve efficiency, versatility, and seamless capability integration. The different components are as follows.

**Data Acquisition:** serves as an entry point to collect data about things. This is done by harnessing Node-RED's ([nodered.org](https://nodered.org)) flow-based mechanisms. Node-RED is an open-source tool for IoT applications, featuring a user-friendly graphical interface for creating workflows through drag-and-drop. It is widely used for connecting devices, APIs, and services, and is good at automation, data processing, and prototyping. Some key strengths include real-time testing and debugging that allow for incremental deployment of changes along with their real-time evaluation. This enables continuous system monitoring and refinement of flows, aiding in achieving accuracy and maintaining a consistent system state.

**IoT-assetization API:** ensures seamless data transmission between the data acquisition component and data processing and management components, as well as provides services to the dashboard module. IoT-assetization API is implemented with FastAPI ([fastapi.tiangolo.com](https://fastapi.tiangolo.com)) framework in compliance with REST principles. FastAPI is a Web framework for building APIs in Python 3.7+, known for its ease of use and speed. It generates interactive API documentation (using Swagger UI and ReDoc) and employs token-based authentication. It uses stateless communication and standard HTTP methods for clarity and consistency in client-server interactions. FastAPI framework also ensures a very high speed and efficiency in handling Web requests and response. Finally, token-based authentication offers secure access to the IoT-assetization API functionalities.

**Data Processing and Mmanagement:** collects,



Table 5: Property/Construct correspondences for the needs of depreciation.

| Asset static-model's<br>property           | WoT-TD information-model |   |
|--|--------------------------|---|
|  | Construct                | Corrective action   |
| <i>p/first-date-of-use:Asset</i>           | <i>p/created:Thing</i>   |   |
| <i>p/original-cost:Asset</i>               | Not available            | add <i>p/amount[schema.org]</i> from <i>InvestmentOrDeposit</i> class to <i>Thing</i> class |
| <i>p/level-of-use:Invocation</i>           | Not available            | add <i>c/SystemLifetime[ssn]</i> to <i>Thing</i> class                                      |
| <i>p/number-of-years-of-use:Invocation</i> | Not available            | add <i>p/validThrough[schema.org]</i> from <i>Offer</i> class to <i>Thing</i> class         |
| <i>p/condition-of-use:Invocation</i>       | Not available            | add <i>c/Condition[ssn]</i> to <i>InteractionAffordance</i> class                           |

Table 6: Property/Construct correspondences for the needs of transferability.

| Asset static-model's<br>property      | WoT-TD information-model |  |
|---------------------------------------|--------------------------|--|
|                                       | Construct                | Corrective action  |
| <i>p/previous-location:Invocation</i> | Not available            | add <i>p/fromLocation[schema.org]</i> from <i>TransferAction</i> class to <i>Thing</i> class |
| <i>p/current-location:Invocation</i>  | Not available            | add <i>p/toLocation[schema.org]</i> from <i>TransferAction</i> class to <i>Thing</i> class   |
| <i>p/transfer-date:Invocation</i>     | Not available            | add <i>p/endTime[schema.org]</i> from <i>TransferAction</i> class to <i>Thing</i> class      |

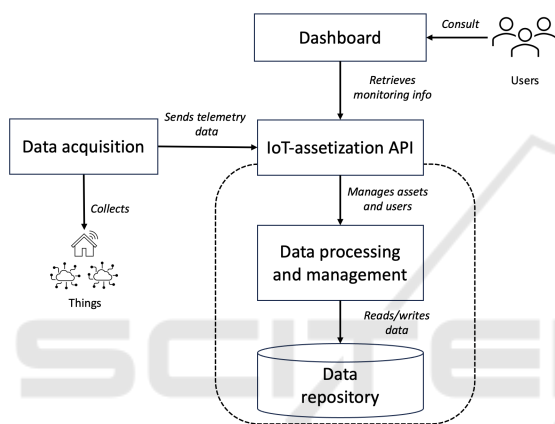


Figure 8: Components and interactions in the system.

transforms, and manages telemetry data and manages users' accounts and asset-related data, such as types, usage information, depreciation over time, and other details.

**Data Repository.** is a PostgreSQL (postgresql.org) database used by the data processing and management component to store all thing- and asset-related data.

**Dashboard.** is a front-end application implemented using Svelte (svelte.dev) framework to provide a visual representation and monitoring interface of assets.

### 4.2 IoT-Assetization API

We delve into the architectural design, functionalities, and potential applications of the IoT-assetization API, explaining its role in asset management. The API is divided into modules that expose a variety of endpoints, each serving a specific purpose within the asset's management lifecycle, accepting JSON payloads for requests and responding with JSON-encoded data. Key modules include:

- **Asset Management:** groups endpoints to create, list, update, and delete assets. Actions over assets are also registered, providing a history of interactions.
- **Task Management:** provides capabilities to track to-do items, with endpoints to create, retrieve, modify, and delete.
- **User Administration:** handles user accounts, providing endpoints for creating, updating, and deleting users. Users must first obtain an access token through a login endpoint, which is then used for subsequent requests. This module also provides a token refresh mechanism, ensuring uninterrupted access for authenticated users.
- **Type Management:** allows the definition and manipulation of asset and action types, ensuring flexibility in asset categorization. Any asset has a data field "type" in which values like sensor, lamp, smart lock, and others can be informed.

### 4.3 Usage Scenarios

Some screenshots from the system's dashboard are presented in Fig. 9 and 10 offering a visual representation of the system's capabilities in managing assets.

Fig. 9 is a consolidated view of all things that are subject to assetization, enabling users to quickly view basic details like type, description, lifespan, and date of creation. Such a layout not only aids in efficient asset management, but, also, ensures transparency regarding asset health, usage, and depreciation.

Fig. 10 is a detailed breakdown of a particular asset, in this case, "ultramegawide sensitive sensor". Here, the system provides a clear depiction of asset management principles applied to things, including the following key elements:

- **Asset Depreciation Graph:** displays the depreciation cost of an asset over time, providing insights

Table 7: Property/Construct correspondences for the needs of disposability.

| Asset static-model's<br>property          | WoT-TD information-model |  |
|---|--------------------------|--|
|   | Construct                | Corrective action  |
| <code>p/salvage-cost:Invocation</code>    | Not available            | add <code>p/minPrice[schema.org]</code> from <code>PriceSpecification</code> class to <code>Thing</code> class                                     |
| <code>p/disposability-reason:Asset</code> | Not available            | add either <code>p/comment[schema.org]</code> from <code>CreativeWork</code> class or <code>c/Observation[sosa]</code> to <code>Thing</code> class |

Table 8: Property/Construct correspondences for the needs of convertibility.

| Asset static-model's<br>property  | WoT-TD information-model      |   |
|-----------------------------------|-------------------------------|---|
|                                   | Construct                     | Corrective action   |
| <code>p/upgrade-date:Asset</code> | <code>p/modified:Thing</code> |   |
| <code>p/upgrade-type:Asset</code> | not available                 | add <code>p/comment[schema.org]</code> from <code>CreativeWork</code> class to <code>VersionInfo</code> class |

into the asset's financial value and expected lifespan.

- **Asset Usage Information:** details how long an asset has been in use, its active usage time, and the condition of use.
- **General Asset Information:** shows different asset types (e.g., washing machine, sensor, smart light), allowing users to quickly comprehend the distribution of asset by type.
- **Asset Health Indicator:** represents assets that are above or below the salvage price, offering a quick overview of their overall health.

In the detailed asset analytics view, users can check nuanced insights, ranging from depreciation-cost trend to usage information and overall health. Such granularity in data presentation facilitates comprehensive asset tracking, ensuring timely maintenance and informed decision-making. Our system showcases the transformative potential of blending IoT capabilities with asset management principles.

## 5 FUTURE WORK

Our future work is to design an asset cost-model that would help estimate the costs of assetizing things and the potential economical, operational, informational, or regulatory benefits of this assetization. We recall that things are subject to depreciation, transferability, disposability, and/or convertibility that would impact the cost of operationalizing them. For instance, depreciation could constitute a source of expense due to maintenance cost while transferability could constitute a source of income due to change of ownership.

According to FasterCapital, a cost model has a scope, drivers, equations, and validation (FasterCapital, 2023). Costs could be of different types such as marginal, average, full, and sunk contributing differently to the final cost model. Our initial thoughts would be a 2-stage asset cost-model where the first stage would cover costs to encapsulate things into as-

sets and the second stage would cover costs to encapsulate assets into services. When working out the asset cost-model, we deem necessary to consider the specificities of things like physical *versus* digital, static *versus* mobile, and online *versus* offline. A physical thing like sensor could require on top of an acquisition cost, a maintenance cost over a period of time. Contrarily, a digital thing like software could require a lease cost (in compliance with SaaS), only, leaving the maintenance cost to the software's owner.

- **From:thing To:asset** cost-model would cover separate costs related to things, assets, and encapsulation, respectively. These costs would vary depending on the nature of thing with focus on carrying out encapsulation operations between things and assets as reported in Table 1. Encapsulation operations are also complemented with additional operations reported in Table 4 where concrete details are cascaded from assets to things in order to have these things activated.
- **From:asset To:service** cost-model would cover separate costs related to assets, services, and encapsulation, respectively. These costs would vary depending on the nature of asset with focus on carrying out encapsulation operations between assets and services as reported in Table 2. Encapsulation operations are also complemented with additional operations reported in Table 3 where concrete details are cascaded from services to assets in order to have these assets activated.

## 6 CONCLUSION

This paper discussed the stages and techniques enabling to transition from Internet-of-Things to Internet-of-Assets by elevating things to assets. Organizations own assets to produce positive economic values. Contrarily to existing initiatives that expose things as services, we "squeeze" assets between things and services allowing to track things' depreciation over time, transferability across locations, dis-

| Name                       | Description                 | Type       | Created at   | Lifespan |
|----------------------------|-----------------------------|------------|--------------|----------|
| Ultrawide sensitive sensor | The one that detects it all | sensor     | Oct 12, 2023 | 12 years |
| Philips smart light        | Smart light living room #2  | lamp       | Oct 12, 2023 | 2 years  |
| Philips smart light        | Smart light living room     | lamp       | Oct 12, 2023 | 2 years  |
| Philips smart light        | Smart light kitchen         | lamp       | Oct 12, 2023 | 2 years  |
| Philips smart light        | Smart light bathroom        | lamp       | Oct 12, 2023 | 2 years  |
| Philips smart light        | Smart light room 3          | lamp       | Oct 12, 2023 | 2 years  |
| Philips smart light        | Smart light room 2          | lamp       | Oct 12, 2023 | 2 years  |
| Philips smart light        | Smart light #1              | lamp       | Oct 12, 2023 | 2 years  |
| Specialized audio sensor   | Ultrasensitive audiosensor  | sensor     | Oct 12, 2023 | 5 years  |
| Huawei Smartlock           | Smart lock upstairs         | smart lock | Oct 12, 2023 | 2 years  |
| Huawei Smartlock           | Smart lock backdoor         | smart lock | Oct 12, 2023 | 2 years  |
| Huawei Smartlock           | Smart lock frontdoor        | smart lock | Oct 12, 2023 | 2 years  |

Figure 9: List of available things after assetization.

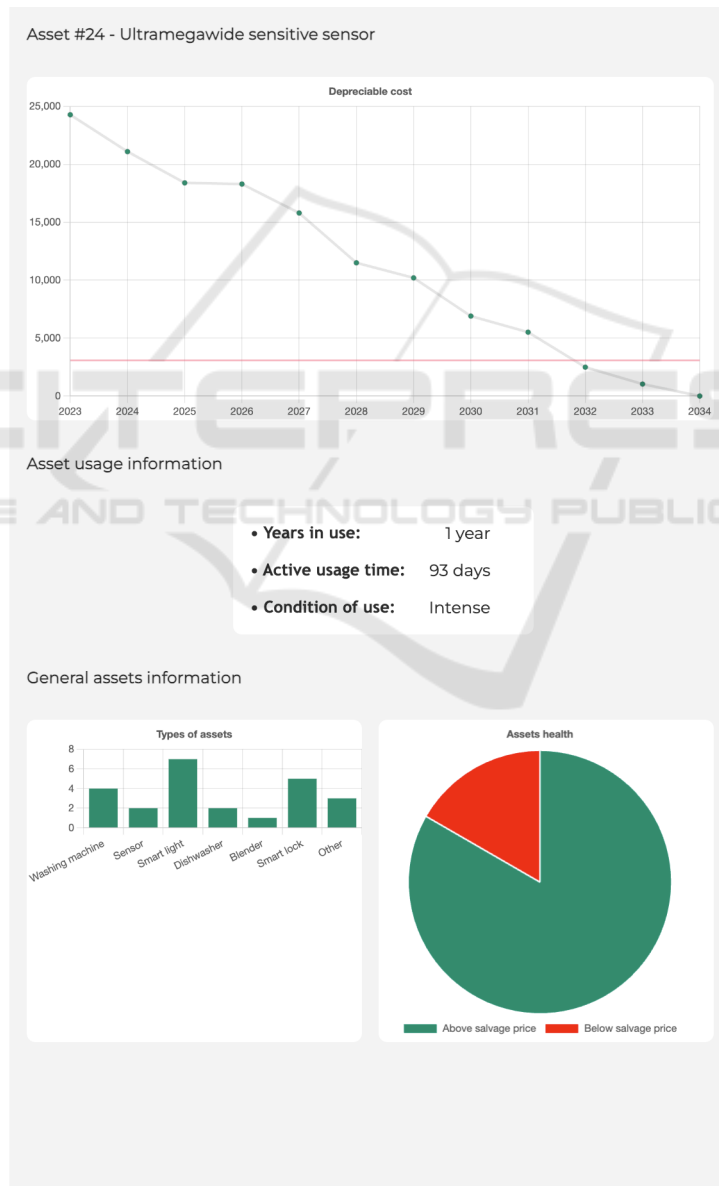


Figure 10: Summary of asset analytics.

possibility after time/use, and convertibility across platforms. These 4 management aspects provide better insights into how to handle things. For a successful thing-asset transition, we developed an IoT assetization framework that encompasses 3 models, thing, asset, and service, and supports 2 stages driving thing assetization and asset servitization, respectively. The implemented system demonstrates the practicality of thing assetization and shows how to effectively handle things as assets. Finally, future research involves the development of an asset cost-model, focusing on the economic implications of IoT assetization through some economical, informational, operational, and regulatory benefits to things' owners.

## REFERENCES

- Becker, S., Hüick, B., Naujokat, K., Schmeiser, A., and Kasten, A. (2013). ODRL 2.0 Revisited. In *Proceedings of INFORMATIK'2013*, Koblenz, Germany.
- Chen, J. (April 2022 (Visited in September 2022)). Conversion in Finance. [www.investopedia.com/terms/c/conversion.asp](http://www.investopedia.com/terms/c/conversion.asp).
- Custers, B. (2022). New Digital Rights: Imagining Additional Fundamental Rights for the Digital Era. *Computer Law & Security Review*, 44.
- Department of the Treasury, I. R. S. (2021 (visited in August 2022, [www.irs.gov/pub/irs-pdf/p946.pdf](http://www.irs.gov/pub/irs-pdf/p946.pdf))). Publication 946 How to Depreciate Property.
- Duan, Y., Sun, X., Longo, A., Lin, Z., and Wan, S. (January 2016). Sorting Terms of "aas" of Everything as a Service. *International Journal of Networked and Distributed Computing*, 4(1).
- FasterCapital (2023). Create Effective Cost Models for your Business. <https://fastercapital.com/>. (Visited in August 2023).
- FreshBooks (2021 (Visited in September 2022)). What Is Disposal of Assets? Definition & Explanation. [www.freshbooks.com/en-gb/hub/accounting/disposal-of-assets](http://www.freshbooks.com/en-gb/hub/accounting/disposal-of-assets).
- Gunnarsson, M. and Gehrman, C. (2022). Secure Ownership Transfer for the Internet of Things. In *Proceedings of ICISSP'2020*, Valletta, Malta.
- Heather, M., Heid, N., and Lien, N. (2019). Systematic mapping of checklists for assessing transferability. *Systematic Reviews*, 8.
- Keundug, P. and Heung-Youl, Y. (2021). Proposal for Customer Identification Service Model Based on Distributed Ledger Technology to Transfer Virtual Assets. *Big Data and Cognitive Computing*, 5(3).
- Kuzminykh, I. and Carlsson, A. (2018). Analysis of Assets for Threat Risk Model in Avatar-oriented IoT Architecture. In *Proceedings of NEW2AN'2018*, Saint Petersburg, Russia.
- Maamar, Z., Faci, N., Ugljanin, E., Baker, T., and Burégio, V. (2021a). Towards a Cell-inspired Approach for a Sustainable Internet-of-Things. *Internet of Things*, 14.
- Maamar, Z., Faci, N., and Yahya, F. (2021b). A Guiding Framework for IoT Servitization. In et al., A., editor, *Next-Gen Digital Services. A Retrospective and Roadmap for Service Computing of the Future*.
- O'Sullivan, A. and Sheffrin, S. (2021). *Economics: Principles in Action*. Prentice Hall, Washington.
- Panicker, P. (February 23, 2022 (visited in August 2022, [www.quicksolv.com/blog/internet-of-things/asset-tracking/types-depreciation-calculations](http://www.quicksolv.com/blog/internet-of-things/asset-tracking/types-depreciation-calculations))). Types of Depreciation and their Calculations.
- Patil, V. and Shyamasundar, R. (2021). Landcoin: A Practical Protocol for Transfer-of-Asset. In *Proceedings of ICISS'2021*, Patna, India.
- Raina, A. and Palaniswami, M. (2021). The Ownership Challenge in the Internet of Things World. *Technology in Society*, 65.
- Team, C. (2022a). Types of Assets. [corporatefinanceinstitute.com/resources/knowledge/accounting/types-of-assets](http://corporatefinanceinstitute.com/resources/knowledge/accounting/types-of-assets). (Visited in July 2022).
- Team, I. E. (2021 (Visited in September 2022)b). What is Asset Disposal? Definition, Benefits and Examples. [www.indeed.com/career-advice/career-development/asset-disposal](http://www.indeed.com/career-advice/career-development/asset-disposal).
- Vandermerwe, S. and Rada, J. (1988). Servitization of Business: Adding Value by Adding Services. *European Management Journal*, 6(4).
- W3C (2018). ODRL Information Model 2.2. <https://www.w3.org/TR/2018/REC-odrl-model-20180215/>. (Visited January 2023).
- W3C (2020). Web of Things (WoT) Thing Description. [www.w3.org/TR/wot-thing-description](http://www.w3.org/TR/wot-thing-description). (Visited in January 2021).
- Weiser, M. (1999). The Computer for the 21<sup>st</sup> Century. *Newsletter ACM SIGMOBILE Mobile Computing and Communications Review*, 3(3).