G.J. Frijters

Design of a Supply Chain Coordination System-of-Systems

Applied to offshore wind power park maintenance

TU Delft

Design of a Supply Chain Coordination System-of-Systems

Applied to offshore wind power park maintenance

By

G.J. Frijters

Master Thesis

in partial fulfilment of the requirements for the degree of

Master of Science

in Mechanical Engineering

at the Department Maritime and Transport Technology of Faculty Mechanical, Maritime and Materials Engineering of Delft University of Technology

to be defended publicly on Monday December 20, 2021 at 01:30 PM

Student number:	4029488	
MSc track:	Multi-Machine Engineering	
Report number:	2021.MME.8576	
Supervisor:	Dr. ir. W.W.A. Beelaerts van Blokland	Faculty 3ME
Thesis committee:	Dr. ir. D. Schott,	Faculty 3ME
	Dr J.M. Vleugel	Faculty 3ME
	MSc. A. Beije	Director BlockLab
Date:	December 6, 2021	

TU Delft

Preface

Since the emerge of Bitcoin I got intrigued by the fundamental concept of achieving trust among distributed systems, without the need of a central trusted party. Not only the technological ingenuity and its brilliant way of incentivizing perpetuation of the network, but also the fair distribution of power appealed to me. In an increasingly digitalised, automated world where machines take centre stage, I could see the potential of blockchain technology.

Although driven by many legitimate reasons, the hesitancy for enterprises to adopt this great new technology always surprised me. How could I transfer the potential that I saw to others? As in most cases, hesitancy is partially fuelled by a lack of understanding. A great way to increase understanding, and to fulfil my own eagerness for creativity, is to showcase the capabilities of blockchain technology through an effective, well-researched application of it. I only had to find the right use case.

Due to the energy transition towards renewable energy and suitability of the Netherlands for offshore wind energy, research into lowering the cost of offshore wind energy became a relevant topic. Influential to the cost of energy is the operational performance of an offshore wind power park. In a series of highly automated processes for the mobilization of the maintenance supply chain, the coordination and contracting of suppliers for scheduled maintenance operations is performed manually. Crucial to this labor intensive process is inter-company communication and processing of commercially sensitive maintenance schedules, asset or resource availability schedules, price rates and business proposals. This was the application I was looking for. Leveraging blockchain technology for trustworthy automation of communication and sensitive information processing between distributed supply chain systems, with the goal to increase operational efficiency.

It took me a while to find the right approach for this research project, but with the gained knowledge and end result I can look back with great satisfaction. I have learned a lot about offshore wind industry, blockchain technology, scientific design research, and myself. I would like to thank both Dr. Ir. W.W.A. Beelaerts van Blokland and MSc A. Beije for their relentless support, wisdom and guidance. Additionally, I would like to thank MSc A. Beije for the opportunity and the resources made available for me to perform this research. I would like to thank blockchain engineer Hamza Suwae for his time and deep technical programming knowledge that helped me achieve feasibility of the developed designs for this research. Finally, I thank my family, and especially my girlfriend Antoinette, for their love and caring that kept me going.

Wout (G.J.) Frijters
Rotterdam, December 2021

Summary

Due to the European Union renewable energy targets for 2030 and beyond, increasing trends of installed offshore wind energy production capacity and increasing wind turbine size emerge. Therefore, the offshore wind maintenance demand increases and becomes increasingly more complex. The most applied maintenance strategy is reliability-centred maintenance, which is a form of preventive maintenance that uses prediction models to determine future ocean and weather states and asset health states. Based on those predictions, maintenance demand is scheduled. The processes that lead up to the generation of the maintenance schedule ranging from asset condition monitoring, to data analysis, to future state predictions, to maintenance scheduling are currently highly automated. The last part of the maintenance organisation cycle, matching and contracting of maintenance supply for the demand is currently still done manually via email and phone. Offshore wind maintenance operations predominantly are multi-party operations, requiring vessels, teams of technicians, spare-parts and ports supplied by the WPP itself, shared WPP inventory, third-party service providers and OEMs. The increasing maintenance demand puts a lot of pressure on the already complex task of the Asset Manager that is processing all this information and communication manually.

The problem that is preventing automation of the matching and contracting process, is the lack of a system of demand and supplier systems, that processes commercially sensitive information, such as maintenance demand schedules and supplier availability schedules, in a trustworthy privacy preserving manner.

The main research question for this design research therefore becomes:

How to design a technical feasible decentralized system-of-systems that enables automated matching and contracting of maintenance supply for scheduled demand through privacy preserving processing of commercially sensitive data?

Automating the matching and contracting of maintenance supply with demand is believed to have multiple benefits. First, elimination of the human information processor leads to more information being processed for better and faster results, and less errors. Second, it prepares the offshore wind industry for data-driven concepts such as Industry 4.0, and autonomous maintenance organisation. Third, process lead times and process labour times are heavily reduced, leading to faster maintenance mobilization for failing assets and reductions in labour costs, that both result in increased revenue for the WPP.

The theoretical framework for this design research is therefore; systems-of-systems theory, blockchain technology and systems engineering methodology. Via an elaborate system-of-systems design process, a design for a decentralised system-of-systems shall emerge trusted with, and capable of automating between networked supplier and demand systems while processing sensitive data through privacy preserving, cryptographic methods.

The design approach followed is based on the agile blockchain application engineering method “ABCDE, complemented with Baseline Protocol design features and design principles of system-of-systems theory. Working closely together with a blockchain developer from BlockLab, regular consults with lead developers and system architects of the Baseline Protocol, and programming to a minimum working s-o-s instance, a technical feasible design is achieved.

The developed s-o-s design consists of three systems; Demand System, Supplier System and Blockchain System cooperating automatically via the designed Workflow. For the s-o-s, the Workflow and the three individual systems detailed design were developed.

As expansion of the system-of-systems theoretical framework, an additional definition layer was introduced by the researcher to describe the smallest, passive pieces of data that are consumed and exchanged in the decentralised information processing system-of-systems, defined as “Data Objects”. Easily distinguishable from other system elements, Data Objects usually are used as unique identifiers such as serial numbers, blockchain addresses or hashes, but could also represent templates for data structures such as JSON schemas. Think of them as the nuts, bolts, pulleys or gears in a mechanical system, the smallest passive parts universally used to enable cooperation between other parts.

Validity of the S-o-s design was proven by means of a case study for a large WPP, with an assumed supply chain of on average four potential suppliers for each of the eight maintenance operation requirements. The designed automated matching and contracting Worklow was compared to the current manual matching and contracting process. Based on activity lead time data of two undisclosed process analyses from BlockLab, the automated Workflow unlocked a matching and contracting process lead time reduction of 56% and a labour process time reduction of 98%.

The novelty of this research in the unprecedented design of a decentralised system-of-systems for automated matching and contracting of maintenance supply for scheduled maintenance demand. In addition to the extension of system-of-systems theory, by the additional definition layer of Data Objects, that is the academic contribution of this design research.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

This introductory chapter introduces the research area and motivation in the first paragraph. The second paragraph zooms in on the context of this research. In the third paragraph, the discovered problem shall be defined, that will be addressed in this research. In the fourth paragraph, a solution direction for the defined problem is presented and explained. In the fifth and final paragraph the main research question and associated sub-questions are presented.

1.1 Offshore wind power

The European energy system is undergoing a transition supporting the fulfilment of the objectives of the Paris Agreement and the European Green Deal. One of the key elements in these climate change combatting policies is the complete decarbonization of the energy sector by 2040. Offshore wind power is one of the attractive renewable energy sources for replacement of the current polluting sources. In 2020, roughly 22 GW of energy generating capacity was installed in European waters, which is projected to grow more than a tenfold for 2050, varying between 230 and 380 GW. On top of this, the average size of an offshore wind turbine is increasing as well. Larger turbines generate more energy and run more efficient[1].

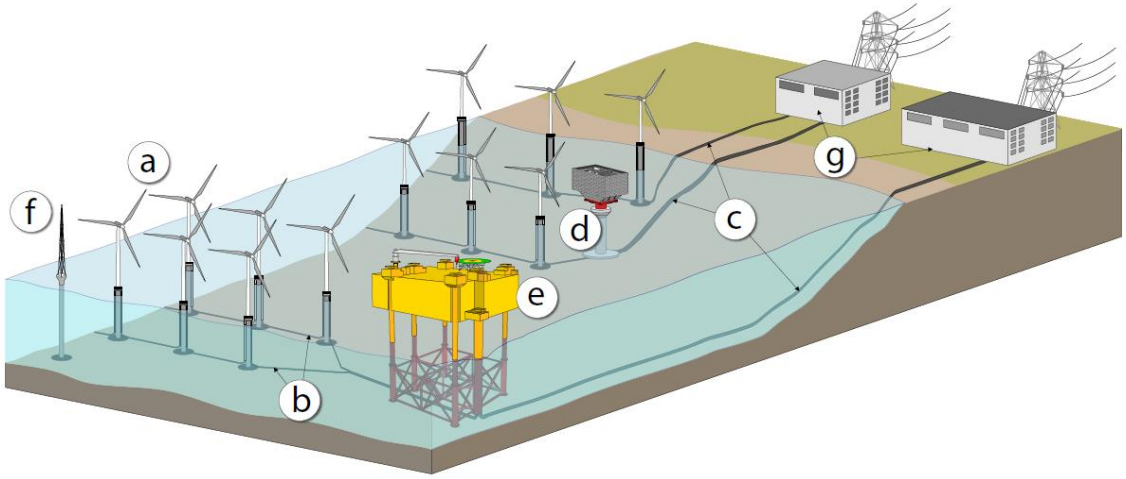


Figure 1: Graphical representation of Offshore Wind Power Park. a) Offshore wind turbine, b) Array cables, c) Export cables, d) Transformer station, e) Converter station, f) Meteorological mast

These developments bring new challenges. With the offshore capacity and turbine size growing rapidly Transmission System Operators (TSOs) and Wind Power Park (WPP) operators are faced with an increasing maintenance demand and complexity. Operating and maintaining the remotely located, difficult to access, assets is already a big challenge, which also requires much more effort to plan logistics needed to support these activities. Since current state-of-the-art wind turbines do not measure the condition of all critical components, unscheduled maintenance will still be needed. In order to ensure reliable operation and high level of platform and WPP availability, optimisation is needed based on the inputs from all stakeholders. Since communication is the key to success, development of the information flow between TSOs, WPP owners and other stakeholders can help achieve ambitious goals from European regulation while insuring lowest cost possible for end users.

Compared to onshore, operation and maintenance (O&M) processes are more complex since both transition systems and connected wind turbines are not as easily accessible as the assets on the land. Maritime operations are a major contributor to O&M costs of offshore wind power parks. WPP operators and TSOs therefore aim at keeping the number of visits to wind turbines and support systems to a minimum. The impact of O&M costs and costs per maritime operation (trip) is set to

increase with increasing distance of WPPs from the shore as only possible option for further offshore capacity integration into the existing system. A list of the planned WPPs until 2030[1] shows that the possible near shore locations in the Dutch part of the North Sea are exhausted and further offshore locations are currently under development. The Dutch wind energy industry has recognised the important role of O&M logistics in their wind energy R&D agenda[2], beacon 37.

Currently O&M logistics is done in a similar way as for the traditional oil and gas offshore sector, where a central party links the request for maritime operators with the suppliers of material, components and personnel. At present, coordination of the logistics is done by a human, which is already a large and complex task. With the vast increase in number of offshore wind installations, there's an immediate need for digitization of supply chain processes because it will become impossible to do this manually. Asset owners call for an increased highly automated process making use of the digitisation of the supply chain and data sharing triggered by data from condition monitoring. In order to automate this process, the demand, supply and execution sides in the logistic chain all need to be digitised from end to end, creating an end-to-end digital supply chain network.

One of the key performance indicators for WPP is wind turbine downtime, and is for multiple reasons an unwanted situation. Primarily because downtime prevents WPPs from making optimal use of the wind to produce energy and secondly because asset owners are contractually bound to deliver a certain level of energy. Although downtime cannot be eliminated, condition monitoring should make unplanned downtime due to component failure a thing of the past. Currently applied maintenance strategy is to perform planned maintenance that is scheduled based on predictive models, which can be improved with joint logistic planning by TSOs and WPPs. When all maintenance is foreseen in either of the scenarios, it is possible to align maintenance tasks on assets of different owners.

1.2 Offshore wind maintenance organisation

Offshore wind maintenance operations (MO) require four essential categories of assets and resources to enable execution, namely spare-parts, technicians, vessels and a port for on- and offloading. A typical WPP owns at least a few crew transfer vessels (CTV), a small dock, a warehouse containing an inventory of frequently needed WT spares and employs a team of wind turbine technicians. Such a configuration enables the WPP to execute most recurring, minor maintenance tasks efficiently, that typically account for 80% of the total maintenance demand [2]. The remaining 20% of the maintenance demand are major repairs and replacements, which require more resources and specialised assets. Owning these is economically inefficient due to low utilization and are therefore usually outsourced, or shared by multiple WPPs. Table 1 gives an overview of the typical MO requirements for different types of failures.

Type of failure	Maintenance operation requirements
Minor failure	A crew transfer vessel
	A team of WPP technicians
	A set of WPP inventory spare-parts
Major failure	A crew transfer vessel
	A large (SOV) vessel
	A team of WPP technicians
	A set of WPP inventory spare-parts
	An OEM spare-part
	A Port
Major replacement	Multiple crew transfer vessels
	A team of WPP technicians
	A team of specialised technicians
	A large (SOV) vessel
	A heavy lifting vessel (HLV)
	A set of WPP inventory spare-parts
	An OEM spare-part
	A Port

Table 1: Typical maintenance operation requirements for each failure type

[1] https://windopzee.nl/onderwerpen-0/wind-zee/waar/

[2] TKI Wind op Zee, *The Netherlands' Long-Term Offshore Wind R&D Agenda*, October 2019, Utrecht, The Netherlands

It's clear that these resource intensive operations require many different involved parties, that need their assets and resources available at the same time, and need to be mobilized fast in order to limit the energy production downtime of the wind turbine.

So, how is this supply chain mobilized after an occurring failure? Due to their remote and hard to access locations, current state offshore wind turbines are smart assets equipped with condition monitoring systems and communication systems, also known as supervisory control and data acquisition (SCADA) systems. These systems gather data from sensors all over the wind turbine's vital components and send the data to onshore analysis systems. The data is automatically being analysed and together with ocean and weather data applied to a parametric model of the wind turbine. With this model a prediction is determined on future health states of the wind turbine. Of course, a sudden unforeseen failure is also detected and immediately influences the current health state of a wind turbine. All the current and future predicted health states of each wind turbine in the WPP, together with the ocean and weather predictions form the input for an automatically generated maintenance schedule, according to the research of Stock-Williams [3]. This schedule describes a prioritized list of which turbines need what maintenance within a predicted optimal time window of low wind speeds for safe operation and production loss restriction. An arbitrary example of such a schedule is pictured below.

Maintenance job ID	Priority level	Wind turbine ID	Description	Required spares	Required vessel	Required technicians	Estimated lead time	Time window
MJ035	High	WT12	Gearbox replacement	Gearbox type ABC, gearbox oil, gaskets	Heavy lifting vessel	9 WT technicians, 9 GB technicians	3 days	Completion in [X, X+14] days
MJ083	Low	WT3, WT4, WT5	Manual inspection type ABC	None	Crew transfer vessel	2 WT technicians	1 days	Completion in [Y, Y+5] days

Figure 2: Arbitrary example of an automatically generated daily maintenance schedule

Based on the automatically generated maintenance schedule by the Asset Management System (AMS), the asset manager starts planning the prioritized maintenance operations [3]. First, suppliers of the right assets and resources need to be sourced and matched to each requirement of the maintenance operation. Those suppliers could be company departments of the WPP, a shared pool of resources between WPPs, or an external party. The asset manager finds this information in the WPP ERP system. Next, the asset manager calls or emails each supplier to request their availability and quotation within the scheduled time window, and for the estimated lead time. From the original equipment manufacturer (OEM) a spare-part is needed to be available prior to the scheduled time window. From both the vessel suppliers and the technician suppliers we need to know when their vessels and technicians are available within the optimal time window for the given lead time. And from the port we need to know when a berth is available for loading and offloading at the beginning and at the end of the MO within the scheduled time window. Once the asset manager has received enough replies from suppliers, he continues with the complex task of finalizing the MO planning.

This task includes:

- Finding overlap in the received supplier availabilities
- Within overlap, determine the earliest possible date of MO execution to limit downtime
- According earliest date, calculate all possible supplier subsets
- Selecting most suitable supplier subset according the WPP preferences
- Contracting the selected suppliers

All these activities are currently done manually by the asset manager and his team. Once the maintenance planning is finalized, preparation for the MO can begin. Unavailable spare-parts are manufactured, whereafter transported to the port of assembly. From the port, the vessels with technicians and spare-parts sail together to the WPP for execution of the maintenance operation. The current state maintenance organisation process as described in this section is illustrated on a high level in figure 3, with an indication of the level of process automation.

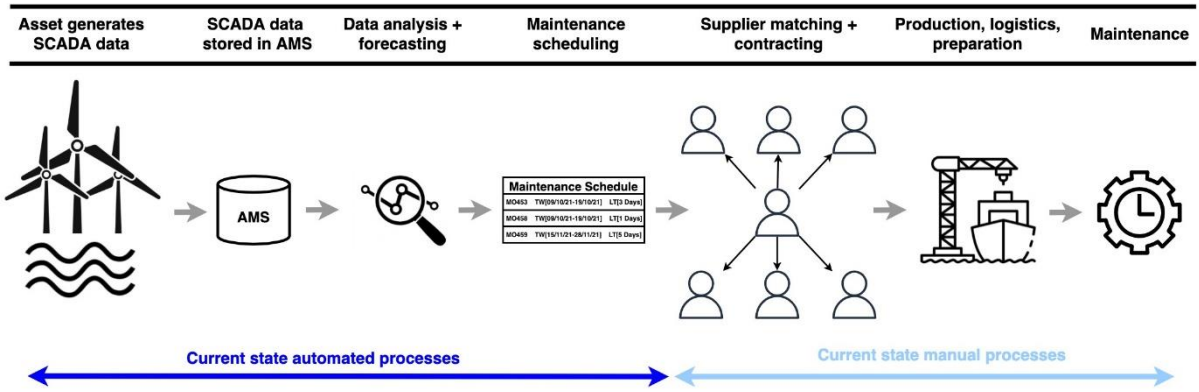


Figure 3: Current state WPP maintenance organisation process, with indication of process automation

The automated processes in the current state maintenance organisation are internal processes of the WPP. The currently manual supplier matching and contracting process is the first process where the information of external parties is involved. For every single scheduled MO, it's necessary for all these parties to exchange commercially sensitive information in order the finalize the MO planning. The sensitive information here is the scheduled maintenance demand of the WPP and the asset or resource availability calendar of each supplying party, which both give an indication about company performance. The maintenance demand schedule, the asset and resource availability schedules, and the related cost information are the most essential pieces of information to finalize the maintenance planning. All these pieces of information reside in the company ERP systems of the maintenance supply chain members, including the maintenance demand parties such as the WPPs and TSOs.

The workload and complexity of the supplier matching and contracting process becomes even more clear when zooming in, which is visualised in figure 4. The figure gives an overview of all potential suppliers a WPP asset manager needs to manually interact with in order to fulfil the requirements of a particular MO. The suppliers above are usually either internal WPP company departments or shared resource pools of multiple WPPs. The suppliers below are usually externally contracted.

The workload of a matching and contracting process lies in the amount of information transactions per MO, and the resulting amount of information to be processed for every MO. The complexity comes first from matching the right suppliers to specific MO requirements, and second from processing of

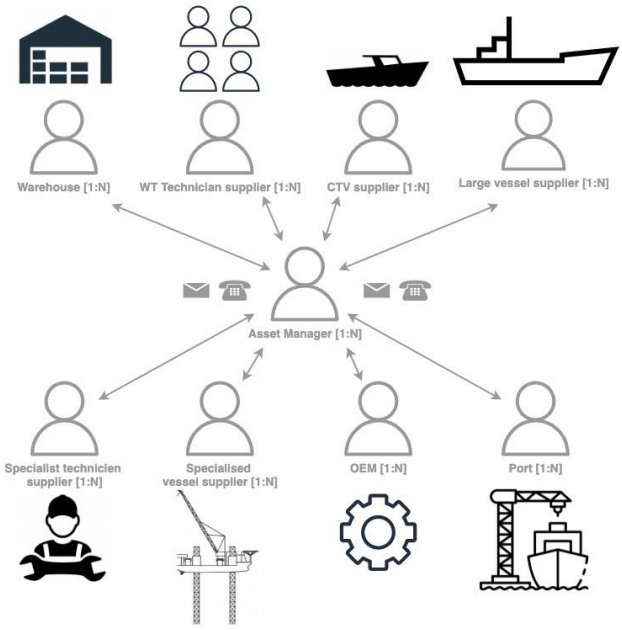


Figure 4: Overview of manual interactions of an WPP asset manager with WPP maintenance suppliers

all received availability and cost data to determine the final MO planning, with respect to the scheduled time window and estimated MO lead time, and the WPP optimisation preferences.

In the context of WPP operational performance it is important that maintenance is performed according schedule, because the schedule is optimised to limit wind turbine downtime, thus maximizing WPP performance. To support all sorts of MOs, including for sudden failures, it is therefore important that the lead time of the supplier matching and contracting process is as short as possible.

1.3 Problem definition

To achieve the renewable energy targets in the European Green Deal, plans are made to increase the installed offshore wind capacity in European water by a tenfold within the coming thirty years.
On top of that, the average size of offshore wind turbines increases because of cost-effective energy production. This results in more complex, multi-party maintenance operations with the need for more specialised, typically outsourced, equipment.
And, because near shore locations are exhausted, WPPs are located further away from shore. This results in longer transit times leaving less in the given time window for actual maintenance, but also in more variations in the maintenance supply chain setup.

Summed up the European offshore wind maintenance supply chain is subject to increasing maintenance operation complexity, for which the demand is about to grow by a tenfold in the next thirty years. The automated processes in the maintenance organisation can easily accommodate this upscaling, but are the manual processes ready for it?

Because of both the high capital and operational expenses of offshore wind maintenance assets and resources, available maintenance assets and resources will become scarce while in increasing demand. This puts tremendous pressure on the already complex job of the asset manager and his team. To be able to accommodate the increasing demand and complexity while complying to the maintenance schedule, they are forced to increase the supply chain size. This leads to more manual information transactions and more information being manually processed for each scheduled MO, within a similar amount of time.

One way of solving this is simply to add more human information processors, but that would negatively influence the WPP operational performance due to the increased costs. Additionally, limitations of the human as information processor are expected to lead to sub optimal solutions for the matching and contracting of suppliers for every MO.

To accommodate the tenfold increase of maintenance demand and complexity, while complying to the maintenance schedule, without negatively impacting the WPP operational performance, the solution has to be found in automated computerised information processing. Since the maintenance demand schedules, availability schedules and cost information reside in company ERP systems of the supply chain parties, an automated matching and contracting system of connected ERP systems could potentially be a solution. However, the crux of the matter is that we're dealing with commercially sensitive maintenance demand schedules and asset/resource availability schedules of supply chain parties that don't necessarily trust each other with that data. Business managers are logically hesitant about integrating their protected ERP systems with a system that consists of varying, unfamiliar supply chain parties on the basis of commercially sensitive data. This is not only a problem specific to the offshore wind industry, but for every industry that has to plan multi-party maintenance operations for a scheduled demand.

The generic underlying problem for the given context is:

A lack of a secure system of connected systems that enables trustworthy data processing for automated matching and contracting of maintenance supply for scheduled demand.

1.4 Envisioned solution

Before presenting the envisioned solution, the conventional approach for the defined problem is discussed. Because the main problem owner is the dominant supply chain party that is in need to maintain their assets, they usually develop a centralised system themselves for which every supply chain party has to integrate with. This centralised system will be hosted on the premises of the demand party or in a cloud environment owned by companies such as Google or Amazon.

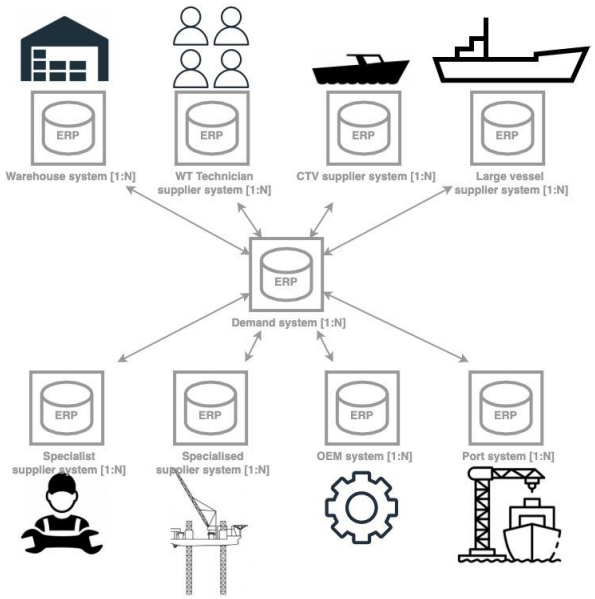


Figure 5: Undesirable, conventional, centralised approach to automate between supply chain systems

This setup is problematic for a few reasons. First of all, there is one dominant party in control over the automated supply chain system. This gives the dominant party the ability to control over access to the system, and control over the automated processes executed by the system and the connected supplier systems.
Second, the dominant party together with the cloud provider has in theory access to all the data processed by the system. The centralised system becomes a data lake full of commercially sensitive data, which is highly undesirable in an era where data is seen as the world's new most valuable resource.
Third, every demand party shall have to develop and host their own system meaning that every supplier has to integrate with every demand system they supply to. The amount of work and effort to manage all those separate integrations already defeats the efficiencies gained through automation, especially from the perspective of the supplier.

Considering the reasoning above, the envisioned solution has to be decentralised of nature, allowing for peer-to-peer exchange of sensitive data that is similar to the current state manual phoning and emailing. The envisioned solution should also include a secure trustworthy way of enabling process automation between distrusting party systems. Ideally it also allows for multiple demand and supplier systems, and easy integration of new supply chain parties. The envisioned solution is visualised in figure 6. One innovative

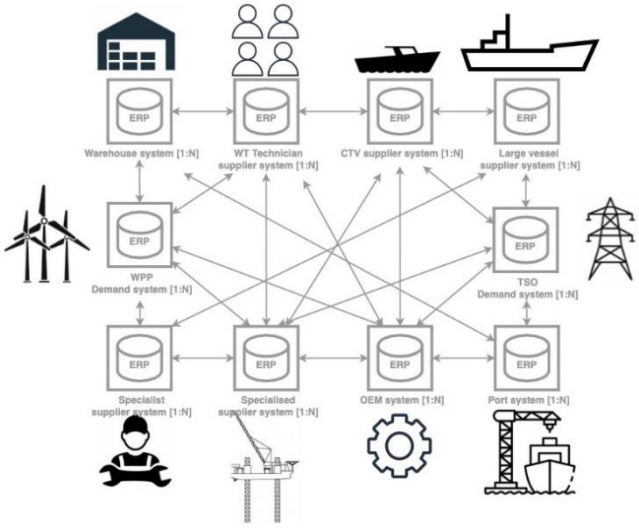


Figure 6: Envisioned, decentralised, approach to automate between supply chain systems, allowing for multiple demand and supplier systems.

technology famous for its decentralise nature and ability to create trust among systems is blockchain technology. Blockchain technology gained popularity through the economic application called Bitcoin, developed by pseudonymous Satoshi Nakamoto in 2008 [4].

The applicability of the technology to supply chains is well described in the book of Vyas, Beije and Krishnamachari (2019), where real world examples of supply chain issues justify the application of blockchain [5]. In comparison with described centralised systems, blockchain technology increases supply chain resilience because it removes the presence of a central authority, and thus a central point of failure. It provides a tamper-proof transaction ledger, and it provides trusted transactions based on algorithmically enforced rules without human intervention. Via encryption it allows for secure peer-to-peer data exchange while preserving the privacy of that data on a shared, public network.
Open source blockchain technology offers data security and cost-effective transmission of transactions in peer-to-peer networks with no central system. Removing the need for an information broker or numerous system integrations with every party in the supply chain. Therefore, it allows for a single, and direct business-to-business integration with all supply chain parties through one single system integration.
Blockchain allows for full transparency and traceability of transactions within a supply chain, which increases supply chain visibility and the ability to track provenance. Besides meeting with increased customer demands, this also aids in solving disputes between parties and increases the quality of delivered services because company underperformance becomes visible for the supply chain.
The use of smart contracts, which are computerized transaction protocols that execute terms of the contract, allow for real-time settlement of information and financial, and automation of these flows. Because blockchain provides a single validated consensus-based source of truth – the shared ledger – every connected party has access to an efficient and effective flow of data, which is proven to be essential for efficient supply chain coordination and responsiveness.

Taking all into consideration, blockchain technology is determined to be a useful tool for developing a solution to the defined problem. Through a single system integration taking part in a system of connected systems that allow for, via smart contracts automated, peer-to-peer exchange of encrypted sensitive data. Dominant parties, sensitive data risks, human processors and manual information transactions are eliminated; while increasing the supply chain's resilience, responsiveness, service quality and visibility. Blockchain also allows for automated financial settlement in digital currencies, that could lead to autonomous operation and coordination scenarios as envisioned in future data-driven concepts such as Industry 4.0 and Supply chain 4.0.

Not only in theory, but also in practise blockchain proves to be a useful tool for automating the supply chain information flow. The Naviporta platform includes a blockchain based digital notary, to notarise the ownership state and transfers of shipping documents. The platform reduces end-to-end documentation processing from 5-10 days to less than 24 hours[3]. The Tradelens platform leverages blockchain to track and share valuable shipping related events, that can lead to millions of dollars on operational savings for large and medium sized LSPs[4]. Twelve of the largest Coca-Cola bottlers use blockchain to transparently streamline cross-organizational transactions. For the entire supply chain, they are expecting $100 million in annual operational savings[5]. These cases will be further elaborated on in the next chapter.

[3] https://naviporta.com/
[4] https://www.tradelens.com/
[5] https://provide.services/news/baselining-the-north-america-coca-cola-bottling-supply-chain

1.5 Research objective and research questions

The determined objective for this research is to design a secure, decentralised system of connected systems to automate the supplier matching and contracting process for scheduled maintenance operations. The designed system-of-system shall take over all the necessary information transactions between supplier and demand systems, and associated information processing in order to accommodate the expected tenfold growth of offshore wind maintenance demand and complexity, while complying with the maintenance schedule and keeping operational expenses low.
Due to its merits, blockchain technology is selected as suitable technology to be included in the design of the envisioned solution. The designed system-of-systems shall push the automation level of maintenance organisation one step further. And precisely because of the use of blockchain technology, real-time financial settlement becomes possible, which opens the door to autonomous operation in a machine-to-machine economy. How the automation level is expected to shift with current and further research is illustrated in figure 7.

The main research question that is determined for this research is:

How to design a technical feasible decentralized system-of-systems that enables automated matching and contracting of maintenance supply for scheduled demand through privacy preserving processing of commercially sensitive data?

The answer to this question is found through answers of the following research sub-questions:

1. *What is the theoretical framework to define and describe envisioned system-of-systems?*
2. *What is the most suitable design method for envisioned system-of-systems?*
3. *How will the design be verified?*
4. *How will the design be validated?*
5. *What are the activities automated by envisioned system-of-systems in current state matching and contracting process?*
6. *What are relevant KPIs for the automated process?*
7. *What are the implications of the final design?*
8. *How is trustworthy processing of sensitive data enabled?*

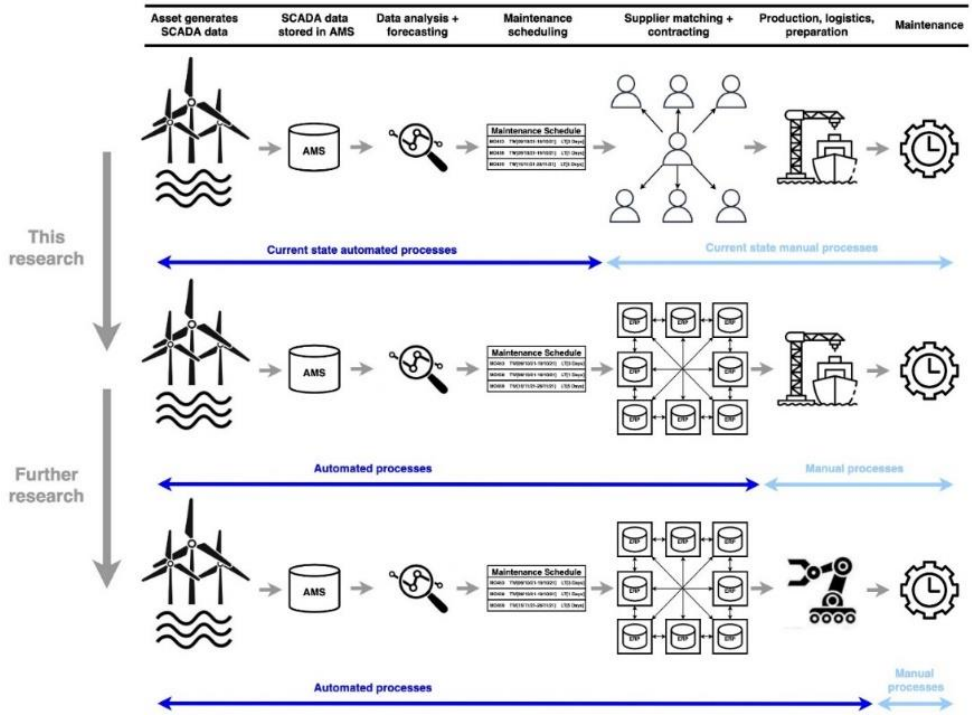


Figure 7: Current and further research objectives for maintenance organisation

Chapter 2: Research design

In chapter 2 the overall research design for this design research is explained. It starts with an elaborate study of literature relevant to the research context and defined research questions. Through the literature analysis, it is expected that the necessary theoretical foundation, design methods and design process requirements are found to help answering the main research questions. The chapter concludes with a determined research approach, including a design scope and design approach, followed by the answering of some research questions.

The three sections that form this research design chapter are:

- 2.1 Literature analysis
- 2.2 Research approach
- 2.3 Answers to research questions

2.1 Literature analysis

Because the objective of this research is to design a decentralised automated system-of-systems, literature is analysed on the topics of system and system-of-system theory, blockchain technology and system engineering methods. This section should give all the tools needed for the design process of the envisioned system-of-system.
The end of this section unveils a theory gap in the system-of-systems theory, that was discovered after analysing all mentioned literature.

The four sections in this literature analysis section are:

- 2.1.1 Systems and system-of-systems theory
- 2.1.2 Blockchain technology
- 2.1.3 System-of-system design methods
- 2.1.4 System-of-Systems theory gap

2.1.1 Systems and systems-of-systems theory

The first topic for which literature is analysed is systems and system-of-systems theory. This section should give us the terminology, properties and design principles for system-of-systems necessary to develop the envisioned system-of-system design.

The notion of holism, the concept that ideas, people or things must be considered in relation to the thing around them to be fully understood led to the development of System Theory. The first effort to capture the concept in terms and definitions was made by Ackoff, of the University of Pennsylvania, in 1971 [6]. The word “system” is a very general term derived from the Greek verb meaning “to compile”, of which many definitions reside in literature. Veeke, Ottjes and Lodewijks, developers of the “Delft Systems Approach” concluded the existing definitions lacked a crucial element, namely the perspective of the researcher. They provided the following definition of a system [7]:

*A **system** is, depending on the researcher’s goal, a collection of elements that is discernible within the total reality. These discernible elements have mutual relationships and eventually relationships with other elements from the total reality.*

A system is composed of **elements**, which are the smallest parts considered by the researcher in view of his goals. Elements can be both material and non-material. Materialized elements are defined as **concrete**, meaning they exist and are tangible. The opposite of concrete elements are **abstract** elements, which are separated from the material; intangible.

The interaction between elements are referred to as **relationships**. In an abstract system, these are conceptual interactions. In a concrete system, there is dynamic exchange. Elements influence each other either mutually or one-sided. Characteristics of one element can influence or initiate values of characteristics on another element, meaning that non-existent characteristics with value 0 could be influenced to have a certain value so that a non-existent characteristic becomes present.
The total reality in which the system, and all other systems, elements and relationships, exists is defined as the **universe**. As the system definition describes, a system is a group of elements distinguished by the researcher. The elements of that system have inter-relationships, but also relationships with other elements within that universe.
The elements in the universe that directly influence the values of characteristics of the system elements is defined as the **environment**. When a company is considered as a system, the society can be seen as a higher-level system that influences the elements of the company and therefore is a definite part of the system's environment.

In order to obtain a clearer insight into complex systems, it is extremely useful to differentiate the system into subsystems and aspectsystems. As a system is built from elements and relationships, it can be described through the lens of the elements or through the lens of relationships.
A **subsystem** is a partial collection of elements in the system whereby all the original relationships between these elements remains unchanged, and completely conforms to the definition of a system. A fuel system can be regarded as a subsystem of a car. So, a top-level car system can be subdivided into partial collections of subsystems, such as the fuel (sub)system, the engine, the bodywork subsystem. The subsystems are a partial collection of elements such as springs, bolts, shafts. A pyramid can be used to place the partial collections into a layered graphical definition framework, where each layer represents a collection of the layers below. For now, the pyramid lacks a later to be added top.

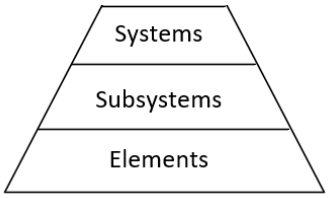


Figure 8: Layered definition framework for describing systems

An **aspectsystem** is a partial collection of the relationships whereby all the original elements remain unchanged. The relationships within an aspectsystem are usually of a single type. Examples of aspectsystems are:

- Thermodynamic aspectsystem; such as the conversion of chemical energy into kinetic energy, resulting in heat transfer and material expansion
- Tribology aspectsystem; the mutual friction of moving parts and the lubrication required
- Economical aspectsystem; the cash flows or the value-added flows within a company

A graphical clarification on sub- and aspectsystems is provided in figure.

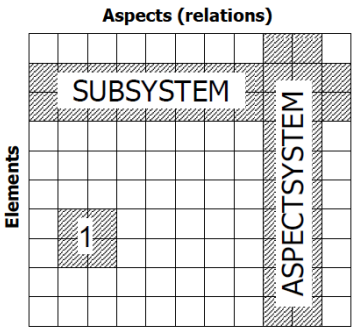


Figure 9: Graphical clarification of subsystems and aspectsystems [7]

A system also can have a **state**, which is the value of the properties at that time in the system. An **event** occurs when the value of the property of an element changes. When one event leads inevitably to other events, this is referred to as **activity**.
Sometimes not only the value of the properties but also the relationships within the system change, which is called a **changing structure**. The opposite is called an **unchanging structure**.
Another distinction can be made in terms of static or time-dependent systems. In a **static** system, we find elements and relationships but no events. In a **time-dependent** system, events and activities must take place to fulfil certain functionalities. In time-dependent systems processes can occur, that transform an input into an output through throughput. For these processes **permanent elements** and **temporary elements** can be distinguished.

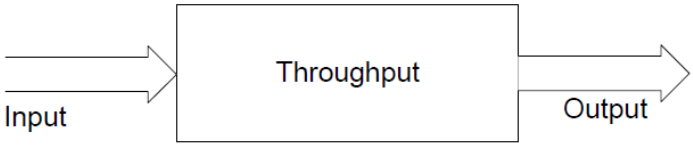


Figure 10: Simplest scheme of a time-dependent system

A **process** is a series of transformations that occur during throughput, which result in a change of the input elements in place, position, from, size function, property or any other characteristic. Through a process, a system fulfils its function in the environment. The fulfilment of that function in the environment is the system's **goal**.

System-of-Systems Theory
To have the ability to describe even more complex systems, a class of systems that are built from components which are large-scale systems in their own right, systems theory was expanded to eventually form the system-of-systems theory. Although commonly used, there was no widespread agreement on the exact meaning of the term System-of-Systems (S-o-s). Maier was the first in 1998 to examine the meaning of it in detail [8]. He proposes to define collaboratively integrated systems as "System-of-Systems" with two distinguishing characteristics for applying the term. If a system meets these characteristics it can be considered as an S-o-s.

A system-of-systems is an assemblage of components which individually may be regarded as systems, and which possesses two addition properties:

- *Operational independence of the components: if a S-o-s is disassembled into its components systems the component systems must be able to usefully operate independently. That is, the components fulfil customer-operator purposes on their own.*
- *Managerial independence of the components: the component systems not only* can *operate independently, they* do *operate independently. The component systems are separately acquired and integrated but maintain a continuing operational existence independent of the system-of-systems.*

Now that a definition of s-o-s is established, let's combine the s-o-s Theory of Maier with the Systems Theory in our layered definition framework for describing systems and s-o-s. Since s-o-s are a collection of individually identifiable systems, s-o-s form the top of the pyramid.

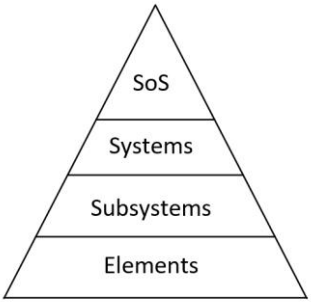


Figure 11: S-o-s added as top to the definition framework for describing systems and s-o-s

Now that a definition of s-o-s is established, let's look at some of the architectural principles that Maier identified to give the definition more body. He derived these principles, which originally were published as heuristics, from observed successfully developed s-o-s.

The first principle is the principle of **stable intermediate forms**, that originated from civil construction. It was recognized that it is desirable for a building to be self-supporting at many stages during its erection. This heuristic is applicable to s-o-s as well, as complex systems will develop and evolve within an overall architecture much more rapidly if there are stable intermediate forms than if there are not.
The second principle is **policy triage**, which gives guidance in selecting and supporting components for s-o-s. In essence it comes down to choosing very carefully what to try and control in a s-o-s design, since all systems should be able to operate and to be managed independently. Attempting to overcontrol will fail for lack of authority, and undercontrol will eliminate the system nature of the integrated system.
The third principle is **leverage at the interfaces**. Derived from the combination of two heuristics: "The greatest leverage in system architecting is at the interfaces. The greatest dangers are also at the interfaces". Again, the operational and managerial independence of the individual systems in an s-o-s leaves that there's nothing else to architect but the interfaces. The architecture of an s-o-s design are the interfaces. The internet *is* the interfaces, the Internet Protocol (IP). Applied to the example of an integrated air defence system, the s-o-s that combines all the independent systems is the command, control and communication network. It is basically the glue that combines the individual pieces.
The fourth and last principle is the principle of **ensuring cooperation**. "If a system requires voluntary collaboration, the mechanism and incentives for that collaboration must be designed in." The cost and benefits of system collaboration should be superior to the costs and benefits of independent operation, because in an s-o-s the independent systems choose actively, to some degree, if they want to participate or not. If no collaboration is incentivized and occurring, it is not a system-of-systems.

Maier provides multiple examples where his definition and principles are applied to. The internet is one of the examples that perfectly fits his theory.

Discriminating Factor	Applicability
Managerial independence of the elements	Component systems as acquired and operated by independent users. Component systems are developed (largely) by commercial firms following market dictates
Operational independence of the elements	Operational coordination is through voluntary adherence to technical standards. The standard setting process is also voluntary. The systems defense against noncooperators is only to exclude them. In the Internet's earlier stages of development it was more deliberately run by the U.S Government. Government sponsored projects continue to be important to the Internet's development
	User of Design Principles
Stable intermediate forms	The structure of the Internet is dynamic, with nodes being added and removed continuously and on their own volition. The main protocols are designed to allow evolution through replacement. The core protocol, IP, is now at version 4 with migration to version 6 beginning
Policy triage	The oversight bodies exercise very limited control, and carefully restrict their control to the network. Applications and underlying physical interconnects are controlled separately, if at all
Leverage at the interfaces	The architecture of the Internet is its interfaces. Nothing else is constant
Ensuring collaboration	The system fosters collaboration through low entry costs and benefits to cooperation. However, it is much weaker at excluding deliberate noncooperators, to the detriment of the system. This is a byproduct of its original development environment
	Classification
Collaborative	The system began with a directed purpose, but now follows purposes imposed upon it by its users. Operation and development is through the collaboration (largely voluntary) of its participants

Figure 12: The Internet and its system-of-systems properties[8]

2.1.2 Blockchain information technology

The second topic for which literature is analysed is blockchain technology. A fundamental understanding of the innovative technology has to be achieved in order to be able to use it for a system-of-systems design. It is also important that the merits and demerits are well understood. First, fundamentals are discussed, followed by classifications and interesting applications, therefore the structure of this section is as follows:

- 2.1.2.1 Blockchain fundamentals
- 2.1.2.2 Enterprise blockchain applications
- 2.1.2.3 Baseline Protocol

2.1.2.1 Blockchain Fundamentals

It started when the anonymous Satoshi Nakamoto released Bitcoin and its whitepaper in 2008 [4]. Cryptographic tools and the internet enabled the creation of a system to transfer value over the internet without the need for a trusted third party, i.e. financial institutions. According Nakamoto, the current way of electronic payments was inefficient because banks have to facilitate reversible transactions when disputed. The costs of mediation, and the cost of employing all the middlemen, limits the minimum transaction size and thus blocking the use of very small and quick payments. In the context of Industry 4.0s machine-to-machine interaction this is already a great loss. Additionally, the reversibility of transactions spreads the need for trust to all people handling the transactions, making the system information heavy and less secure. It also disables services that require irreversible transactions, such as in automated systems. As Nakamoto stated, "what is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers." The proposed solution to the double-

spending problem is using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more computing power than any cooperating group of attacker nodes.
That whitepaper became the foundation for a disruptive, unprecedented technology that is now defined as blockchain technology. Blockchain is often used interchangeably with distributed ledger technology, but it's only a type of distributed ledger. A distributed ledger is a database of replicated, shared and synchronized digital data that is geographically spread across multiple locations. It provides for an auditable history of information and is visible to anyone in the network. Distributed ledgers have, like blockchain, a mechanism of reaching consensus among the nodes. What makes blockchain unique is that is organizes data in blocks and updates the entries using an append-only structure. The following sections will further elaborate on the technology that Nakamoto laid the ground work for.

In order to help place the information below into context, a boiled down explanation of blockchain technology is provided. First, it is important to define the public ledger as a record-keeping system. It holds the list of all addresses in the network and their respective holdings together with the generated blocks. A block is a set of mutations (i.e. transactions) that transition the ledger from one state to the other. In that respect, a blockchain system is described as:

- A distributed peer-to-peer network of nodes, that can be full or light;
- Where each node holds a (partial) copy of the shared append-only ledger;
- That is formed and continuously growing by validated transactions among actors in the network, that are combined into blocks. Also, a reference to the previous block is added;
- Facilitated by the full nodes that provide computing power for the processing and validation of these transactions;
- After validation, the full node – or validator – broadcasts the block of transactions throughout the network;
- Via a consensus mechanism, automated network-wide agreement is reached about which block of transactions complies with the rules of the mechanism, and thus is the right one to add;
- After which the transactions in the block are applied to the entries of each individual copy of the existing ledger. The chain of blocks represents all the mutations done on the genesis version of the ledger, hence the term "blockchain".

Transactions

Vitalik Buterin, the founder of Ethereum, which is the second generation blockchain that incorporates the use of smart contracts, clearly explains that from a technical perspective, the Bitcoin ledger can be seen as a state transition system [9]. One state consists of the ownership status of all existing Bitcoins, where a "state transition function" takes the state and a transaction and outputs a new state, as schematically pictured below.

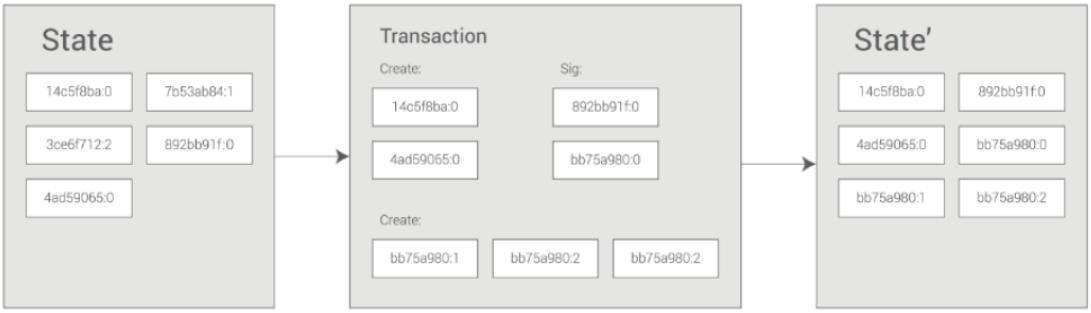


Figure 13: Schematic representation of a state transition in the Bitcoin system [9]

The new state is the updated Bitcoin ledger, showing the updated ownership status of all existing Bitcoins. In a standard banking system, the state is represented by the balance sheet. When someone wishes to transact, the state function reduces their account balance and increases the account balance of the receiving party. The state in Bitcoin is the collection of all unspent transaction outputs, or UTXO, with each UTXO having a denomination and an owner. The owner is defined by a 20-byte address which is the cryptographic public key. A transaction has inputs of one or more references to an existing UTXO, and a cryptographic signature produced by the private key associated with the owner's address. The transaction also has one or more outputs, with each output containing a new UTXO to be added to the state. Basically, the state of the ledger is updated with new address containing unspent Bitcoin and cleared of addresses that have spent their Bitcoin. The state transition function can be roughly described as [9]:

```
APPLY(S, TX) -> S' =
   1. For each input in TX:
         a. If the referenced UTXO is not in S, return an error.
         b. If the provided signature does not match the owner of UTXO,
            return an error.
   2. If the sum of the denominations of all input UTXO is less than the
      sum of the denominations of all output UTXO, return an error.
   3. Return S with all input UTXO removed and all output UTXO added.
```

Step 1a prevents transaction senders from spending coins that do not exists. Step 1b prevents transaction senders from spending other people's coins. Step 2 enforces conservation of value and step 3 returns the new state as ownership status.

Consensus

Because of the decentralized nature of the state transition machine, there has to be a way to reach agreement on the order of all transactions being done to prevent double spending from happening. In centralized systems, the company or person in control has gained the trust to decide on the correct order of transactions. However, Bitcoin was purposely designed to eliminate centralized control so the no single party could grab the power of the system. So how does a decentralized system become trustworthy? First, it is made extremely difficult and costly to tamper with the system. Second, by incentivizing good faciliatory behaviour. That is rewarding So how does a decentralized system reach agreement on the order of transaction? The state transition system needs to be combined with a consensus system for everyone to agree on the order of transactions. Nowadays, many different consensus systems exist of which the most common are; proof of work (PoW), proof of stake (PoS), delegated proof of stake (DPoS), proof of burn (PoB), Practical Byzantine Fault Tolerant (PBFT) or Raft [10][11].

In the Bitcoin system, consensus is reached through proof-of-work, by a process called mining. Nodes in the network, also known as miners, continuously attempt to produce packages of transactions called blocks. For each new block, a computationally intensive puzzle needs to be solved. The network is set to produce roughly one block every 10 minutes, with each block containing a timestamp, a nonce, a hash of the reference to the previous block and a list of all the transactions that have taken place since the previous block. Miners are increasing the nonce x until $H(x)<y$, where H is a secure hash function and y is a target hash. When y gets smaller, more hashes need to be calculated before finding the right x, that is the puzzle to be solved. All miners are simultaneously computing hashes until someone finds the correct one. Once a miner finds a solution for on for x, all the transactions since the last block are combined in the new block and the block is broadcasted to

the network. Eventually, a large ever-growing chain of interrelated blocks is created that constantly updates to represent the latest state of the Bitcoin system as pictured below.

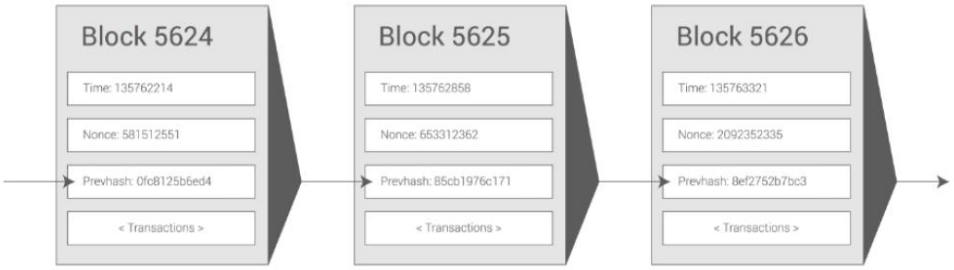


Figure 14: A chain of blocks containing transactions. Each block represents a saved state of the system. [9]

Any other miner who receives the broadcasted will verify if the block is correct and the transactions are valid through the process described below. If the block is found to be correct, the new block is added to their copy of the blockchain. And the process restarts.

```
Check if block is valid =
   1. Check the existence and validity of the previous block referenced by
      the proposed block.
   2. Check that the timestamp of the block is greater than that of the
      previous block and less than 2 hours in the future.
   3. Check that the proof of work on the block is valid.
   4. Let S[0] be the state at the end of the previous block.
   5. Suppose TX is the block's transaction list with n transactions. For
      all I in 0…n-1, set S[i+1]= APPLY(S[i], TX[i]). If any application
      returns an error, exit and return false.
   6. Return True, and register S[n] as the state at the end of this block.
```

The order in which the miner includes transactions in the block is very important, because when transaction B spends a UTXO created by transaction A, then transaction A has to be placed before B and not otherwise. If the miner includes B before A, the block gets rejected. Every miner creates and proposes blocks which are broadcasted throughout the network, but only the correct one will eventually be added to the blockchain.
The miner that found the correct block is rewarded with an amount of Bitcoin, which gradually decreases with the network's age. He is also rewarded with the transaction fees paid for every transaction in the block. Because of this economic incentive, it is interesting for miner to facilitate the Bitcoin network. And, moreover, it becomes far more lucrative to simply comply and facilitate than to perform undermining behaviour. The mining process is schematically pictured in figure 15.
In a situation where two miners produce a correct block at the same time, a natural fork of the chain is formed. However, the chain that is subjected to the most computational power shall finally prevail and the forks shall be blended with the dominant chain again.

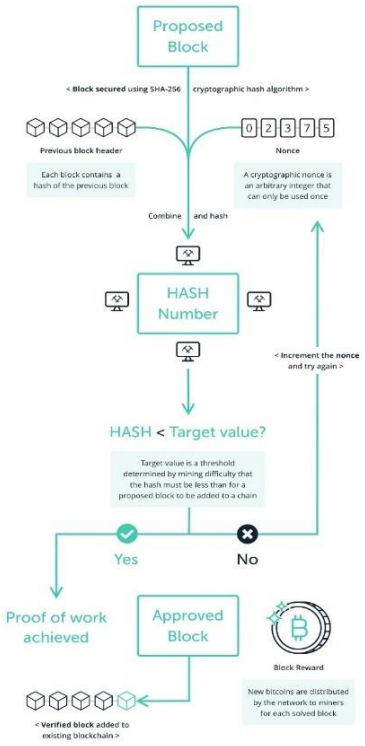


Figure 15: The mining process.

Security

The security of blockchains comes from their decentralized, immutable and tamper-proof features. To assess the current state of the security, the CIA security triad model is used. The model is composed of three areas: confidentiality, integrity and availability.

Figure 16: CIA security triad model

Confidentiality

According to the National Institute of Standards and Technology (NIST), confidentiality in information technology means "*the property that sensitive information is not disclosed to unauthorized individuals, entities, or processes*"[6].
Current blockchain technology prevents this in three ways. First, in private, or permissioned, blockchains authentication and authorization controls could be set up to prevent unwanted parties from network access [12]. Second, end-to-end encryption of the blocks is provided in certain blockchains, meaning that the data contents of a block remain fully encrypted while in transit. So even when the data is flowing through an untrusted network, it remains confidential. And third, by the use of private and public encryption keys. The endpoints of the transaction, the users, both own a pair of public and private keys. If user A wants to send data to user B, he creates a transaction to the public key (recipient address) of user B and he signs the transaction with his own private key. Only those who hold the private keys of the involved public keys are authorized to decrypt and see the data of that transaction.

Integrity

The NIST describes integrity as "*guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity".* For information systems it is of extreme importance that data always remains consistent and integer during lifetime. Regardless of the stage the data is in; in transit, work or rest storage, data integrity can be assured by for example encryption, hash comparison, or digital signing. Blockchains provide data consistency and integrity in two ways. First, the blocks and transactions are interrelated by incorporating the hash of a previous block into the hash of a new block combined. That interrelated chain combined with a large incentivised decentralized network prevents improper information modification or destruction. Second, every transaction added to the blockchain is digitally signed and timestamped so it can be traced back to a specific time period and the involved parties can (only) be identified by their public key. In this manner, it is assured that someone cannot duplicate the authorship of a transaction they originated. The traceability increases the integrity of the blockchain as fraudulent transactions are associated to a user's public key. With every new block addition, the global state of the ledger changes and the previous state is hashed and stored in the new block, resulting in a fully traceable history log. The ease of auditability provides a level of transparency and increased security for involved parties.

Availability

Availability is defined by the NIST as "*ensuring timely and reliable access to and use of information*". Because of decentralization and operating on a peer-to-peer network, blockchain is highly accessible and operational resilient. If part of the network is down or under attack, those nodes could be made redundant and business could continue as normal. So even when a node goes offline, all the information can be accessed at the next nearest node. Also, the decentralization provides no single point of failure. However, treats definitely exist.

[6] Source: https://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7298r2.pdf

A well-known attack on the availability of an internet service is a DDoS attack, where a server is overloaded with requests that it becomes inaccessible. Given that blockchains are distributed platforms, a DDoS attack on a blockchain is much harder and more costly to do. One way is to send a lot of empty transactions to overload the network, although it would cost the attacker a lot of transaction fees. Nevertheless, Ethereum suffered from a DDoS attack in 2016 and the Bitcoin network in 2014, so adequate protection measures are still necessary. And because of the growing base of unsecure installed IoT devices, online availability of DDoS malware and the availability of even higher bandwidth speeds, DDoS attacks will remain a persistent treat [12].
Another major issue would be a global internet outage, even for public blockchains. Therefore, private blockchains need to ensure that their network is sufficiently distributed globally and resilient with no single point of failure.

Scalability

As of today, blockchain scalability remains an issue that needs to be resolved to reach widespread adoption. For most computer systems, scalability refers to the system's capability to handle a growing amount of work. If it can't handle the growing amount of work by simply adding additional resources, the system has a limited scalability. In blockchains, scalability is generally simplified as the transaction throughput per second (TPS) it can handle, while remaining secure and accessible. At time of writing Bitcoin handles on average 7 TPS and Ethereum reaches 15 to 25 TPS, which is in sharp contrast with payment provider Visa's 1700 TPS[7]. There are blockchains in existence with a much higher throughput, like the Ripple blockchain, which tested at 1500 TPS but lack on other essential features. Blockchain's scalability is limited in a couple of ways:

- Limited block size, that caps the maximum amount of data in a block
- Increasing size of the blockchain, that requires increasing hardware capabilities
- Response time
- Transaction fees, that increase with the amount of traffic on the blockchain
- High electricity usage for PoW consensus based blockchains, such as public Ethereum

Blockchain classifications

As already mentioned in the material above, blockchains can be classified as either permissioned or permissionless. On top of that there are additional distinctions to be made. Blockchains can be classified based on access to the blockchain data and access to transaction processing, which lead to the following class definitions [13].

Class	Definition
Public blockchain	A public blockchain is a blockchain, in which there are no restrictions on reading blockchain data and submitting transactions for inclusion into the blockchain. Published data may however be encrypted.
Private blockchain	A private blockchain is a blockchain, in which direct access to blockchain data and submitting transactions is limited to a predefined list of entities
Permissionless blockchain	A permissionless blockchain is a blockchain, in which there are no restrictions on identities of transaction processors, those are users that are allowed to create blocks of transactions
Permissioned blockchain	A permissioned blockchain is a blockchain, in which transaction processing is performed by a predefined list of subjects with known identities.

Table 2: Blockchain classes

[7] https://hackernoon.com/who-scales-it-best-blockchains-tps-analysis-pv39g25mg

The classes above do not provide sufficient coverage of the different categories of blockchains in existence, further detailing gives the following republished table complemented with examples of existing projects.

	Access to transaction processing:	
Access to transactions:	**Permissioned**	**Permissionless**
Public	Public read access but permissioned network facilitation (e.g. Internet, Corda)	Existing cryptocurrencies (e.g. Bitcoin) and smart contract platforms (e.g. Ethereum)
Regulated	Direct read / transaction creation access for clients (e.g. Hyperledger Fabric)	Ability to create transactions can be regulated (e.g. stablecoins such as USDC, built on Ethereum)
Private	Access limited to transaction processors; benefits of blockchain technology are diminished (e.g. Hyperledger Fabric)	Not applicable

Table 3: All blockchain categories [13]

Whether a blockchain is either permissioned or permissionless makes a lot of difference in terms of security, scalability, customizability and operational effort. In permissionless setups, anyone can use and/or facilitate the network. This comes with the advantages that most users or applications don't have to spend effort on hardware and running the network. The networks are usually huge, geographically decentralized and subject to algorithmic consensus, making them secure but slow. The downside is that the permissionless network isn't owned by anyone, merely facilitated by a large group of pseudo-anonymous validators that have to vote on updates and decisions concerning the network, which limits the scalability and customizability. On the other hand, permissioned networks only have a few facilitators which makes them much faster and scalable, but therefore lack the decentralization and the security that comes with it. Permissioned blockchains are regarded as suitable for enterprise applications, where permissionless blockchains are usually regarded as most suitable for public and government applications.

2.1.2.2 Enterprise blockchain applications

Now that a fundamental understanding is reached about blockchain technology merits, demerits and capabilities, let's look at some examples of implemented enterprise blockchain applications specifically applied to multi-party business processes, to see what it can deliver in practise.

Naviporta

Co-founded by Blocklab, Naviporta focusses on the digitalization of documentation required for logistics and customs processes for international shipments, and the necessary accompanied system integrations. Their goal is to seamless integrate the physical, information and financial flows, for which they created an open and neutral platform to exchange assets and information in a digitally trusted and secured way[8]. The platform provides for real-time and end-to-end visibility of containers, access to real-time validated information for supply chain participants and authorities, and immutably record what has been done by whom via the digital notary. The digital notary, built on the open-source public Ethereum blockchain, also allows for notarisation of shipping documents like eCMR and bill-of-lading. Their solution is expected to reduce end-to-end documentation processing time from 5-10 days to less than 24 hours, and with a large 50% industry-wide adoption could potentially save $4 billion per year[9].

[8] https://naviporta.com/
[9] https://naviporta.com/2021/05/ebl-trial-rotterdam-singapore-naviporta/

Tradelens

Where Naviporta focusses on the digitalization of logistics and customs required documentation for shipments, Tradelens focusses on sharing of all the valuable events and information around these shipments such as events regarding consignment, transport equipment, shipments and transport plans. A permission matric and blockchain are utilized to ensure every party in the interconnected ecosystem of supply chain partners has access only to their information and a secure audit trail of all transactions, enabling unprecedented collaboration and sharing of data. They use the IBM blockchain platform that is based on the open-source permissioned Hyperledger Fabric blockchain, where the network validators are known and certified. Currently they have 2.4 billion events tracked, 22 million documents published and 46 million containers processed on their platform. For medium sized logistic service providers using Tradelens, yearly operational savings can run over $1 million[10].

CONA

For the twelve largest Coca-Cola bottlers in North America representing over 500 bottling sites, a supply chain platform was created by CONA Services together with Provide Services[11]. In 2019, the first set op bottlers adopted a Hyperledger Fabric platform to streamline the relationship between franchised bottling companies to make cross-organization supply chain transactions frictionless and transparent. To extend the use-case from the internal network to a larger audience, they utilize the Baseline Protocol to establish a “bottling harbour” enabling a low-barrier network onboarding process for bottling suppliers and other external suppliers.

The relationship between buyers and suppliers results in referencing business objects like Request, Proposal, Purchase Order, Delivery, Invoice and Payment, but lead to various to coordination and integration challenges:

- Undesired and unnoticed changes on Purchase Orders may lead to a buyer receiving the wrong delivery;
- Deliveries may be different from orders due to manual errors on supplier’s side;
- All process required documentation and information needs to be repeatedly distributed manually, by email for example;
- Small participants (without ERP) are prevented from integrating with the ecosystem due to technical and cost barriers.

Digitalization of these business objects should eliminate - among others - complex disputes, information asymmetry, redundant manual data distribution and manual errors. The Baseline Protocol proves to be an outstanding technical backbone for the desired solution by:

- Using public Ethereum blockchain as an always-on, pay-per-use frame of reference;
- Keeping enterprise data in traditional systems of record;
- Enabling complex, private, interorganizational business process automation;
- Providing extensibility for Decentralized Finance and asset tokenization use-cases;
- Is openly-governed open source, on its way to become an OASIS standard.

The expected results of the platform are $650 million of tokenized invoice value 2021, $100 million in yearly operational savings and supply chain dispute reduction of 97%.

[10]https://www.tradelens.com/

[11] https://provide.services/news/baselining-the-north-america-coca-cola-bottling-supply-chain

2.1.2.3 Baseline Protocol

Inspired by the CONA business case, where supply chain ERP systems are connected together via a common frame of reference – i.e. public Ethereum network – to streamline and digitalise common business practices, this section analyses the Baseline Protocol in detail.

As of this moment, there are quite some projects focussed on merging multilateral enterprise processes with blockchain technology. The University of Cambridge analysed live projects in a second global enterprise blockchain benchmark study [14], which gives a clear view of the current landscape supported by the figures below. The dominant use-case, although fragmented, for these networks appears to be supply chain tracking. The main supported protocol framework for these applications is Hyperledger Fabric. For 72% of all live projects, cost reduction is the predominant objective, followed by respectively hybrid objectives (14%) and novel market models (8%). What is interesting to observe is the dominant projects represented in figure 22 are permissioned regulated blockchain

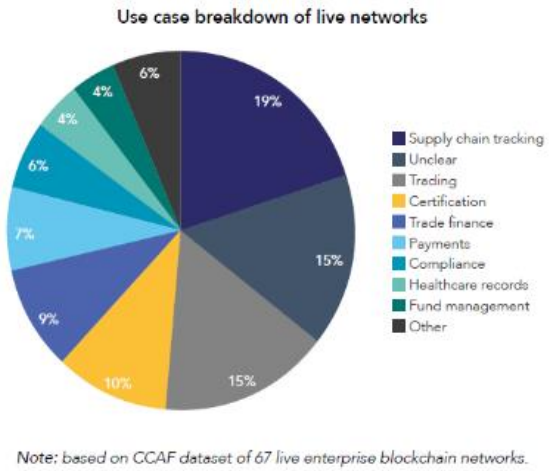


Figure 17: Enterprise blockchain use-case breakdown

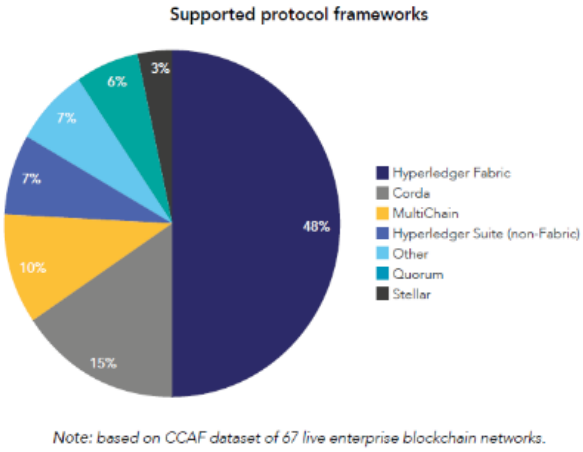


Figure 18: Enterprise blockchain protocol breakdown

networks, also known as consortium blockchains. Stellar on the other hand has features from both type of systems. Any node may enter the network permissionless without having to pass through a central gatekeeper. However, for a node being able to validate transactions it is obliged by the consensus protocol to share the same network state value as the majority of the nodes, which can be regarded as the permission part.
It is no coincidence that permissioned regulated blockchains are in favour for enterprise applications. In comparison to public blockchains, they allow for consortium control over features like governance, ledger transparency, network updates, and transaction cost. A risk with this type of setup is that the number of independent nodes can be or become very small, to a point where it defies the main principle of blockchain; decentralisation. A disadvantage of this type of setup is that it requires the participants to run the entire digital infrastructure themselves. This forces them to get knowledgeable about the technology; purchase and install the needed hardware; operate, monitor, and maintain the software and hire software engineers. For the majority of SMEs, especially those not operational in the IT domain, such an investment is infeasible from a business perspective.

What the Baseline Protocol sets apart is that it's a set of configurable, mainly open-source, techniques that include the public permissionless Ethereum blockchain, specifically designed to *baseline* different systems of record, such as ERP, CRM and other internal systems. Because it includes the relative mature Ethereum blockchain, it is at the upper boundary of decentralisation and cryptographical trustworthiness, and allows for business logic executed by smart contracts. Additionally, the existing hardware and digital infrastructure can be utilized on a pay-per-use manner because the (fluctuating) transaction costs are the only cost for operating on the network, allowing

for SMEs to be included into the system.
A business process is considered *baselined* when two or more systems store data and run business in a verified state of consistency, enabled by using a network as the common frame of reference. The ability to connect different ERP systems, on a trustworthy pay-per-use network that allows for involvement of SMEs, while most of the business logic being automated and enforced by smart contracts can be a major relief within the current contractually rigid context of offshore wind maintenance.

The components that together form the Baseline Protocol are divided in on-chain and off-chain components. On-chain components are the smart contracts that, once deployed, reside on the blockchain network and are executed by the Ethereum Virtual Machine. The off-chain components facilitate the ability to perform a multilateral business process enforced by these on-chain components. The on-chain components are:

On-chain BP components	**Function**
OrgRegistry contract	"Rolodex"contacts list of involved participants
Shield contract	Gatekeeper that calls the Verifier contract if a participant listed in the OrgRegistry sends a proof. Also holds a fingerprint of the current state of the Merkle Tree
Verifier contract	On-chain component of the ZeroKnowledge-service that a baseline proof verification is only deposited on the network if involved counterparties have performed the Workflow Step consistently and have adhered to the rules of any previous Workflow Step

Table 4: On-chain components of Baseline Protocol

The off-chain components that facilitate operations with on-chain components are listed in the table below. BP leverages best in class open-source products for the implementation of these components, for example the utilization of NATS as messenger service.

Off-chain BP components	**Function**
Messenger service	Decentralised private automatable messaging between participants, ideally sends data point-to-point without intermediate storage, able to specify different participants and workflow steps, balance between liveness and security, and handle long session management.
ZK service	Privacy tool to provide zk-SNARK functionalities (Zero Knowledge Succinct Non-interactive Argument of Knowledge). Zk-SNARKS are mathematical concepts and tools to establish zero knowledge verification of succinct proofs, which convert logical statements to arithmetic circuits, that are then leveraged to generate proofs. It allows to prove logical statements without disclosing any information and yet proving the validity of the proof. The service provides functionalities to convert business logic into arithmetic circuits, create Verifier contracts and generate proofs that are to be verified in the Verifier contract. In essence it converts business logic into unrecognizable complex mathematical tests (Verifier contract) and answers (proof).
ERP connector	Component that integrates existing ERP systems and their user interfaces with the BP components. It allows for data exchange between the ERP and BP components, and a BP UI through the existing enterprise software eliminating the need for a separate system for employees to work with.

Items below aren't represented in literature's component list, but are considered to play a vital role and are also included in the example reference implementations Radish34 and BRI-1. Purpose of BP is to provide a framework for building baselined offerings, therefore, the components below have been included.	
Blockchain client	Component that integrates the (Ethereum) blockchain with the off-chain components and allows for the ability to perform transactions and smart contract function calls, but also listens to the Verifier contract for emitted verifications and manages the associated commits.
Identity Service	Service that provides for verifiable off-chain digital identities and signatures in order for participants to be able to digitally sign legal documents
Databases	One database is used to store general BP process data such as messages, attachments, documents etc. received through the Messenger service. Another database is used to store a complete copy of the Merkle Tree that holds the commits emitted by the Verifier contract. Important to note that each connected participant holds a similar local copy of the Merkle Tree, and the fingerprint of it is stored and updated in the on-chain Shield contract

Table 5: Off-chain components of Baseline Protocol

Baseline process

The first step in baselining is setting up the Workgroup [15]. The **Workgroup** is the group of all the counterparties involved in a shared business process, which is usually the case in supply chains or consortia. When baselining a supply chain, each participant has to be added to the Workgroup. The business process they all perform together is called the **Workflow**. The Workflow consists of minor and major **Worksteps**, which are the separate steps that together form the entire business process the supply chain performs. After each successfully verified Workstep, an event notification shall be broadcasted throughout the network and a **Commit** (receipt of the verified Workstep) shall be stored locally in each system. Commits allow each connected system to keep up with the state of the Workflow. Let's take an international shipment as example. Two counterparties make an agreement on the purchase of a certain item. That item has to be picked up by a truck, transported to a harbour, loaded onto a vessel, offloaded at the import harbour, go through an import customs check, picked up by a truck again and eventually delivered to the receiving party of the trade. The Workgroup involved in the shared business process of executing this shipment is at a minimum: two businesses involved in the trade, two trucking companies, one carrier company, one customs agency, and two sea terminal operators. The Workflow they together perform is the execution of the shipment. Some of the major Worksteps involved in this Workflow are the signing of the business deal, the pick-up of the item and the delivery of the item.

From a detailed technical perspective, the baseline process goes as follows. One initiating party sets up the Workgroup by either:

- Adding an entry to an existing OrgRegistry smart contract on the Mainnet;
- Selecting existing entries on a universal OrgRegistry;
- Creating a new OrgRegistry and adding entries to it.

The OrgRegistry can be regarded as a corporate phone book, where each entry represents a connectable business entity with their contact details.

The next step is to establish point-to-point connectivity with the counterparties in the Workgroup by obtaining their endpoint for the OrgRegistry and send an (email) invitation to each counterparty. The invitation contains a JSON Web Token, that takes care of the configuration of the counterparty's system. Now the counterparties are connected securely.

The following step is to set up a Workflow. A Workgroup may run one or more Workflows, that contains one or multiple Worksteps. As the Workflow represents the shared business process, the business logic and rules have to be included in the Workflow. For secure and consistent execution of

the Workflow, major Worksteps need to be verified by a trusted party. The inputs for a major Workstep usually are commercially sensitive information, like the combination of a purchase order and an invoice that acts as proof of a business agreement. For a trusted party, that is the public blockchain, to have the ability to verify executed Worksteps without revealing the commercially sensitive inputs zero-knowledge circuits and proofs are used. The busines logic for a Workstep is transformed into a mathematical zero-knowledge circuit, the counterparty that wants to prove the execution of a Workstep has to enter parameters into the circuit, and generate and provide a zero-knowledge proof.

Once the business logic is rendered mathematically, the smart contract that guard and verify the Workflow can be deployed. As mentioned, first an OrgRegistry contract is deployed. Second, a Verifier contract will be deployed that is able to verify zero-knowledge proofs of a counterparty as proof of a successfully executed Workstep. After each successful verification, the Verifier contract emits a verified event and connected systems will store a Commit in their local MerkleTree. Third, a Shield contract is deployed that authorizes OrgRegistry registered parties to send proofs to the Verifier contract and holds a fingerprint copy of the MerkleTree that is held locally at each connected system. The smart contracts act as an authority party protecting and enforcing the connected parties to comply with the shared business logic they agreed upon.

Now that the WorkGroup, Workflow and decentralized enforcers of the Workflow are established, it is time to become operational and run the Worksteps. Applying it to the example of a shipment again and on figure 19 below; Counterparty 1 and 2 make a trade, share an invoice together and send a zero-knowledge proof of that invoice to the Verifier contract on the Mainnet. The system of trucking company Counterparty 3 receives in real-time a valid verification of that invoice, but not the invoice or the content itself. The trucking company automatically knows about the trade and knows that he can prepare to pick-up some item at Counterparty 1. The trucking company, Counterparty 3, prepares a transport order and sends the original document to Counterparty 1 and subsequently sends a zero-knowledge proof of the transport order to the Verifier on the Mainnet, after which all connected systems - and in particular receiving Counterparty 2 – are made aware in real-time about the upcoming transport of the first segment of the shipment. All these activities are supported and executable while operating through their existing ERP systems.

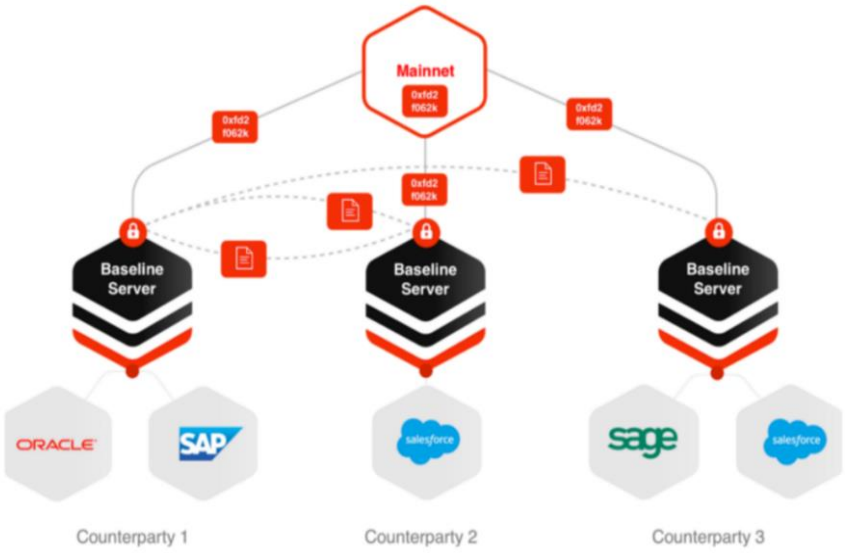


Figure 19: Graphical representation of ERP systems in a Workgroup connected through the Baseline Protocol

To conclude on the Baseline Protocol; it's a pay-per-use infrastructure because no hardware has to be acquired while making use of open-source software. It facilitates operation through existing ERP systems and other systems of records. It can be integrated with any level of system, from the advanced SAP to Google Sheets. It allows for business process automation and real-time notification for all participants, secured and enforced by smart contracts, while sensitive data never leaves the premises of a participant.

2.1.3 System-of-systems design methods

This section describes the relevant literature that is analysed to determine the right design method for the envisioned system-of-systems. First is looked at the well-known system engineering method, followed by a more specialised, agile, blockchain system engineering method.

The determined research problem identifies a lack of a secure system-of-systems for automated matching of maintenance demand with supply. In the previous section was concluded that blockchain technology potentially is an enabler for the envisioned process automation between multiple participants in a supply chain, and it also might fulfil the role of the entity that processes sensitive data in a trustworthy and securely manner. The systems to be connected in this context are information systems such as the Asset Management System of the WPP, the ERP systems of both the WPP and the ERP systems supply chain participants, and the blockchain system itself.

A key requirement for the final design is feasibility, so the selected design method should aim for a feasible output. The method should also be suitable for the domain of information systems in order to have a smooth and fit for purpose design process. Since the limited resources for this research, the method should apply an agile approach to get the best result within a short timeframe.

2.1.3.1 Systems Engineering

The first design method that is taken into consideration is the Systems Engineering (SE) method. The International Council on Systems Engineering defined SE as:

"..a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods." [12]

As the definition states, the comprehensive SE method takes many disciplines into account to address the entire lifecycle of an engineered system. The dominant model for systems engineering is the V-model, derived from the linear Waterfall Model and proposed by Paul Rook in 1986 [16]. The original model was designed for software development, but went through minor changes to adapt to modern technologies and different domains. Since its inception the model is adopted by governments, militaries, space agencies and technology industry leaders all over the world to develop their multidisciplinary systems.

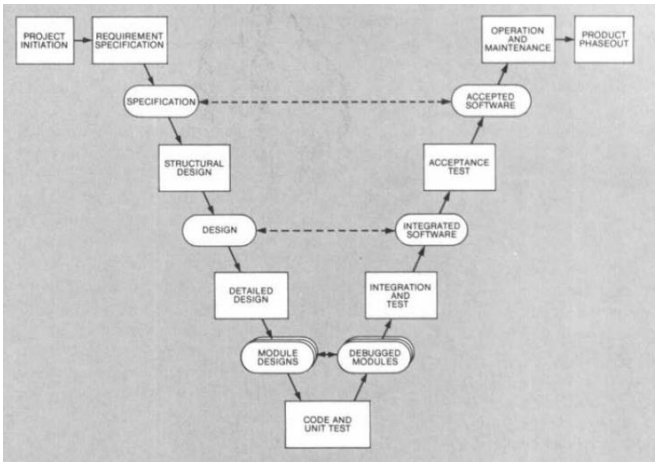


Figure 20: Original V-model by Paul Rook[17]

[12] https://www.incose.org/about-systems-engineering/system-and-se-definition

The key development phases in the V-model as described by Rook are:

1. **Requirement specification phase:** a complete, validated specification of the required functions, interfaces and performance of the product.
2. **Structural design phase:** complete, verified specification of the overall hardware-software architecture, control structure and data structure for the product.
3. **Detailed design phase:** complete, verified specification of the control structure, data structure, interface relations, sizing, key algorithms and assumptions for each program component.
4. **Code and unit test phase:** complete, verified set of program components.
5. **Integration and unit test phase:** a properly functioning software product.
6. **Software acceptance test phase:** an accepted software product handed over to the customer.
7. **Maintenance phase:** a fully functioning update of the software product.
8. **Project termination phase:** a completed project history document benchmarking the initial goals and plans to the actually realized goals and plans
9. **Product phase-out:** a clean transition of the functions performed by the product to its successors (if any).

Although the SE method is very suitable for this research, due to its level of abstraction it is deemed somewhat unpractical to result in a feasible design. Additionally, there is a lack of agility because of the phased and sequential approach.

2.1.3.2 Agile Block Chain Dapp Engineering (ABCDE)

An even more suitable design method than the SE method is the “agile block chain DApp engineering method”, or the ABCDE method. It is deemed more suitable because the method is specifically developed out of a lack of disciplined, organised and mature development process for blockchain based products. On top level the method contains similar design steps as the proven SE method, therefore it can be regarded as a practical implementation of the SE method, specifically applied to blockchain based systems.
The ABCDE method was proposed in 2019 by Marchesi et al. [17] for the trending area in software development of decentralized applications, or “Dapps”, that typically run on a blockchain. The researchers included agile practices, because they are suited to develop systems whose requirements are not completely understood at the beginning, or tend to change. This also applies to the development of the design for a novel, unprecedented s-o-s where this research is aiming for. Additionally, according the researchers ABCDE is based on Scrum, and is therefore iterative and incremental.

The comprehensive ABCDE design process is captured in the following illustration:

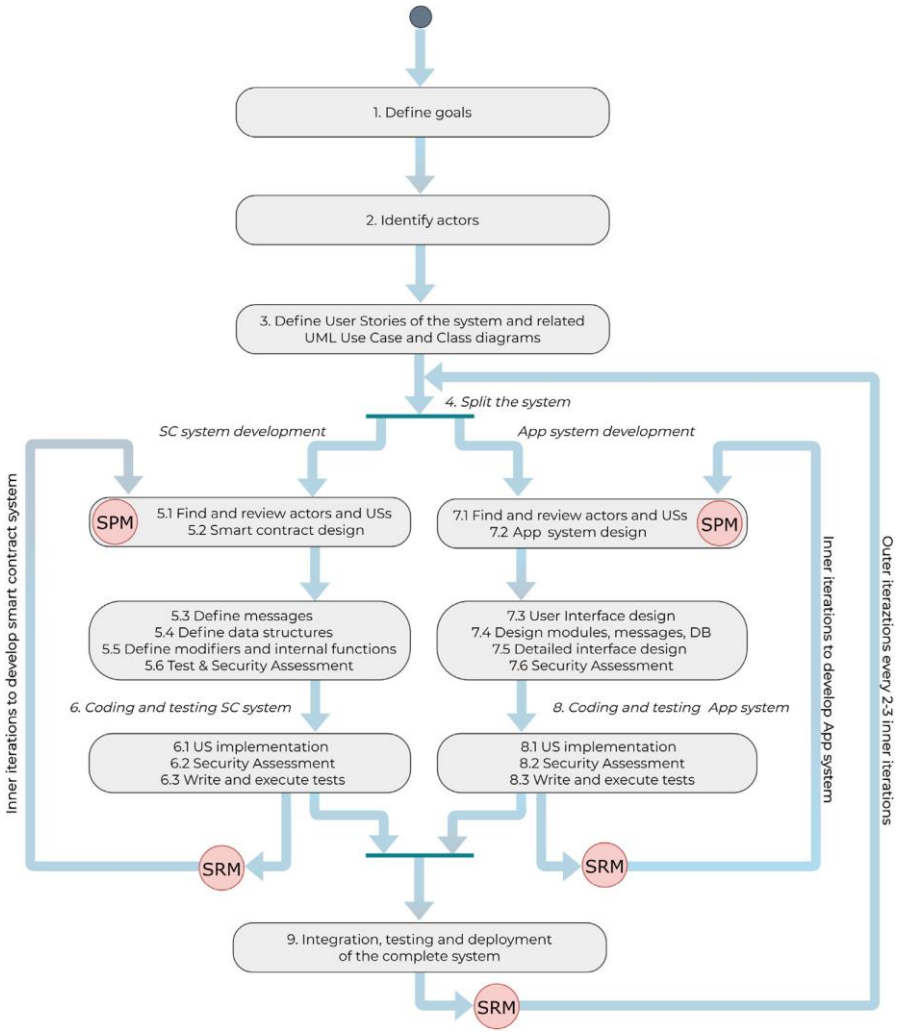


Figure 21: Summary of the ABCDE design process. The circles represent the sprint meeting, SPM = Sprint Planning Meeting, SRM = Sprint Review Meeting [17]

A detailed explanation of every step in the ABCDE method is given below.

1. **Goal of the system:** write down a short description of the goal of the system and display it for the whole team. The idea is borrowed from Scrum practices and a practice in object-oriented analysis.
2. **Find the actors:** identify the actors who will interact with the system. The actors are human roles and external systems or devices.
3. **User stories:** the system requirements are expressed as user stories regardless of the technical implementation, to be able to follow the classical agile approach for project management in Scrum. It might be useful to use a UML Use Case diagram to graphically show the relationships among the actors and user stories.
4. **Divide the system in two subsystems:** first the blockchain system, that is predominantly represented by the smart contracts running on the blockchain. Second, the App system, that is the external “off-chain” system that interacts with the blockchain.

At this stage, an architecture of the whole system and a data model should be drafted. The guideline is that the smart contracts (SCs) should manage the data and processing that need to be transparent and immutable for the DApp to be trusted by its actors.

5. **Design of the smart contract (SC) system:** a multi staged step about designing the SCs, through iterations that include coding and delivering increments of SCs.
   5.1. Replay step 2 and 3 by focusing only on actors directly interacting with the SCs.
   5.2. Define broadly the SCs composing the SC subsystem. For each SC, state its responsibilities to store information and to perform computations, and the related collaborations with other SCs.
   5.3. Define the flow of messages and cryptocurrency transfers among the SCs.
   5.4. Define in detail the data structure of each SC, its external interface and the relevant events that can be raised by it.
   5.5. Define the internal, private functions and modifiers – special functions that usually test the preconditions needed before a function can be safely executed.
   5.6. Define tests and perform the security assessment practices.
6. **Coding and testing the SC system:** following the agile approach, the SC system is built and tested incrementally. Activities for this step are:
   6.1. Incrementally write and test the SCs.
   6.2. Perform security assessment and gas optimization.
   6.3. Write automated Unit Tests and Acceptance Tests for the SCs and user stories implemented.
7. **Design of the app system:** the app system interacts with the users and devices, send messages to or listen to events from the blockchain, and can manage its own databases.
   7.1. Redefine the actors and the user stories for the app system.
   7.2. Design the high-level architecture for the app system, including server and client tiers, and detail the way it accesses the blockchain.
   7.3. Define the UI of the app system.
   7.4. Define how the app system is decomposed in modules, their interfaces and the flow of messages between them.
   7.5. Perform a security assessment of the app system.
8. **Coding and testing the app system:** In parallel to the SC system, the app system is built and tested incrementally using the same approach of the SCs development. Activities for this step are:
   8.1. Incrementally write and test the app system.
   8.2. Perform security assessment and gas optimization.
   8.3. Write automated Unit Tests and Acceptance Tests for the user stories implemented.
9. **Integrate, test and deploy the combined DApp system:** to integrate the two separately designed systems, the system must be deployed into a local or a testnet blockchain, and integration tests must be run to check whether all the components interact together as expected.

The similarities between the SE method and the ABCDE method are the following. ABCDE step 1 – 3 are a more practical implementation for SE step 1. After ABCDE step 4 similar progress is made to SE step 2. ABCDE then splits up the on-chain and off-chain system, so ABCDE step 5 and 7 are similar to SE step 3. ABCDE step 6 and 8 represent SE step 4, and ABCDE step 9 is similar to SE step 5.

2.1.4 System-of-systems theory extension

After analysing the literature on system-of-systems theory, blockchain technology and system design methods, a lack in available definitions was discovered for describing and distinguishing parts of an s-o-s. The established definition framework can be viewed as a pyramid, where every layer defines increasingly smaller parts of an s-o-s. The operational success of each layer depends on the presence of vital parts in the layer below. A road logistics system-of-systems consists of a road system, vehicle systems and warehousing systems; all operated and managed independently. The road system does not function without its street lighting subsystem. And the lighting subsystem does not function without its LED elements. Every present system in an s-o-s can be described by one side of the pyramid, meaning that an s-o-s consisting of N systems can be described by a N-sided pyramid as illustrated in figure 22.

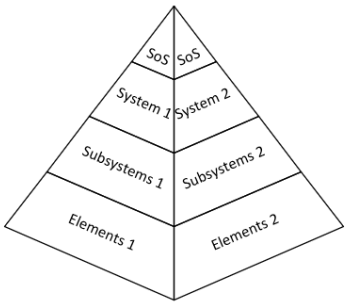


Figure 22: S-o-s three-dimensional definition framework, each side of pyramid represents an individual system in the s-o-s

According the theory of "Delft Systems Approach" [7], the smallest distinguishable parts of an s-o-s are the elements. This level of detail is sufficient for most physical s-o-s, however for s-o-s that include some form of collaborative information processing, a definition is lacking. What lacks is a definition to describe the smallest, passive, standardized pieces or structures of data that are required to be consumed by other parts of the s-o-s for its operational success. That distinguishable group of s-o-s parts is defined as "**Data Objects**", which is an additional definition layer at the bottom of the s-o-s definition framework as illustrated in figure 25. Data objects mostly appear in the form of template data structures or unique identifiers. Data objects are among the smallest parts of an s-o-s, usually only a few characters long. They are passive because they have no in- or output, they cannot execute a function. They are standardized because they are consumed by various elements and component systems that make up a s-o-s, and without them the s-o-s is unable to operate properly.

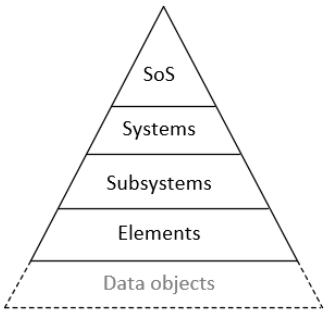


Figure 23: Extended definition framework for identifying and describing System-of-Systems parts

Returning to the road logistics s-o-s example, part of the road system are the standardized road and traffic signs. They're small relative to the size of the s-o-s, passive, standardized and consumed by elements of the vehicle systems. Without them, the road logistics s-o-s would be useless because the vehicle systems can't navigate. In a supermarket s-o-s, or a warehousing s-o-s data objects appear for example as product/price tags and aisle numbers.

For this research the distinguishable data objects, vital to the operational success of the envisioned s-o-s, are the templates for the maintenance and availability schedules, smart contract and wallet addresses, hashes, Commit hash templates, and other data structures. Since its importance to the operational success, data objects need to be thought of and designed separately, and therefore deserve their own definition in the s-o-s framework of identifying and describing s-o-s parts.

2.2 Research approach

Now that the theoretical and design framework for this research is determined and explained, the research approach shall be defined in this section. First, the identified main research question is revisited:

How to design a technical feasible decentralized system-of-systems that enables automated matching and contracting of maintenance supply for scheduled demand through trustworthy processing of sensitive data?

Because this is a design research for the development of a system-of-systems design, the research approach consists of a detailed design scope and a design approach. The design scope shall explain what is exactly designed for the system-of-systems that enables automated matching and contracting of maintenance supply for scheduled demand. The design approach shall describe the exact approach taken to come up with a technical feasible design.

2.2.1 Design scope

According the definition framework of systems-of-systems theory, the design scope for this research shall consist of developing designs for the three separate systems that together form the system-of-systems:

- **Demand system**, which is a generic system for the maintenance requiring asset owners
- **Supplier system**, which is a generic system for suppliers in the maintenance supply chain
- **Blockchain system**, which is the system that enables decentralized process automation and trustworthy management of sensitive data

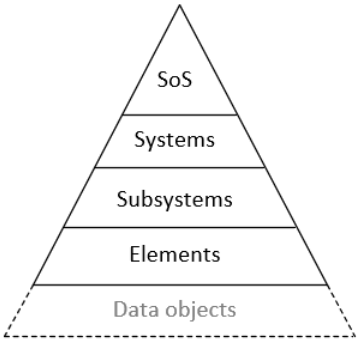


Figure 24: System-of-systems definition framework

The designs for each of these systems shall be in Unified Modelling Language (UML) format and supporting graphical representations. For each system design, subsystems shall be defined and specific elements and data objects shall be designed, all in order to develop a technical feasible s-o-s design that is able to achieve its goal.

The envisioned three-system s-o-s design captured in the theoretical framework can be visualised as a three-sided pyramid as shown below.

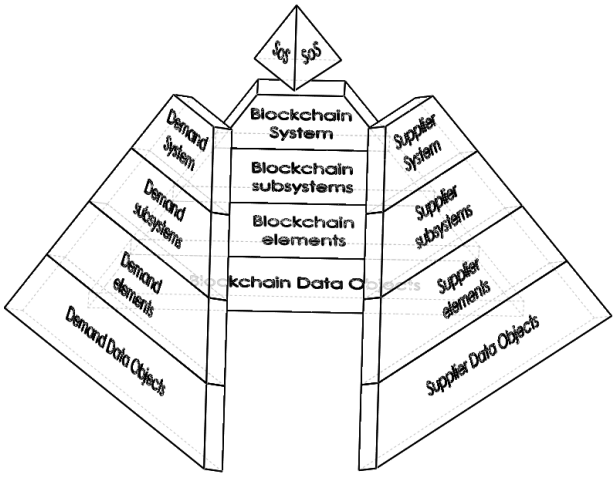


Figure 25: Visualisation of the three systems that make up the envisioned s-o-s, within the theoretical framework

Additional to the individual system designs, a design has to be created for the s-o-s part that forces these systems to operate together and act as one. Since this is a design for a decentralised s-o-s, with no central overarching system at the top, the s-o-s design is actually the automated information sharing process (the Workflow) that operates on element level in the s-o-s. So the final piece of the design scope is the design of:

- **Demand and supply matching system-of-systems,** top level architecture and automated Workflow that enforces the connected systems to operate as one on element level.

The Workflow design shall be in UML supported by a graphical illustration of the top-level architecture of the s-o-s. The start of the Workflow begins with a machine-readable maintenance schedule and ends when suppliers are secured for the scheduled maintenance operations. Since the Workflow operates on element level, the total s-o-s design can be visualised as shown below. It's also important to notice that the top of the pyramid, to represent the s-o-s, has been removed. Because the design is about a decentralised system-of-systems no dominant central system is positioned on top of the others. The design is about facilitating equal collaboration between connected systems to reach a common goal. Collaboration between the systems of these supply chain participants and competitors is only possible when the processing of commercially sensitive data happens in a secure and trustworthy manner, which is hardly possible with a centralised system-of-systems owned by one party.

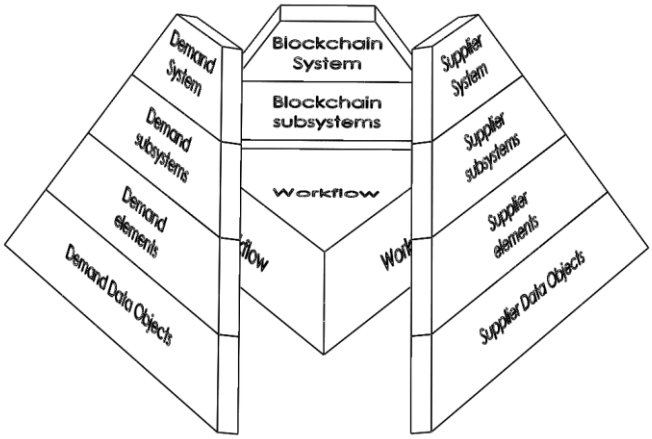


Figure 26: Visualisation of the total scope of the s-o-s design, within the theoretical framework

2.2.2 Design approach

This section elaborates on the approach taken to develop the designs for the s-o-s, and its internal systems, elements and data objects to achieve a technical feasible s-o-s design that enables automated matching of maintenance demand with supply.
For the design approach, the agile blockchain application engineering method ABCDE [17] shall be followed closely, merged with some Baseline Protocol design steps [15], while taken into consideration the s-o-s design principles of Maier [8]. The merged design approach followed for this research is defined below.

1. Define the goal of the s-o-s (ABCDE step 1)
2. Identify the actors (ABCDE step 2)
3. Define initial s-o-s architecture on high-level (BP design feature)
4. Define User Stories of the s-o-s (ABCDE step 3)
5. Define initial s-o-s Workflow on high-level (BP design feature)
6. Split the s-o-s in blockchain and non-blockchain systems (ABCDE step 4)
7. Design the blockchain system (ABCDE step 5 and 6)
8. Design the demand and supplier systems (ABCDE step 7 and 8)
9. Design integration for final s-o-s architecture and Workflow (ABCDE step 9)

As mentioned, for each of these steps the design principles from Maier for s-o-s shall be considered. This means at first, each of the separate systems shall be a stable intermediate form. Second, according policy triage we will only focus on what we should influence. Third, leverage at the interfaces leads to special focus on the interfacing blockchain system and the Baseline Protocol elements. And last, the s-o-s design should incentivize, and therefore ensure cooperation between the individual systems.

The technical feasibility of all designs is achieved by working together with a blockchain developer, Hamza Suwae, from BlockLab on the design iterations. The designs are being built through coding and testing, whereby the technical feasibility of the designs is being verified. The code is not included in the scope of this research, it will merely be used as a tool to verify the feasibility. On top of that, occasional virtual meetings will be held with core Baseline Protocol developers and architects for additional feasibility verifications, and for help when the development progress gets stuck. The people we'll consult with are:

- Sam Stokes – Software architect at ConsenSys
- Kyle Thomas – CEO and founder of Provide Services (known from the Baseline Cona implementation)
- Daven Jones – Product owner Provide
- Brian Chaimberlain – Software Engineer

Additional verification shall be done through evaluation of achievement of the defined user stories and through evaluation of compliance with the system-of-system design principles as defined by Maier [8].
The validity of the design shall be determined through the evaluation of its performance in a case study in an offshore wind industry context. For the case study, KPIs shall be defined by which the performance is compared for the current state manual process and the automated s-o-s design.

2.3 Answers to research questions

The following research questions have been answered in this chapter.

1. What is the theoretical framework to define and describe envisioned system-of-systems?

The combination of:

- The Delft System Approach, by Veeke, Ottjes and Lodewijks [7]
- Systems-of-systems theory by, Maier et al. [8], with the proposed extension of *data objects* to distinguish the smallest standardized passive, consumable, pieces or structures of data
- Bitcoin: a peer-to-peer electronic cash system, by Nakamoto [4]
- A next-generation smart contract and decentralized application platform, by Buterin [9]

gives all the necessary theoretical fundamentals to understand, define and describe the envisioned system-of-systems. "*System-of-system theory"* for understanding the concept, the properties and the design principles of system-of-systems. The "*Delft System Approach"* for defining the characteristics, surroundings and components of a system. "*Bitcoin: a peer-to-peer electronic cash system"* as introduction to the fundamentals of blockchain technology, and "*A next-generation smart contract and decentralized application platform"* as introduction to the fundamentals of smart contracts.

2. What is the most suitable design method for envisioned system-of-systems?

The design method determined as most suitable for the envisioned system-of-systems is a combination of the system engineering inspired ABCDE method of Marchesi et al. [17], combined with design features of the Baseline Protocol [15] for designing the shared, multi-party supply chain Workflow that shall me executed automatically. Top level design steps of the merged design method are:

1.	Define the goal of the s-o-s	(ABCDE step 1)
2.	Identify the actors	(ABCDE step 2)
3.	Define initial s-o-s architecture on high-level	(BP design feature)
4.	Define User Stories of the s-o-s	(ABCDE step 3)
5.	Define initial s-o-s Workflow on high-level	(BP design feature)
6.	Split the s-o-s in blockchain and non-blockchain systems	(ABCDE step 4)
7.	Design the blockchain system	(ABCDE step 5 and 6)
8.	Design the demand and supplier systems	(ABCDE step 7 and 8)
9.	Design integration for final s-o-s architecture and Workflow	(ABCDE step 9)

3. How will the design be verified?

Verification of the design shall be done through:

- Teaming up with blockchain engineer to program functionalities to test feasibilities
- Regular consults with lead developers and architects of the Baseline Protocol for the design iterations
- Evaluation of compliance with the defined User Stories from the ABCDE method
- Evaluation of compliance with the system-of-system architectural principles according the system-of-systems theory

4. How will the design be validated?

Through a case study where the performance of current state manual coordination and contracting on process KPIs is compared to the performance of the new designed automated scenario.

Chapter 3: Design of the System-of-Systems

This chapter describes the development process of the system-of-systems design and the individual system designs. The chapter is subdivided in paragraphs according the design approach defined in section §2.2.2. First the s-o-s goal, actors and user stories are defined. Then the system designs for the Blockchain, Demand, and Supplier System are presented, followed by the final s-o-s architecture and Workflow design.

3.1. Define the goal of the system-of-systems

The first step in the ABCDE method is to provide a short definition for the goal of the s-o-s, and visualise it in the workplace. The goal is to address the problem as defined in section §1.2, which states a lack of a secure system of connected systems for the automated matching and securing of maintenance demand with supply.

More specifically, in section §1.2 was revealed that the input for maintenance sourcing is the automatically generated maintenance schedule. The maintenance schedule is a prioritized list of scheduled maintenance operations that states for each maintenance operation which offshore wind turbines are involved, what type of maintenance they require, an estimated lead time for the operation, and a time window for predicted optimal ocean and weather conditions. The schedule is generated by the automated scheduling module of the asset management system, but could also be determined manually by the asset manager. Section §1.2 also revealed that, once the maintenance supply was secured, automated routing optimisers calculate the optimal routing for the vessels travelling to and within the WPP. This marks the end of the maintenance operation planning procedure, whereafter the physical preparation and execution of the maintenance operation follow. Additionally, since this system-of-systems is designed to be used in a real-world enterprise setting, it should provide asset managers the ability to include certain selection preferences when automatically sourcing for maintenance supply.

Taking the above into consideration, the goal of the system-of-systems is defined as:

Automatically match, contract and coordinate the right maintenance resources for each prioritized maintenance operation as presented in the maintenance schedule, with respect to the given time window and preferences of the asset manager, to ensure maintenance operation feasibility.

3.2 Identify the actors

The task here is to identify all the actors that will interact with the s-o-s. Actors could be human roles, and external systems or devices that exchange information with the system. The envisioned system should be able to interact with each connected supplier in the maintenance supply chain, represented by their operational planner through their ERP system. The Baseline Protocol shall be used to enable this interaction, with the Ethereum blockchain as common frame of reference between the connected systems.

3.2.1 Human Roles

The identified human roles at the WPP that will be interacting with the system are:

- **WPP asset integrity manager**
  The asset integrity manager is responsible for planning, budgeting and forecasting costs associated with the maintenance and service of the turbine and balance of plant. They work closely with the operations and maintenance manager [18]. They are the main decision makers with regard to insourcing, outsourcing and resource allocation.
- **WPP site supervisor**
  The site supervisor manages the day-to-day activities of a team of wind turbine technicians.

They ensure work is completed in line with health and safety regulations, prepare daily reports and plan annual service and maintenance schedules [18]. They manage the wind turbine technician allocation for the wind farm, whether employed by the turbine OEM, wind farm itself, or a third-party service provider.

- **WPP warehouse manager**
  The warehouse manager plans daily operations of the warehouse, interfaces with clients and ensures safety procedures are followed [18]. They manage the wind farm spare part inventory, containing frequently used items. Expensive and low-frequently spares are directly purchased and supplied by wind turbine OEMs.
- **WPP marine coordinator**
  The coordinator manages the movement of personnel and the operations of vessels. This includes checking and tracking vessel certifications, monitoring weather, emergency response planning and liaising with interested parties, such as Maritime and Coastguard Agency [18].

The equivalent of the above roles also exists outside of the WPP, at locations that manage a shared WPP inventory, at OEMS, and at third party service providers. Technicians could also be supplied by turbine OEMs under a service agreement, or by third party providers, or from a shared WPP crew. Spare-parts are usually supplied by turbine OEMs during the typically 5-year warranty period, whereafter they are sourced from own or shared inventory or third-party supplier. The case for vessels is that smaller vessels (CTVs) are usually owned by the wind farm or shared across multiple wind farms. Depending on the configuration of the site and the distance to shore, some wind farms also own a Service Operation Vessels. Additional to the maintenance resources above, offshore maintenance operations also require a dock or berth to load and offload the vessels that support the MO. Overall, the offshore wind maintenance supply chain is organised in a variety of configurations, mostly depending on WPP size, distance to shore and proximity to other WPPs.

The envisioned s-o-s should be generic and applicable to all sorts of WPP maintenance supply chain configurations; therefore, an overview is created of all the potentially involved participants and their human representative.

Side	Generalised supply chain role	Human role	Source of supply	Range of instances
Demand	Asset owner	Asset manager	Offshore wind power park	One multiple
Supply	Technician supplier	Planner	WPP own technician team	One or multiple
			WPP shared technician team	
			OEM	
			Third-party technician supplier	
	Spare-part supplier	Planner	WPP own inventory	One or many
			WPP shared inventory	
			OEM inventory	
			Third-party spare-part supplier	
	Vessel supplier	Planner	WPP own fleet	One or many
			WPP shared fleet	
			Third-party vessel supplier	
	Port	Planner	WPP own dock	One or a few
			WPP shared dock	
			Third-party dock provider	

Table 6: Overview of maintenance supply chain participants and the human roles that interact with the s-o-s.

3.2.2 External systems

The external systems interacting with the envisioned system-of-systems are identified in this section. The first interaction is with the system that outputs the maintenance schedule, the automated maintenance scheduler that is part of the WPP asset management system. As mentioned in section §3.1, it should also be possible to upload maintenance schedules created otherwise, given formatted correctly. Second, maintenance operations in the schedule should be matched with detailed lists of required resources for the operation. These lists could be found in a maintenance database, asset management system or ERP system. Third, each required resource should be matched with potential capable suppliers found in the WPP's ERP system. Fourth, for every listed potential supplier, the envisioned s-o-s needs to collect availability and cost information. This information usually resides in secure ERP systems of the suppliers. Large enterprises have the financial resources to operate on expensive ERP systems, such as SAP or Microsoft Dynamics. Companies with less financial resources such as SMEs, might manage their asset planning on a simple spreadsheet such as Google Sheets or Microsoft Excel. Fifth, all these systems are supposed to interact with each other through the envisioned system-of-systems, enabled by the Baseline Protocol technology stack and the Ethereum blockchain system.

3.2.3 Actor overview

To conclude on the sections above, an overview of the identified and generalised actors is pictured below. In terms of human roles, the s-o-s interacts with one or multiple asset managers and with one or multiple supplier's planners. In terms of external systems, the s-o-s interacts with one or multiple asset management systems, one or multiple demand side ERP systems, one or multiple supply side ERP systems and a single blockchain system that facilitates the process automation.

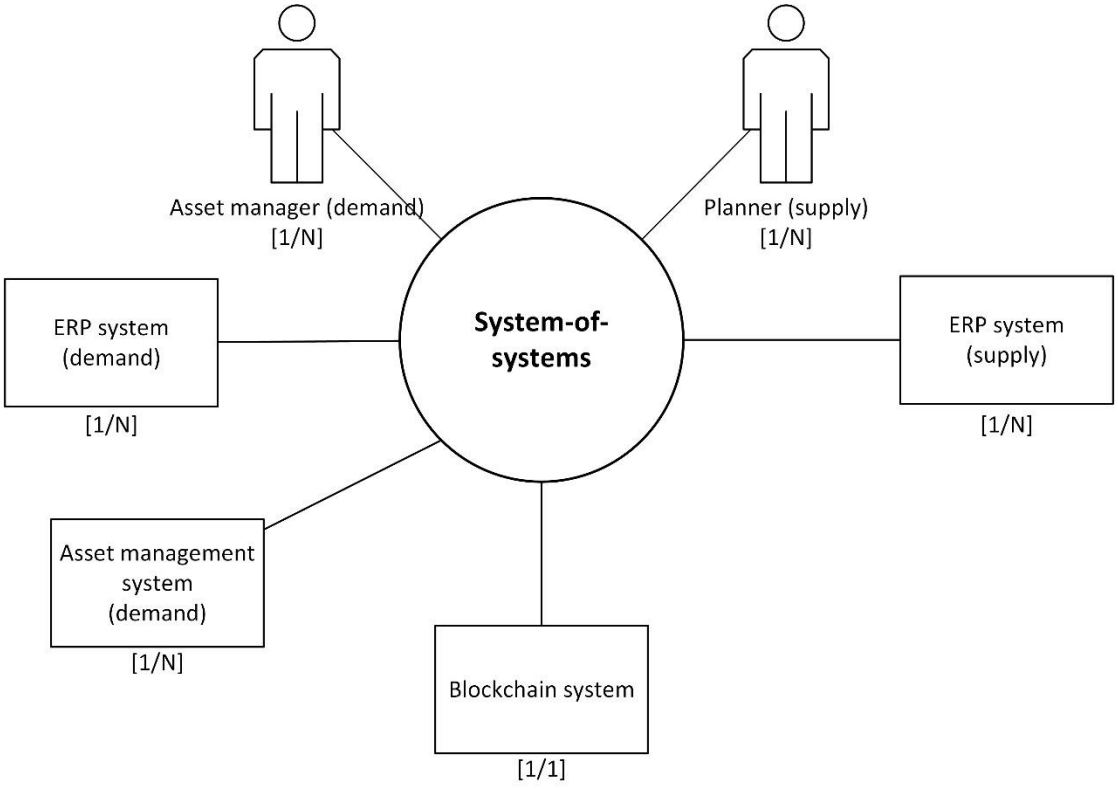


Figure 27: Identified system-of-systems actor overview, subdivided in human roles and external systems

3.3 Define initial system-of-systems architecture on high-level

This design step is incorporated from the Baseline Protocol and its design method. According the method, the first thing to do is defining is the Workgroup. The Workgroup is the group of supply chain participants that want to baseline their systems for the execution of the multilateral Workflow. Different from the previous step, the Workgroup is defined on enterprise level instead of individual actor level, from the perspective of interaction with the shared process instead of interaction with the system-of-systems.
For the merged design methodology applied in this research, the Workgroup definition step is extended a little to already reveal some structure between the supply chain participants, their operating systems and the external systems. Although on a high, abstract level this initial system-of-systems architecture is useful to keep in mind for development of the following design steps.

What we know so far, is that our system-of-systems includes three main systems; a demand side system, a supply side system, and a blockchain system. The blockchain system is supposed to be the binding system that enforces collaboration between all sorts of configurations of connected demand and supplier systems. The demand system is supposed to interact with the asset management system and the ERP system as defined in section §3.2.3. The supplier system is supposed to interact with the ERP system on the supply side. The ERP connector in the Baseline Protocol elements allow for operation through ERP system, so both the asset manager and the supplier's planner shall interact with our system-of-systems through their ERP systems.

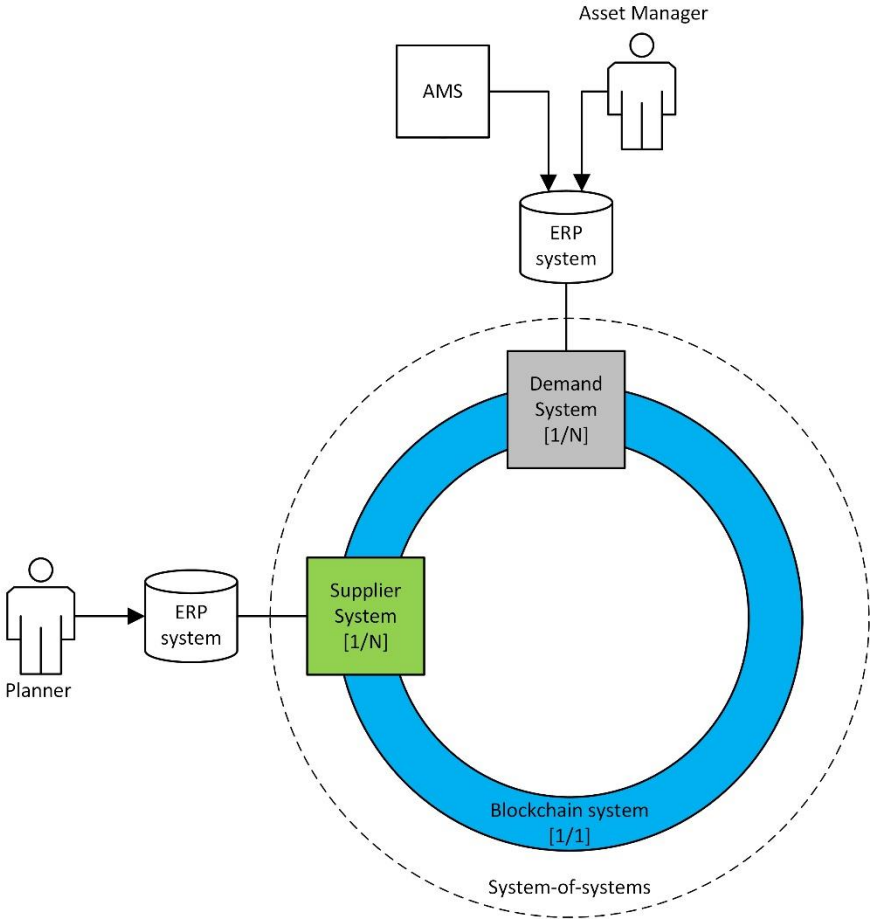


Figure 28: Initial s-o-s architecture design that supports all varieties of supply chain configuration

The initial s-o-s architecture design is illustrated in figure 28. The to be designed system-of-systems is captured within the dotted line. Within the s-o-s three separate systems can be distinguished; the grey demand system, the green supplier system and the blue blockchain system. As we can see the ERP systems here are included in the defined demand and supplier systems as they function a vital role in the interaction between the human roles and the system-of-systems. This also complies with the property of operational and managerial independence, according the system-of-systems theory as described in section §2.1.1.2.

3.4 Define user stories for the system-of-systems

The next design step in our chosen method is to define user stories for the human roles interacting with the s-o-s. The stories are written according the guidelines of M. Cohn [19], which describe the following properties for good user stories:

- Independent: limit dependencies between stories.
- Negotiable: stories are short descriptions of functionality, for which the details have to be negotiated between development team and customer.
- Valuable to users or purchasers: each story must be valued by either the user or the purchaser of the system.
- Estimable: able to estimate the amount of time it takes to develop a story.
- Small: stories should be comprehensible and plannable
- Testable: stories must be written so as to be testable in order to prove a successful development.

Although there are no end users involved in this research, the user stories shall reflect common business practices regarding matching, sourcing and contracting of suppliers for multilateral operations. The working experience of the researcher, myself, is valuable for defining the user stories. I worked as a project manager for the construction and installation of large LED displays that were used for advertisement purposes. These displays were installed on buildings, in sport facilities, next to highways, and on Ferris wheels. For each installation my task was to request supplier availabilities and plan the multilateral operation involving parties such as; internal and external technicians, the production facility, traffic control, crane operators, ground workers, municipalities, customers. Although the context is different, on a fundamental level the process steps are similar.

Additionally, as the s-o-s is highly automated, most user stories are supposed to be performed by the envisioned s-o-s instead of actual users. The user stores are defined from the perspective of the two identified human roles in section §3.2.1, the asset manager and the planner. Automated user stories shall be described as to be executed by the s-o-s, but always from a user's perspective.
On top of the user stories, also constraint stories shall be defined. Constraint stories are performance related, instead of function related, and state certain performance or security requirements from a logic business practice perspective.

3.4.1 User Stories for the demand side Asset Manager

The following user stories are constructed for the asset manager. The first step is the ability to provide the system with a maintenance schedule. This maintenance schedule should be machine-readable; therefore, it shall be digital and have a uniform structure. User stories attributed to the "s-o-s" should be read as: "The Asset Manager wants the s-o-s to..".

Story ID:		**User stories for demand side Asset Manager**
US1	Story:	Asset Manager or Asset Management System can submit a machine-readable maintenance schedule of a specific predefined extension
	Notes	
US2	Story	S-o-s stores the high prioritized, unplanned maintenance operations listed on the maintenance schedule for further processing
	Notes	How MOs are prioritized depends on AMS or Asset Manager
US3	Story	S-o-s can match potential suppliers from ERP system to fulfil the execution of the stored maintenance operations
	Notes	
US4	Story	S-o-s requests availability and cost information from the Supplier System for each matched potential supplier for the MO
	Notes	
US5	Story	S-o-s proposes the optimal subset of matched available suppliers according the set preferences of the Asset Manager
	Notes	
US6	Story	S-o-s presents prefilled templated business proposals for each supplier in the chosen supplier subset, to enable the Asset Manager to review and edit certain terms of the proposal
	Notes	
US7	Story	All users can sign, edit and send business proposals to relevant connected parties via the s-o-s
	Notes	
US8	Story	S-o-s must notify Asset Manager of state and outcome of pending business proposals and update maintenance schedule accordingly
	Notes:	
US9	Story	Asset Manager can start and stop the system at any time and continue supplier sourcing for maintenance operations the traditional manual way
	Notes	
CS1	Story:	S-o-s should be able to support supply chains in order of magnitude of 100 members
	Notes	
CS2	Story:	The S-o-s setup and integration of new members should take maximum 8 manhours of work
	Notes	As a measure of acceptable cost and time
CS3	Story:	System should reduce the amount of manual labour performed on the sourcing process by at least 80%
	Notes	
CS4	Story:	System should be easy to use and understandable for the average office worker
	Notes	

Table 7: Defined user stories for the s-o-s from the perspective of the demand side Asset Manager

3.4.2 User Stories for the supply side Planner

The following user stories are constructed from the perspective of the Planner at the supply side. Most of the functionalities are already captured on the demand side, where the demand side wants the supply side to be enabled to reach the common goal of matching maintenance demand with supply. The stories defined from the perspective of the Planner are more related to security and protecting against unnecessary data mining.

Story ID:		**User stories for supply side Planner**
US10	Story:	S-o-s is able to verify if a cost and availability data request from the Demand System is valid and prevent unnecessary data harvesting
	Notes	
US11	Story:	Planner can disconnect from the automated information sharing Workflow at any time he pleases, while still be able to continue to business in the traditional manual way
	Notes	
US12	Story:	S-o-s can present supplier incoming business proposals to enable review, editing, signing, declining, and sending of handled proposal
	Notes	Story is large because of overlap with US6
CS5	Story:	S-o-s must equal the level of security of typical business IT applications
	Notes	

Table 8: Additional defined user stories for the s-o-s, from the perspective of the supply side Planner

3.5 Define initial system-of-systems Workflow on high-level

Through the combination of the goal of the s-o-s, the initial s-o-s architecture, and the user stories, the initial s-o-s Workflow can be designed. The Workflow is the automated multilateral business process executed with each connected system in order to reach the defined goal; the matching and contracting of maintenance demand with supply. Due to the decentral character of our s-o-s, the Workflow operates on the element level of the individual systems that make up the s-o-s. Returning to the pyramidic illustration of the s-o-s in figure 29, the individual systems are now coloured according the colour scheme used in section §3.3 while the Workflow in coloured white.

The swimlane diagram in figure 30 on the next page shows the design for the initial Workflow. Each activity is marked with a number, and the manually executed activities are additionally indicated with a handshaped cursor. The coverage of the user stories by the Workflow is shown in table 9.

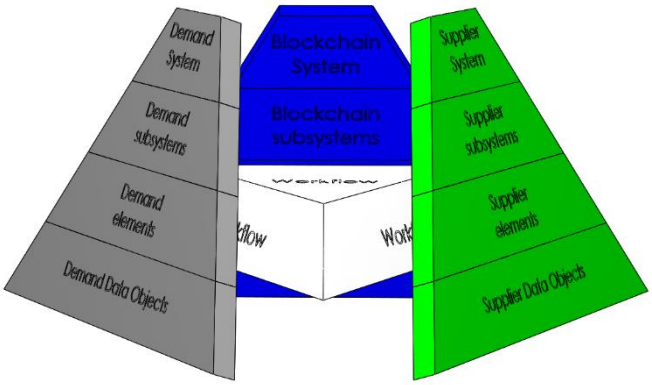


Figure 29: Graphic illustration of s-o-s pyramid in colour, including the Demand (grey), Supplier (green) and Blockchain (blue) system and the s-o-s Workflow (white)

User story	Activity
US1	1
US2	2
US3	3
US4	4,6,7,8
US5	9,10
US6	11,12
US7	13,15
US8	N.a.
US9	N.a.
US10	5
US12	15
CS1, CS2, CS3, CS4, CS5	N.a.

Table 9: User story coverage by initial Workflow

The initial s-o-s Workflow design starts with the intake of the maintenance schedule by the Demand System. After the system matches potential suppliers to each MO requirement, availability and cost information is requested from each matched Supplier System. The Blockchain System shall validate and register the request on the public network, whereafter the Supplier system is allowed to calculate and return the supplier availability and cost information for the given time window. From the returned availabilites, the Demand system calculates overlapping supplier availability to be able to meet all MO requirements. Once the MO is determined to be feasible, the Demand System selects and proposes the optimal supplier subset to the Asset Manager. After confirmation, the Demand System fills templated business proposals and allows the Asset Manager to review, edit, sign and send the proposals. The Blockchain System validates and registers the proposals before the Supplier System notifies the Planner. The Supplier System allows the Planner to review, edit, sign or decline the proposal. In case the Planner confirms the proposal by setting his signature, the business proposal technically become a double signed business agreement. The Blockchain System again validates and registers the business agreement, while the agreement is returned to the Demand System. Due to the fact that availability and cost information is known at setup of the business proposals, the chances of supplier confirmation are much higher. In case a supplier declines the offer, the Demand System picks another available supplier from earlier calculations and restarts the contracting process.

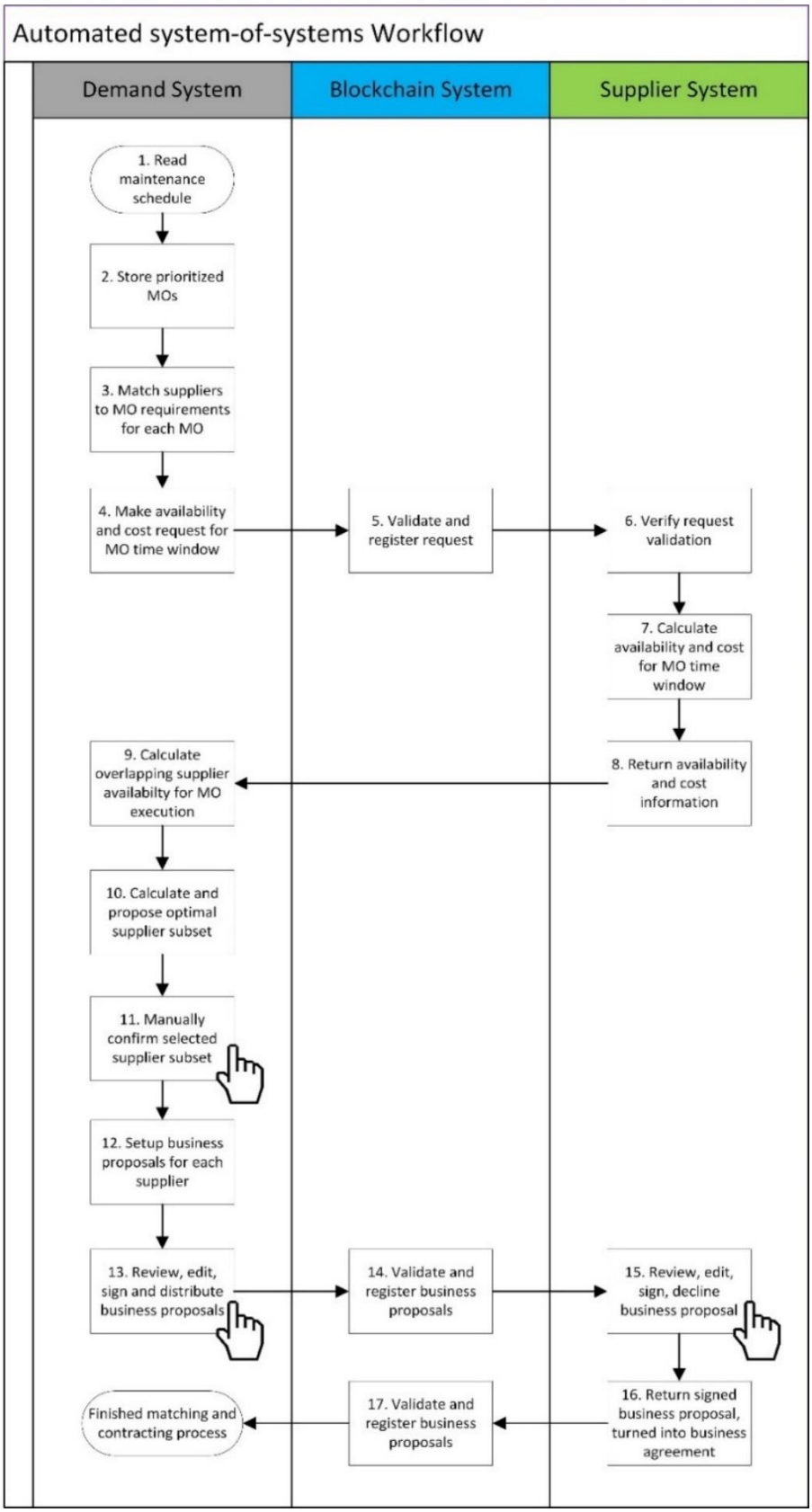


Figure 30: Initial design of automated system-of-systems Workflow, on high-level. Manual interaction is indicated with the hand cursor. Note that multiple Demand and Supplier systems could be involved in the system-of-systems.

3.6 Design the Blockchain System

This section describes the development of the design for the Blockchain System within the system-of-systems. According the ABCDE design method, it is the first system that has to be designed. The design work on the Blockchain System is rather limited because the infrastructure is already covered by the Ethereum public network. While most of the elements, i.e. the smart contract, and the data objects are already defined and partly designed in the Baseline Protocol. The interaction between the smart contract is also defined by the Baseline Protocol, therefore the only pieces that have to be designed in the Verifier smart contract and the Commits that are released and broadcasted after a successful verification. The table below gives an overview of the most essential items on each detail level of the system, and if an item is in the design scope.

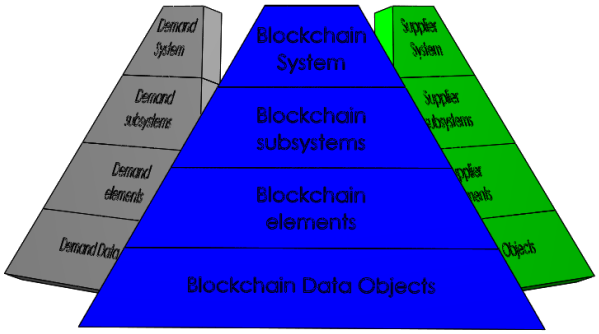


Figure 31: Blockchain System theoretical framework for reference

Level of detail	Items	Instance	Design scope	Part of
Blockchain subsystems	Network	Internet protocol	No	Internet
	Nodes	Networked hardware (i.e. computers)	No	Internet
	Virtual machine	Ethereum Virtual Machine	No	Ethereum
	Ledger	Ethereum blockchain	No	Ethereum
	Consensus mechanism	Proof of Work	No	Ethereum
Blockchain elements	Transactions	Regular (value) transactions	No	Ethereum
		Contract deployment transactions	No	Ethereum
	Smart contracts	OrgRegistry	No	Baseline Protocol
		Shield	No	Baseline Protocol
		Verifier	Yes	Baseline Protocol
	Data structures	Merkle Tree (stores Commits)	No	Baseline Protocol
Blockchain data objects	Commits	To be defined	Yes	Baseline Protocol
	Addresses	Wallet addresses	No	Ethereum
		Smart contract addresses	No	Ethereum

Table 10: Overview of most essential items in Blockchain System, subdivided on detail level

Because the Blockchain System design incorporates the Ethereum blockchain and Baseline Protocol items and interactions, the system architecture and the interaction within the Blockchain System are already defined. The system architecture together with the interactions is pictured on the next page in figure 32. The Blockchain System is a network of nodes, i.e. Demand and Supplier systems, taking part in the decentralised Ethereum public network amongst other unrelated nodes. Within that network three smart contracts are deployed; the OrgRegistry, the Shield and the Verifier. Section §2.1.2.2.1 elaborated on these smart contracts, therefore only the interactions within the Blockchain System shall be described here.

1. A node (e.g. the Demand System) generates a cryptographically secure proof of an executed Workstep in the Workflow. Additionally, the node prepares a hash (Commit) of the valuable information associated to that Workstep. The node sends both proof and Commit to the Shield contract.
2. The Shield contract authenticates the sender by searching for his address in the list of connected participants in the OrgRegistry contract. If the address is found, the Shield forwards the received proof to the Verifier contract.

3. The Verifier contract receives the proof of the Shield contract and calculates if the proof is correct and in compliance with the cryptographically transformed business logic stated in the Verifier contract.
4. When the proof is correct, the Verifier contract emits a successful event, which is intercepted by the Shield contract.
5. The Shield contract releases the Commit, earlier received by the sender node, and updates his storage of Commits (i.e. the MerkleTree). The nodes that are configured to listen to the Shield contract (nodes listed in the OrgRegistry) intercept the Commit and store it in their local MerkleTree data storage. This way, each connected system in the network is automatically and continuously updated with Commits.

A verified event is proof of a verifiably executed Workstep, and the broadcasted Commit is the hash of the original relevant data related to that Workstep. Which means that the systems in possession of the original data can easily hash that data and compare it to the Commit broadcasted in the network. The connected Demand and Supplier systems listed to the Shield contract and update their local MerkleTree of Commits accordingly. This way the system-of-systems automatically synchronizes the state of the Workflow with each system while preserving data privacy on a public network, and only the interacting parties in possession of the original data can verify if the broadcasted Commit is a correct representation of the original data.

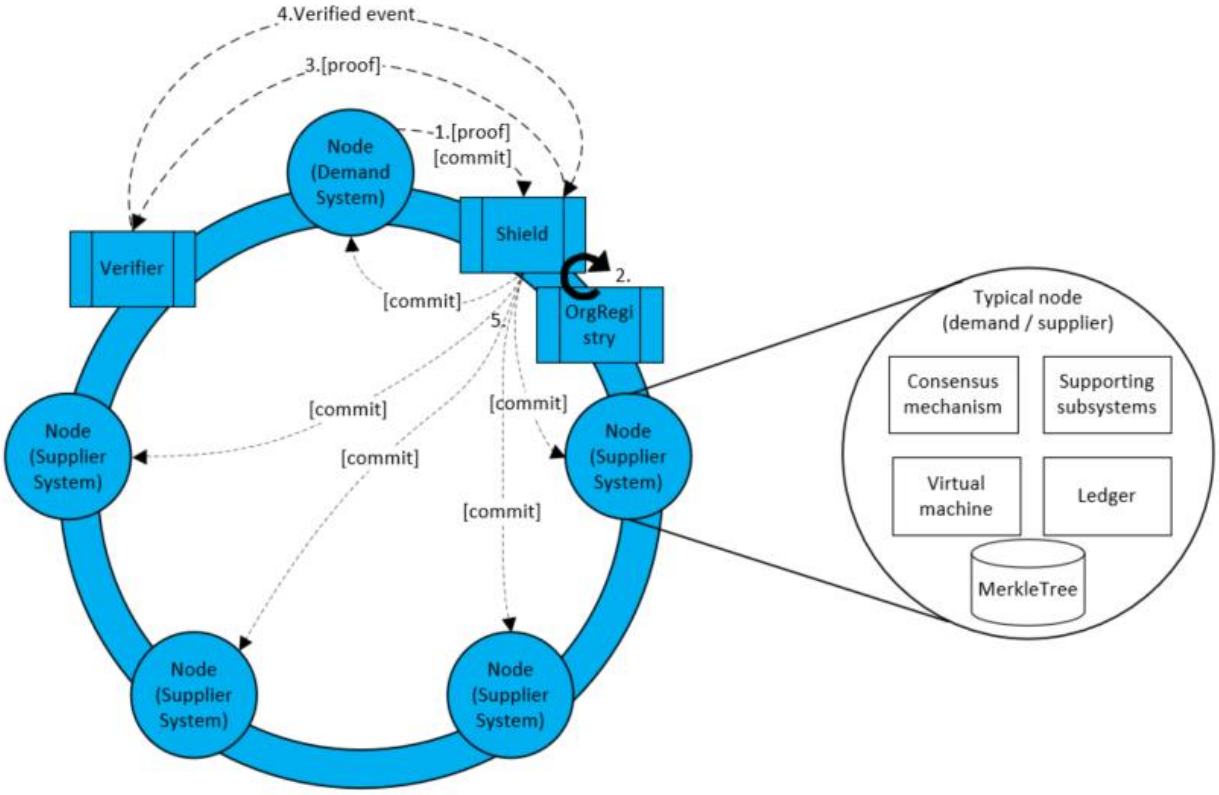


Figure 32: Graphic representation of system architecture for Blockchain System, with zoomed in node and displayed interactions between the subsystems and elements

The Blockchain System design, of which most design effort went in to the Verifier contract, went through five design iterations, which are summarized below and explained in detail in Appendix A.

Design Iteration	Summary
1	Due to the limitations of the Zokrates toolbox, the Verifier is not able to calculate variable outputs and handle multi-party inputs, therefore the idea of calculating multi-party availability date ranges on submitted availability schedules became infeasible. Concluded was not to calculate supplier availability with the Verifier contract.
2	The idea of having multiple Verifier contracts dedicated specifically to each important Workstep was discarded due to high operational costs and due to highly complex routing of data flows between all systems and their elements. Concluded was to use one Verifier contract dedicated to the Workflow for each individual maintenance operation
3	Having one Verifier contract deployed for each maintenance operation was still deemed complex in data routing and also leads to high operational costs because of expensive contract deployment costs, therefore was concluded to use one generic reusable Verifier contract that covers all Workflows for demand and supply matching for all possible maintenance operations
4	The initially determined hashing algorithm used for the multi-stage data hashing scheme in the Verifier contract was the widely used SHA256, due to the high cost of this hashing operation on the Ethereum blockchain, the long processing times and the enormous amount of outputted lines of code for the Verifier contract, it was concluded to use the far more efficient Pedersen hashing algorithm[13].
5	To prevent unlimited data mining from the Demand Systems, a Genesis Commit was added to the initial set of Commits. Letting the Verifier contract publish any kind of Commit costs transaction and calculation fees. The Genesis Commit is added to mark the start of the automated Workflow, meaning that the Demand System first has to make expenses before it can request any data from the Supplier Systems. Additionally, a Selection Commit was added to mark the end of the supplier matching and selection process. Via the Selection Commit Supplier Systems are notified about their selection and already can block their availability schedules before entering the contracting part of the Workflow to decrease the chance of sudden unavailability.

Table 11: Overview of design iterations related to the Blockchain System

[13] https://zokrates.github.io/toolbox/stdlib.html

3.6.1 Review actors and user stories

Zooming in on the Blockchain System, the only actors it has to serve are the connected Demand- and Supplier systems. The Demand Systems are chosen to be the main system interacting with the Shield contract, because they are in need of maintenance so it's only fair they pay the cost for using the blockchain infrastructure. Only when cancelling an ongoing matching and contracting process, a Supplier System has to communicate with the Blockchain System. Due to the public nature of the network, the Shield contract possibly has to deal with unsolicited or malicious interaction with an external node. This security issue is covered by the way the interaction between the Shield, OrgRegistry and Verifier smart contract is arranged in the Baseline Protocol. The reviewed user stories below are specifically focussed on the Verifier smart contract and the Commit data objects, because those are the only items that need to be designed. The user stories are setup in consideration of the best practices, limitations and requirements of the Zokrates toolbox[14], which is a toolbox for Ethereum smart contracts that use zk-SNARKs for privacy preserving data processing.

User stories specifically for the Verifier contract

1. Able to verify various types of Workstep execution proofs in one permanently reusable Verifier smart contract
2. Able to protect against unnecessary data mining of Demand Systems (in line with US10)

3.6.2 Elements design

Verifier smart contract

A crucial piece of the Blockchain System design is the element Verifier smart contract. The function of the Verifier is to act as an automated trusted third party that verifies data that proofs the execution of a Workstep in the Workflow. Besides the manual confirmation and signing actions, verified events emitted by the Verifier contract trigger subsequent Worksteps at connected systems in our s-o-s, thus automate collaboration between counterparty systems that don't necessarily trust each other. It is important to note that the design for the smart contract is actually design of the arithmetic circuit that is able to verify business logic of a Workstep. This arithmetic circuit is transformed cryptographically in to a data privacy preserving zkSNARK verification scheme that will be deployed as a Verifier smart contract on the public Ethereum network.

Taking into consideration of the above design iterations, the idea for the Verifier contract design is to verify proof of correct formulation of the data related to a certain Workstep. The data related to this proof is hashed in a multi-stage hashing operation that outputs one single hash; the Commit hash which is emitted after a successful verification by the Verifier. First of all, participants have to be in possession of the correct formulation scheme in order to be able to regenerate emitted Commits that end up in each local MerkleTree storage of each connected system. This is a secure method of Workflow state synchronization between connected systems on a public network. Every node in the public network is able to intercept the emitted Commits, but only the supply chain participants are able to regenerate those Commits with the formulation scheme in order to synchronize their systems and verify compliance to the business logic. However, within our supply chain sensitive bilateral business data related to a Workstep also has to remain private for the two involved parties, as they don't want other supply chain participants to have full insight into their business agreements. This protection is provided by hashing commercially sensitive inputs (or documents) in the formulation scheme, so only the parties in possession of the original data can verify the emitted Commits.

[14] https://zokrates.github.io/

Table 12 gives an overview of which parties can access and learn from the emitted Commits on the public network.

Data on public network	Data accessibility	Data insight	Reasoning
Emitted Commit hash	Whole public network	None	Impossible to decrypt formulation scheme of data and hashes
	Supply chain participants in public network	Workflow state and MO ID	Unsensitive supply chain data and hashed sensitive data used as input for the formulation scheme, to rebuild and verify the Commit hash
	Two supply chain participants in public network involved in bilateral Workstep	All data; Workflow state, MO ID, involved suppliers, business proposals, business agreements	Unsensitive data and sensitive data used as input for formulation scheme, to rebuild and verify Commit hash

Table 12: Data accessibility and insight for emitted Commit hashes on public Ethereum network

The following inputs are defined for the hash formulation scheme. Details on the formulation scheme used for the arithmetic circuit that serves as base for the Verifier smart contract are provided in Appendix B.

Input name	Description	Data relevance and insight	Format
State	One of the 5 states of the Workflow	Supply chain	Non-negative integer value
MJ ID	Maintenance operation ID	Supply chain	Non-negative integer value
SupplierID	Involved supplier in bilateral Workstep	Involved maintenance demand party and supplier	Non-negative integer value
DocHash1, DocHash2	Two parts of the Pedersen hash of the Business proposal document from the Demand System to the Supplier System	Involved maintenance demand party and supplier	Non-negative integer value
ContractHash1, ContractHash2	Two parts of the Pedersen hash of the Business agreement document between the Demand System and the Supplier	Involved maintenance demand party and supplier	Non-negative integer value
LC1, LC2	Two parts of the Commit hash last added to the MerkleTree locally stored in each connected system	Supply chain	Non-negative integer value
NC1, NC2	Two parts of the newly generated Commit hash to be added to the local MerkleTree (the output of the formulation scheme) in each connected system after successful verification.	Depending on the Workstep, either supply chain or involved maintenance demand party and supplier	Non-negative integer value

Table 13: Inputs for the hash formulation scheme used for the Verifier smart contract

3.6.3 Data objects design

Commits

Now that the design for the Verifier contract is finished, the emitted Commits can be designed. The Commits are emitted after each successfully verified Workstep to synchronise the connected systems, and function as a fingerprint of the data associated to that particular Workstep. Commits could be seen as process state markers for the demand and supply matching Workflow for a particular maintenance operation. The five defined Commits are:

Commit name	Workflow State	Description	Function
Genesis Commit	1	Marks the start of the maintenance demand and supply matching Workflow for a particular MO.	Automated Workflow start notification and monetized protection against free data mining of Demand System, covers US10.
Selection Commit	2	Marks the end of calculating overlapping supplier availability and supplier selection for the MO, and marks the start of supplier contracting phase	Automated notification that available suppliers are selected for the MO, incentive for them to block their availability schedules for the determined MO time window.
Business proposal Commit	3	Marks a single business proposal successfully proposed to a selected supplier	Registered proof of business proposal from demand party to supply party
Business agreement Commit	4	Marks a single business agreement successfully	Registered proof of bilateral business agreement between demand and supply party
Finalization / cancelation Commit	5	Marks the end of a matching and contracting Workflow for an MO, either due to successful match of maintenance demand and supply, or due to cancelation of one of the suppliers	Automated notification of successfully executed Workflow, whereafter preparation for the MO starts. Or, automated notification of cancelation of Workflow, which restarts a new matching and contracting Workflow for the particular MO.

Table 14: Overview of the designed data objects

3.7 Design the Demand and Supplier System

This section describes the designs and the development process for the Demand and Supplier Systems synchronised by the Blockchain System, according the ABCDE design method. Both system designs should adhere to the system-of-systems requirements and design principles as stated in section §2.1.1. The Demand and Supplier systems share many similar subsystems, elements and data objects in order to be synchronised through the Blockchain System, except for a few elements and data objects specifically designed to enable the user stories defined for each. Both systems include the existing ERP system where they are operated through, and elements from the Baseline Protocol.

3.7.1 Design of the Demand System

The Demand System is the initiator for the automated maintenance demand supply matching and contracting Workflow, executed for each prioritized maintenance operation presented in the maintenance schedule. Within the boundaries of the WPP digital infrastructure, the system includes the demand side ERP system, communicates with the Asset Management System and is operated by the Asset Manager. In the system-of-systems context it communicates with the Blockchain System and the connected Supplier Systems. Similar to the design development for the Blockchain System, let's start with an inventory overview of relevant items for the Demand system.

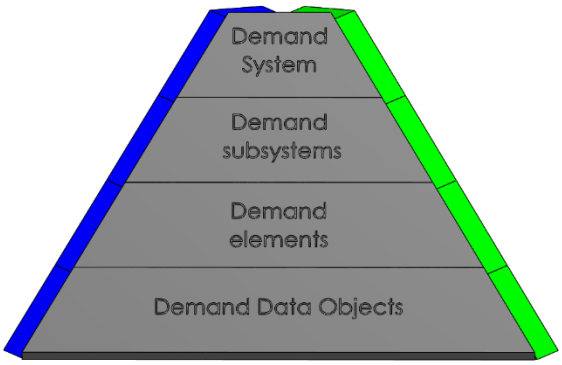


Figure 33: Demand System theoretical framework for reference

Level of detail	Items	Instance	Design scope	Part of
Demand System subsystems	ERP system	SAP, Microsoft D365, Excel	No	Digital infrastructure asset owner
	To be defined	To be defined	Yes	Design research
Demand System elements	Messenger service	NATS	No	Baseline Protocol
	Zeroknowledge service	Zokrates toolbox	No	Baseline Protocol
	ERP connector	SAP connector, D365 connector	No	Baseline Protocol
	Blockchain client	Geth	No	Baseline Protocol
	Vault service	Provide Vault	No	Baseline Protocol
	Blockchain database	MerkleTree (local)	No	Baseline Protocol
	To be designed	To be designed	Yes	Design research
Demand System data objects	Maintenance schedule	Machine-readable offshore wind turbine maintenance schedule	No	Asset management system
	To be designed	To be designed	Yes	Design research

Table 15: Overview of most relevant items in Demand System, subdivided on detail level

The Asset Manager interacts with the Demand System through the existing ERP system via the ERP connector provided by the Baseline Protocol. The Demand System also has to fetch a lot of supplier information from the ERP system to match them to MO requirements. The Demand System interacts with the Supplier Systems directly via the Messenger service, and indirectly via the Blockchain System through the blockchain client, both provided by the Baseline Protocol. The Zeroknowlegde service is used to create the Verifier contract and the cryptographically secure and abstract Zeroknowledge proofs (ZKPs) of the data related to important Worksteps in the Workflow, that are being send to the Verifier contract. Through the Vault service an Asset Manager is able to digitally

sign and verify signatures on business proposals and business agreements. The remaining elements and data objects in support of the goal of the system-of-systems and the user stories have to be designed.

An overview of the design iterations the Demand System went through is shown in the table below. Detailed descriptions are provided in Appendix A.

Design Iteration	Summary
6	To limit communication with the database, and to make communication between system elements and external systems more efficient; the choice was made to include a JSON data object "MO object" to store all the MO related data.
7	A crucial functionality for the connected systems is to be able to regenerate Commits based on the MO data a system received through the Messenger service, and to be able to compare the regenerated Commit with the Commit emitted by the Shield contract that was stored in the local MerkleTree. This is how each connected system automatically verifies the data used for Workflow synchronization. Therefore, a new system element had to be designed: Commit Generator
8	The data used in in- and outbound communication via the Messenger service required a lot of formatting. Instead of doing all the formatting at the respective element, decided was to add a new system element "Data Formatter"

Table 16: Overview of the design iterations related to the Demand System

3.7.1.1 Review actors and user stories

Zooming in on the Demand System, the actors it has to serve are the Asset Manager, the Blockchain System and the Suppliers System. Although the user stories are defined for the s-o-s, the Asset Manager operates on the Demand System, therefore all the defined user stories US1 – US9 in table 7, for the Asset Manager have to be covered, for the most part, by the Demand System.

3.7.1.2 Elements design

MO Extraction Module

US1 and US2 address the ability to automatically open and read maintenance operation schedules from a certain location, and select and store prioritized MOs for further processing. To enable this ability an element called "MO Extraction Module" was designed. The assumption is that either the Asset Manager or the AMS generate an MO schedule and store it at a certain location in the ERP system. For each MO schedule stored in the ERP system, the MO Extraction Module extracts relevant MO data and store it into an MO object dedicated to each prioritized individual MO. The most important parameters to store are:

- MO ID
- Estimated MO lead time
- Optimal MO time window
- MO requirements, either obtained via an MO code in the AMS or included in the MO schedule

Additionally, the MO Extraction module was designed to enable US3, the matching of potential suppliers to MO requirements found in an MO object. For each MO requirement, the Matching Module searches the ERP system for suppliers capable of providing the MO requirement, for example the need for a heavy lifting vessel, and lists those suppliers in the MO object.

Program details of the MO Extraction Module is found in Appendix C.

Commit Generator

The next step, US4, would be to request the availability of the potential suppliers for the determined

MO time window. The design of the Blockchain System revealed that a Genesis Commit had to be published first, to protect the Supplier Systems from unnecessary data mining from the Demand System. A system element had to be designed to enable the ability of generating Commits in order to pass them to the Shield contract, and to be able to verify a Commit pushed to each MerkleTree. The designed "Commit Generator" has to main functionalities:

- Generate Commits that together with Zeroknowledge Proofs (ZKProofs) are send to the Shield contract in order to synchronize the state of the Workflow with other connected systems. Commits are generated from data in MO objects received from a local database.
- Verify Commits that are newly added to the MerkleTree in the Blockchain database. First, generate a Commit out of the data in the MO object received through the Messenger service. Take the latest Commit addition from the MerkleTree, and compare both Commits. If the Commits are equal, the data received from the Messenger service is trustworthy and exactly matches the data used in the verified Workstep. If the Commits are unequal, there is data asymmetry in the system-of-systems and an exception should be raised.

Program details of the Commit Generator are found in Appendix D.

Workflow Database
To cover US2 and support the other elements, the need for a Workflow database was determined. This database stores the MO objects and other Workflow related data objects and metadata.

Matching Module
According US5, once the Demand System has received above threshold number of Supplier System replies on the availability and cost information request, the Demand System is expected to propose an optimal subset of available suppliers capable of executing the maintenance operation. This seems easily calculated, but it is actually quite a complex computation. An optimal MO time window can span for example 10 days, and an MO lead time can for example be 2 days. Supplier availability can be fragmented within that 10 day time window, and vary for each supplier. A supplier can, for example, be available on day [1, 2, 5, 8,9,10]; therefore, this supplier's availability for the 2 day MO operation is [1,2], [8,9] and [9,10]. The Matching Module is designed to calculate the optimal subset of suppliers according the following logic:

1. Wait until number of Supplier System responses is above threshold. A Supplier System only gives a response to the availability request if they are available for the MO lead time in the MO time window.
2. Out of all the Supplier availabilities, calculate the day of earliest convenience in the MO time window that all MO requirements are met by at least one supplier.
3. Based on that determined day, generate all possible supplier subsets chronologically and calculate the total costs of each subset. Costs are calculated by summing up the product of supplier rate times the MO lead time.
4. Select and propose the supplier subset according the set preferences of the Asset Manager, for example the cheapest subset or the earliest available one.

Program details of the Matching Module are found in Appendix E.

Data Formatter
The need for the Data Formatter originated out of the internal communication with the Messenger service, as design iteration 8 describes. Data for the in- and outbound messages for other connected systems needed to be transformed for further processing in the system itself.

3.7.1.3 Data objects design

MO object

Both US2 and design iteration 6 lead to the design of the data object “MO object”. The MO object is a JSON schema type of data storage object, that is easily passed around and processed by system elements. It contains the following fields of data for each individual maintenance operation:

- MO ID
- State
- Wind Turbine IDs
- MO lead time
- MO time window
- MO code; reference for type of maintenance operation
- MO description
- MO requirements
- Matched potential suppliers
- Calculated day of earliest convenience
- Generated supplier subsets
- Selected supplier subset
- Business proposals per selected supplier
- Business agreements per selected supplier
- Commits related to the MO

3.7.1.4 Demand System – final design

Putting all the defined and designed pieces from the above section together, we arrive at the final design of the Demand System as pictured in figure 34. The figure shows the total Demand System design, the identified subsystems, the designed elements and the data flow between the elements. A complete breakdown of the system is given in the table below.

Level of detail	Items	Instance	Design scope	Part of
Demand System subsystems	ERP system	SAP, Microsoft D365, Excel	No	Digital infrastructure asset owner
	Security subsystem	N.a.	Yes	Design research
	Communication subsystem	N.a.	Yes	Design research
	Workflow Support subsystem	N.a.	Yes	Design research
Demand System elements	Messenger service	NATS	No	Baseline Protocol
	Privacy service	Zokrates toolbox	No	Baseline Protocol
	ERP connector	SAP connector, D365 connector	No	Baseline Protocol
	Blockchain client	Geth	No	Baseline Protocol
	Vault service	Provide Vault	No	Baseline Protocol
	Blockchain database	MerkleTree (local)	No	Baseline Protocol
	MO Extraction module	N.a.	Yes	Design research
	Commit Generator	N.a.	Yes	Design research
	Workflow database	N.a.	Yes	Design research
	Matching Module	N.a.	Yes	Design research
	Data Formatter	N.a.	Yes	Design research
Demand System data objects	Maintenance schedule template	Machine-readable offshore wind turbine maintenance schedule	No	Asset management system
	MO object	N.a.	Yes	Design research

Table 17: Demand System - final design system breakdown

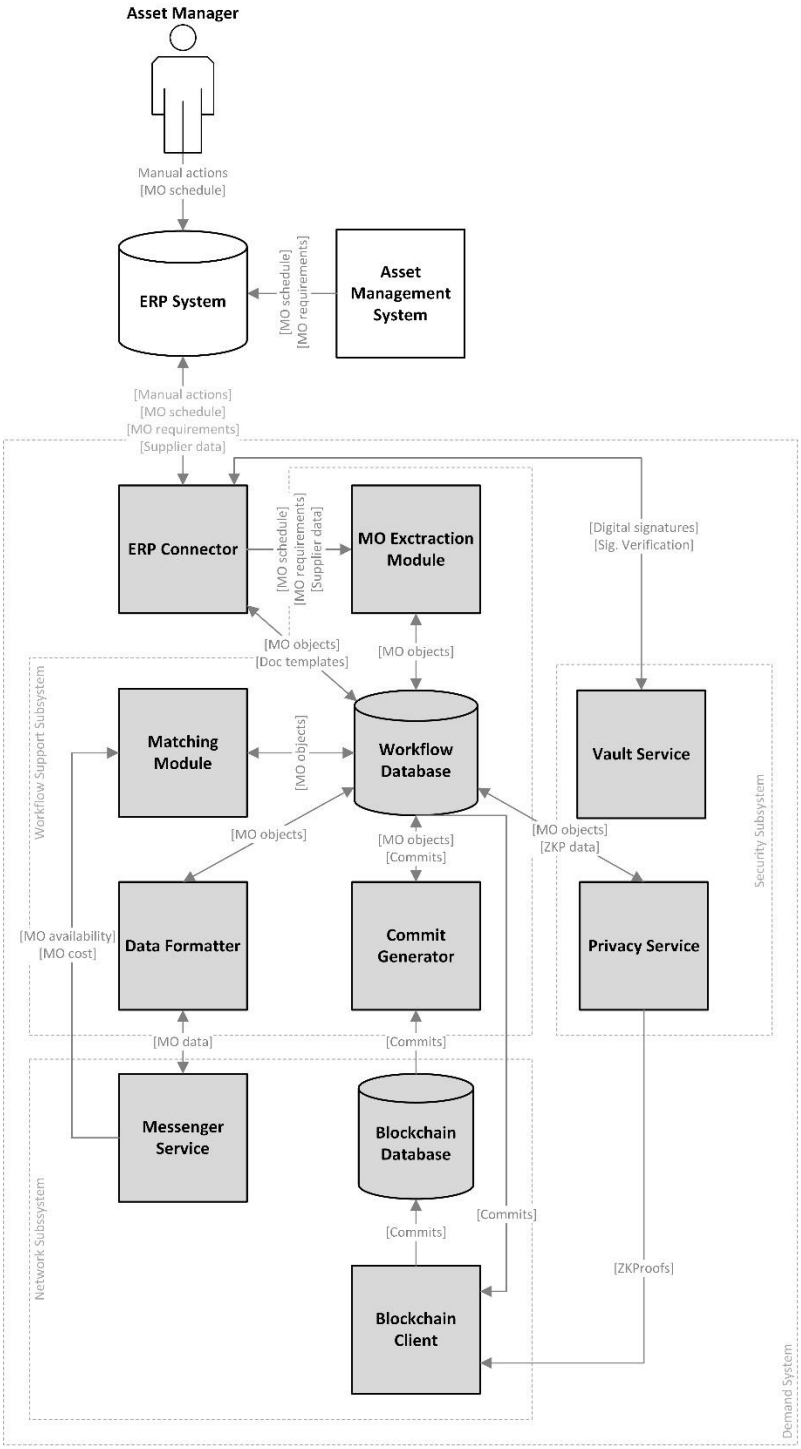


Figure 34: Demand System - final design

3.7.2 Design of the Supplier System

The design development of the Supplier System within the system-of-systems is described in this section. The first goal of the Supplier System is to serve the Demand System with product, asset or workforce availability and cost information, required for automated matching and contracting of maintenance demand with supply. The second goal for the Supplier System is to support the contracting process for every selected supplier to perform in the maintenance operation. Apart from some special features, the Supplier System design quite similar to the Demand System design. Let's start with an overview of the most relevant items, shown in the table below.

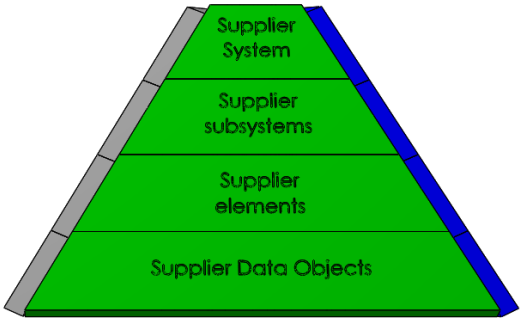


Figure 35: Supplier System theoretical framework for reference

Level of detail	Items	Instance	Design scope	Part of
Supplier System subsystems	ERP system	SAP, Microsoft D365, Excel	No	Digital infrastructure supplier
	To be defined	To be defined	Yes	Design research
Supplier System elements	Messenger service	NATS	No	Baseline Protocol
	Zeroknowledge service	Zokrates toolbox	No	Baseline Protocol
	ERP connector	SAP connector, D365 connector	No	Baseline Protocol
	Blockchain client	Geth	No	Baseline Protocol
	Vault service	Provide Vault	No	Baseline Protocol
	Blockchain database	MerkleTree (local)	No	Baseline Protocol
	To be designed	To be designed	Yes	Design research
Supplier System data objects	Availability schedule template	Machine-readable asset/product/workforce availability template	No	ERP system
	To be designed	To be designed	Yes	Design research

Table 18: Overview of relevant items in Supplier System, subdivided on detail level

On receiving an availability request of a Demand System through the Messenger service, the Commit Generator first fetches the latest Commit out the Blockchain database and compares it to the Commit generated from the data sent with the availability request. Once the request is verified, a to be designed module should calculate the supplier's availability based on the received MO time window, MO lead time and supplier availability calendar that resides in their ERP system. The determined availability is then returned to the requesting Demand System for further processing. If the Demand System publishes a Selection Commit that includes a certain supplier, that supplier knows he is selected to proceed in the contracting process and therefore can block his availability. Eventually the Supplier System receives a business proposal via the Messenger Service, the Commit Generator verifies the business proposal with the Commit in the Blockchain database again, and lets the supplier's Planner review the proposal via the ERP system. In the case the Planner agrees with the business proposal, he puts his signature, the second signature, on the proposal via the Vault service. The business proposal has now become a bilateral signed business agreement. Supplier System then creates a new Commit out of the business agreement, and a ZKProof via the Privacy Service and sends both data objects via the Blockchain Client to the Shield contract. The original business agreement is sent to the Demand System via the Messenger service.

Because the design iterations for the Blockchain System and the Demand System ironed out all the flaws in the initial designs, no additional design iterations had to be made for the Supplier System.

3.7.2.1 Review actors and user stories

The actors directly interacting with the Supplier System will, as previously identified, be the Planner, the Blockchain System and the Demand System. Regarding the user stories, US10 to US12 were specifically designed for the Planner, and thus for the Supplier System. Additionally, to enable US4 for the Demand System, the Supplier System should also be able to support that one.

3.7.2.2 Elements design

Availability Module

To enable US4, the Availability Module was designed to return first days of availability for the span of MO lead time, within the date range of the MO time window. It calculates the availability from the asset/product/workforce availability schedules in the ERP system. Because it is undesirable to disclose commercially sensitive information, the Supplier System only returns the first days on which the supplier is available for the MO lead time, instead of sending the availability calendar spanned according the MO time window.

Program details of the Availability Module are found in Appendix F.

3.7.2.3 Data objects design

No specific data objects had to be designed. The Supplier System uses the already designed MO object and Commits, on top of the out-of-scope availability calendar.

3.7.2.4 Supplier System – final design

Putting together the pieces of the above sections and elements and data objects of the already designed Demand System, we arrive at the final design for the Supplier System, as shown in figure 36. A complete system breakdown is presented in the table below.

Level of detail	Items	Instance	Design scope	Part of
Supplier System subsystems	ERP system	SAP, Microsoft D365, Excel	No	Digital infrastructure supplier
	Security subsystem	N.a.	Yes	Design research
	Communication subsystem	N.a.	Yes	Design research
	Workflow Support subsystem	N.a.	Yes	Design research
Supplier System elements	Messenger service	NATS	No	Baseline Protocol
	Privacy service	Zokrates toolbox	No	Baseline Protocol
	ERP connector	SAP connector, D365 connector	No	Baseline Protocol
	Blockchain client	Geth	No	Baseline Protocol
	Vault service	Provide Vault	No	Baseline Protocol
	Blockchain database	MerkleTree (local)	No	Baseline Protocol
	Data Formatter	N.a.	Yes	Design research
	Commit Generator	N.a.	Yes	Design research
	Workflow database	N.a.	Yes	Design research
	Availability Module	N.a.	Yes	Design research
Supplier System data objects	Availability calendar	Machine-readable asset/product/workforce availability calendar	No	ERP system
	MO object	N.a.	Yes	Design research

Table 19: Supplier System - final design system breakdown

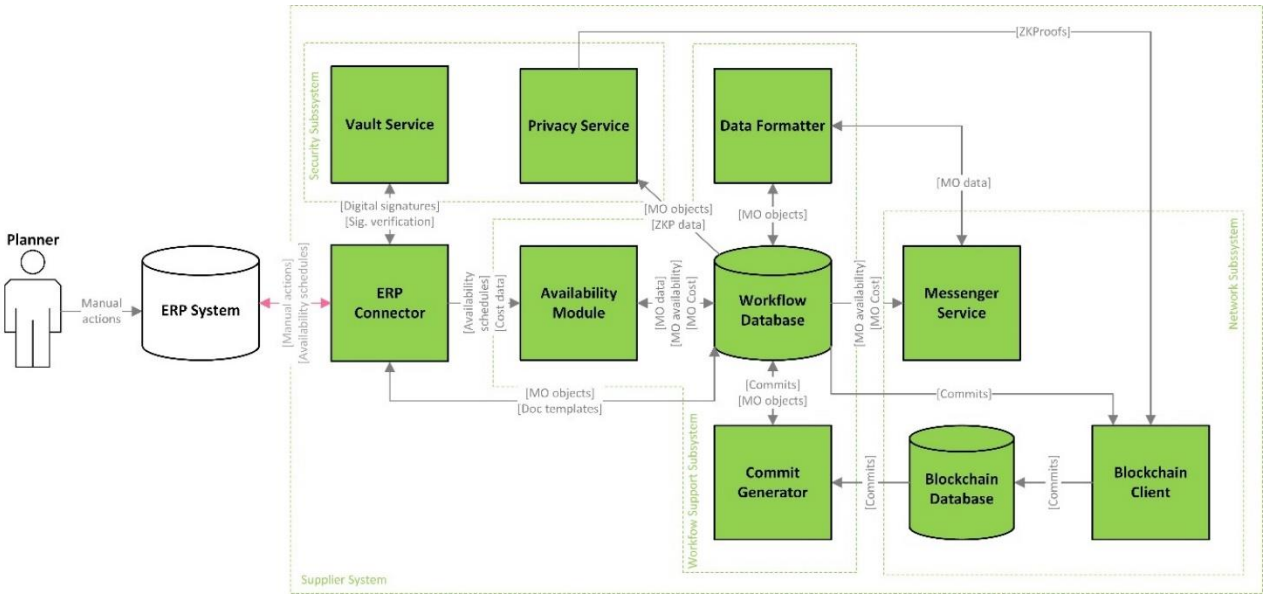


Figure 36: Supplier System - final design

3.8 Design integration - final s-o-s architecture and Workflow

3.8.1 Design of the s-o-s architecture

For the final s-o-s architecture design, the system designs of the Blockchain, Demand, and Supplier systems are integrated into one system-of-systems design, as illustrated in figure 37. The design allows for multiple Demand Systems and Supplier Systems, in varying configurations, to collaborate on the designed s-o-s Workflow with the goal to match and contract supply for each scheduled MO.

The s-o-s allows for multiple Demand Systems to initiate and run Workflows simultaneously for varying configurations of Supplier Systems. A Workflow is initiated by feeding a maintenance schedule to a Demand System. From there, the Workflow is executed by elements of all three of the component systems.

Synchronization of involved systems to a Workflow state occurs via the distribution of Commits and p2p messages. All connected systems receive the Workstep finalizing Commits of all Workflows in their Blockchain Database via the blockchain network. However, only the involved systems also receive a Commit related message via the peer-to-peer Messenger Service. These messages contain a topic, recipients, payload data and Commits as reference. Based on the type of message, first a Commit is generated based on the message payload data that is compared to the Commit received in the Blockchain database to verify the validity of the message and data. If successfully verified, the Messenger Service forwards the data to the subsequent system elements according the Workflow.

3.8.1.1 Initialisation

To initialize current s-o-s, a group of supply chain participants (Workgroup) comes together for the design of one or multiple Workflows. For each Workflow, they design and program generic Workstep verification logic that supports the entire Workflow. Once approved by the Workgroup, a Workgroup admin is appointed to deploy the Shield, OrgRegistry and Verifier contract. After each participant in the Workgroup has downloaded and installed either the Demand or Supplier system, they send their blockchain addresses and messenger service IDs to the admin. The admin adds the blockchain addresses of the Workgroup participants to the OrgRegistry contract, and adds participant IDs to the messenger service. Afterwards, the admin generates a JSON web token (JWT) of his system configuration and via email distributes the JWT to the participants for them to configure their systems. Now the s-o-s is initialized and ready to run Workflows with the connected systems of the Workgroup.

3.8.1.2 Adding and removing participants

When a new participant is involved in the supply chain, all that needs to be done is for that participant to download and install their system of choice. The participant creates a blockchain address and messenger service account, which are shared with the admin. The admin adds the participants blockchain and messenger service details, generated a JWT and emails it to the other participants for them to update their system configuration.

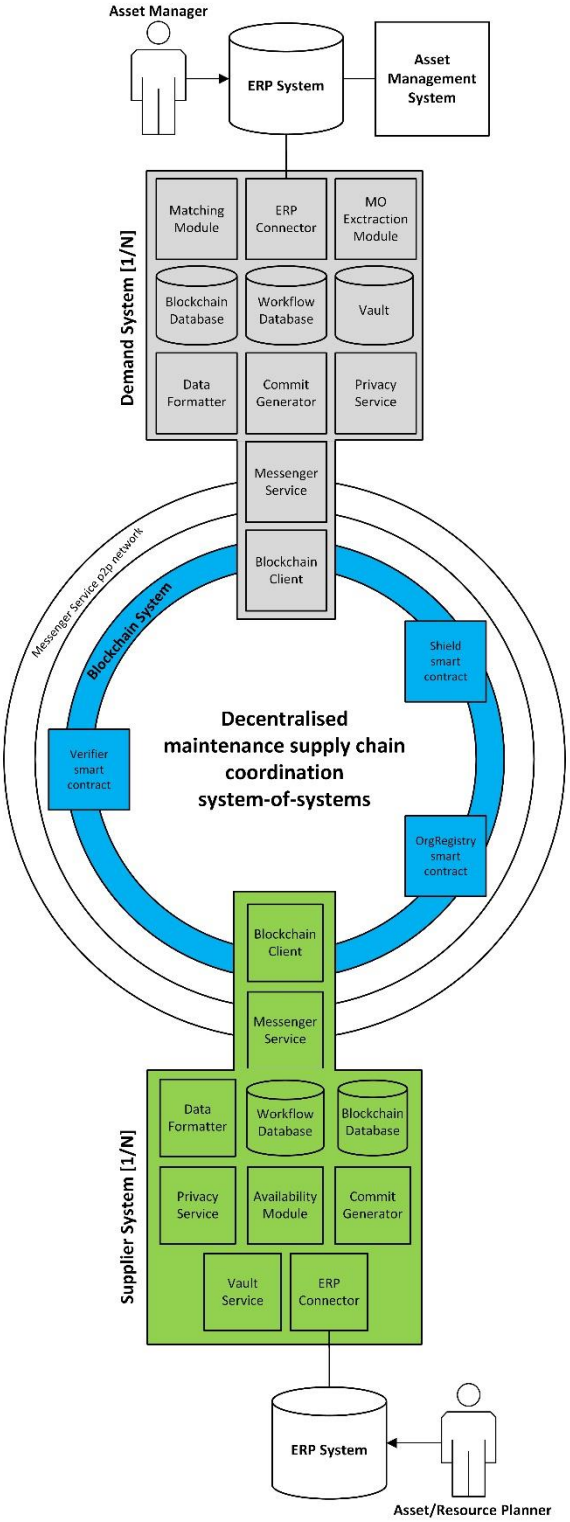


Figure 37: S-o-s architecture – final design

3.8.2 Design of the s-o-s Worfklow

Inspired by the initially designed Workflow and the goal defined for the s-o-s, the final Workflow design is presented in this section. The Workflow ensures the collaboration between systems in the s-o-s for the purpose of achieving its goal of contracting suppliers for every MO requirement, for each scheduled MO, for the given MO time window and lead time. Synchronization to the state of the Workflow occurs through the published Commits by the Blockchain System, as explained in the Blockchain System design. The Commits are a result of verified executed Workstep by one of the involved systems in the Workflow. Since the Workflow is executed in a decentralised manner, multiple systems executed the Worksteps. The topic of the p2p messages tells these systems what to do with the payload data. The five designed Commits designed for the Blockchain system, the Workstep they represent, the related p2p messages and an explanation is given in the table below.

Work step	Commit	Related P2P message (topic)	Message payload	Explanation
1	Genesis	Availability request	State (Workstep number), MO ID, Supplier ID, MO description, MO time window, MO lead time, Commit hash	After linking suppliers to each MO requirement for a particular MO, a Demand System first creates a Genesis Commit, whereafter the availability request follows to all linked suppliers. Commit is also protection against data harvesting. This marks the beginning of the workflow.
2	Selection	Supplier selection	State (Workstep number), MO ID, Supplier ID	Once asset manager confirms selected optimal supplier subset, a Selection Commit is published as an incentive for each requested supplier to either block/unblock their asset/product/resource availability.
3	Business proposals	Business proposal	State (Workstep number), MO ID, Supplier ID, business proposal document and hash	Once asset manager has reviewed and signed business proposals for every supplier for the MO, each of the proposals is notarized on the blockchain via this Commit
4	Business agreements	Business agreement	State (Workstep number), MO ID, Supplier ID, business proposal hash, business agreement document and hash	Once a supplier reviewed and signed a business proposal, the resulting business agreement is notarized on the blockchain via this Commit
5	Finalization / cancelation	Finalization / Cancelation	State (Workstep number), MO ID, Supplier ID	Once all suppliers are successfully contracted, a Final Commit is published so that each supplier knows preparation for MO can begin. Likewise, if one participant publishes a Cancelation Commit all participants know that the MO is (temporary) cancelled.

Table 20: Workflow worksteps, Commits, messages and explanation

Based on the above Workflow expressed in Worksteps, and in support of the goal of the s-o-s the final Workflow design is presented in figure 38. The scoped Workflow design on element-level can be found in the Appendices, however due to its size it was condensed to a system-level Workflow in the figure. As intended, the Workflow is predominantly automated. The only remaining manual activities are marked with a hand cursor. Data objects specifically designed for each system are also presented in the figure.

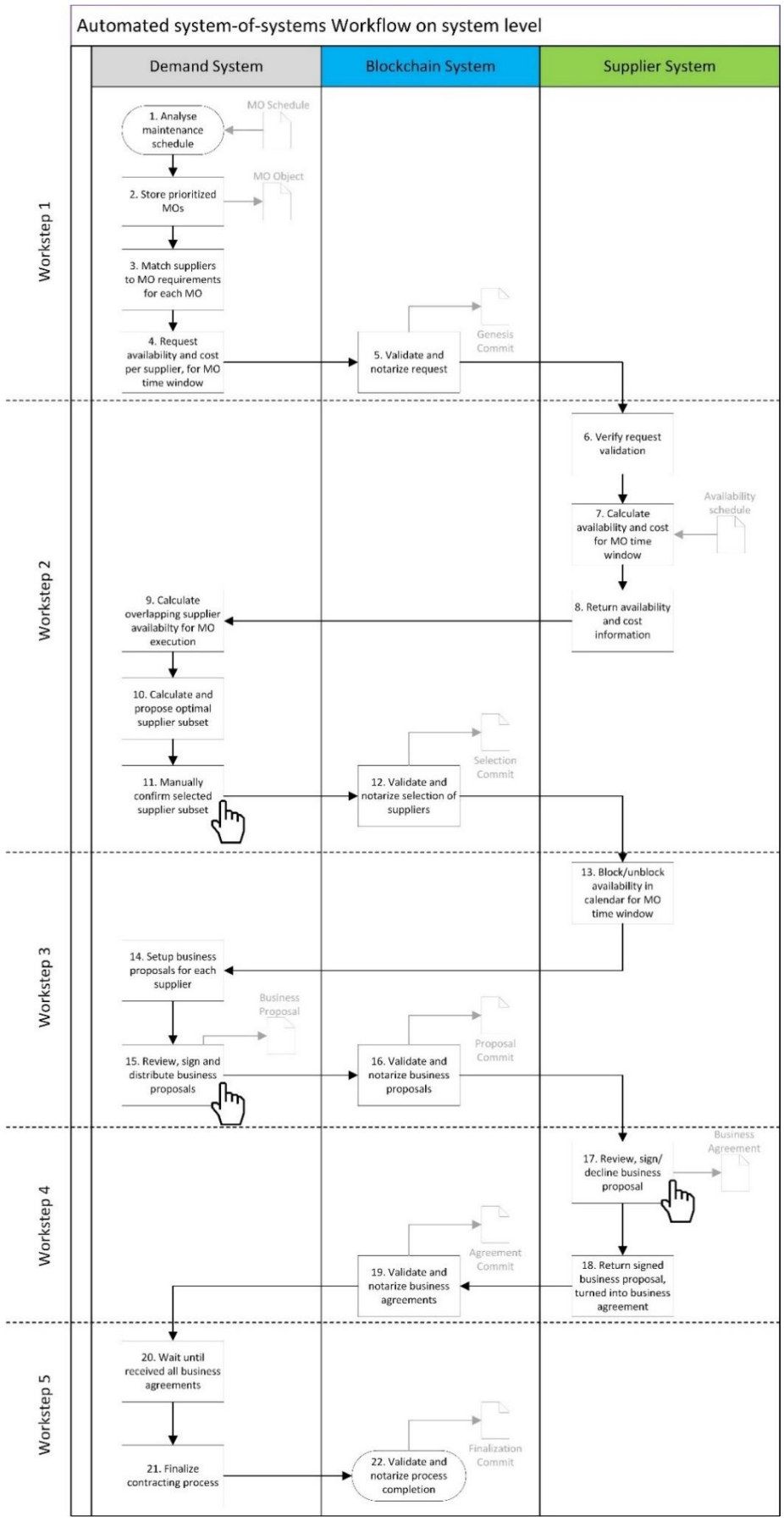


Figure 38: Final s-o-s Workflow design, on system level

3.9 Coding and testing of the System-of-Systems

Coding and testing of the design for the system-of-systems was determined not to be a deliverable in the design scope. However, to prove and verify the feasibility of the design we programmed a minimal, but working instance of the system-of-systems. The entire codebase is found via the following Github link: https://github.com/Meuko/baseline-ganache/tree/master/examples/bri-1/base-example, apart from the Shield and OrgRegistry contract: https://github.com/Meuko/baseline-ganache/tree/master/examples/bri-2/contracts/contracts.

For the coding, we took into consideration one Demand System, one Supplier System and the Blockchain System, which were all created as individual Docker containers on one local machine. Roughly we created 2000 lines of new code in a total time span of 150 hours. The table below shows all the s-o-s components and how they were represented in the minimally coded version of the s-o-s design.

System	System level	Item	Representation
Blockchain system	Subsystems	Network	Ganache
		Nodes	Docker container (Demand System)
			Docker container (Supplier System)
		Virtual Machine	EVM
		Ledger	Ganache
		Consensus Mechanism	Ganache
	Elements	Transactions	Ganache
		Smart contracts	Shield contract
			Verifier contract
			OrgRegistry
		Data structures	MerkleTree
	Data Objects	Commits	Genesis Commit
Demand System	Subsystems	ERP system	Local file location
		Security subsystem	Security subsystem
		Communication subsystem	Communication subsystem
		Workflow subsystem	Workflow subsystem
	Elements	Messenger service	NATS
		Privacy service	Zokrates
		ERP Connector	None
		Blockchain Client	Ganache
		Vault service	None
		Blockchain database	MongoDB
		Commit Generator	Commit Generator
		Workflow database	None
		MO Extraction module	MO Extraction module
		Matching Module	Matching Module
		Data Formatter	Data Formatter
	Data objects	Maintenance schedule	Text file
		MO object	MO object
Demand System	Subsystems	ERP system	Local file location
		Security subsystem	Security subsystem
		Communication subsystem	Communication subsystem
		Workflow subsystem	Workflow subsystem
	Elements	Messenger service	NATS
		Privacy service	Zokrates
		ERP Connector	None
		Blockchain Client	Ganache
		Vault service	None
		Blockchain database	MongoDB
		Commit Generator	Commit Generator
		Workflow database	None
		Data Formatter	Data Formatter
		Availability Module	Availability Module
	Data objects	Availability calendar	Text file
		MO object	MO object

Table 21: Overview of system-of-systems components and their representation in the coded design verification

3.10 Answer to research questions

The following research questions have been answered in this chapter.

5. What are the activities automated by envisioned system-of-systems in current state matching and contracting process?

- Linking suitable suppliers in ERP system to each MO requirement for each scheduled MO
- Creating and distributing availability and cost requests to each linked suitable supplier
- On supplier side, calculating and returning supplier asset/resource/product availability based on MO time window and MO lead time
- Finding overlap in received supplier availabilities for every MO requirement within MO time window, and determine earliest date of feasible MO execution
- Calculating all possible available supplier subsets and propose the best option according given WPP optimisation preferences, for determined MO execution date
- Entering supplier and MO information in templated business proposals and handling the distribution of them

7. What are the implications of the final design?

Developed s-o-s design supports any type of supply chain configuration, and any type of maintenance strategy. It allows for multiple Demand and multiple Supplier Systems to collaborate, using an existing pay-per-use digital infrastructure. The s-o-s is operated through existing company ERP systems, and can be used in parallel to current state manual operations. Since the ERP connector integrates with a broad range of ERP systems, from Microsoft Excel to SAP, even SMEs can participate in the automated workflow. The main bottleneck for adoption is expected to be company holding of and payment with digital currencies, although that also opens the door to immediate financial settlement, decentralized finance (DeFi) and compatibility with machine-to-machine and autonomous operation concepts such as Industry 4.0

8. How is trustworthy processing of commercially sensitive data enabled?

Privacy preserving processing of commercially sensitive data is achieved through the following:

- Local processing of maintenance and availability schedules, whereafter minimally disclosing results are shared directly with involved supply chain participants via p2p messenger service.
- All data, including the business proposals and agreements, processed by Blockchain System first undergoes local cryptographic hashing and zero-knowledge proof transformation before submitted on the public network.

Chapter 4: Case Study

This chapter describes how the developed system-of-system design performs in a case study, to evaluate the validity of the design.

4.1. Introduction

To validate the developed system-of-systems design for automated matching and contracting of maintenance supply for the demand, its performance is evaluated through a case study. A comparison shall be made between current manual matching and contracting process and the automated matching and contracting process enabled by the s-o-s. The key performance indicators for the process, and this case study, is the total process lead time and the total amount process labour, expressed in time. Process lead time is important because shortening the time between (upcoming) asset failures and maintenance execution decreases wind turbine downtime, thus increases energy production and WPP revenue. Process labour time is important for operational expenses, less labour spent on the WPP life cycle long matching and contracting process results in a decrease of operational expenses, thus increases WPP revenue.

The selected business case is matching and contracting maintenance suppliers for the annual maintenance demand of four neighbouring WPPs located at the Dogger bank in the North Sea. The maintenance demand data is used from a maintenance logistics optimisation research by Steendijk and Beelaerts van Blokland [20], on the grouped WPPs in the Dogger bank. Process activity data is used from two internal process analyses performed by BlockLab. One analysis is focussed on the process lead times, subdivided on activity level, for multi-party information exchange required for international shipments. The second analysis focusses on process lead times, on activity level, related to the information exchange for export processes from the Netherlands to the United Kingdom.

4.1. Business case

At the Dogger bank, four large WPPs are being developed, namely Dogger Bank Creyke Beck A & B and Dogger Bank Teesside A & B. Both Creyke Beck WPPs consist of 300 offshore wind turbines and the Teesside WPPs both consist of 200 turbines. The sites are located very closely to each other relative to their distance to shore, therefore they could easily be maintained as a whole. Also, because of their large distance to shore, the WPPs are configured to have their own fleet of maintenance support vessels. The table below provides a combined overview of annual wind turbine failures and maintenance support vessels for each individual WPP and the group total. The annual wind turbine failures are based on failure data from Carroll et al. [2].

	Creyke Beck A	Creyke Beck B	Teesside A	Teesside B	Total WPP group
Wind turbines	300	300	200	200	1000
Minor failures	1853.4	1853.4	1235.6	1235.6	6178
Major failures	318.6	318.6	212.4	212.4	1062
Major replacements	79.2	79.2	52.8	52.8	264
Large vessels	3	3	3	3	12
Small vessels	13	13	11	11	48
Crane vessels	2	2	2	2	8

Table 22: Annual failures and maintenance support vessels for WPPs at the Dogger bank

The goal of the case study is to determine the process lead time and process labour time for the process of matching and contracting maintenance supply for each type of failure in the table, and calculate the annual savings of the designed automated process on the current manual process.

First, all the high-level activities for the contracting and matching process need to be mapped, which is done in figure 39. Second, for each type of failure needs to be determined how much capable suppliers are potentially matched and how much suppliers are actually needed to execute the maintenance. Third, for each activity the lead time and labour time needs to be determined for the manual and automated scenario. And last, the findings are applied on the Dogger bank business case to calculate the process lead time and labour time for both the manual and automated scenario.

4.2 Case study assumptions

For each type of failure has to be assumed what the number and type of MO requirements is in order to determine the number of supplier interactions. The assumptions are based on the MO requirements stated in the research of Carroll et al. [2]

Type of failure	MO requirements
Minor failure	A small vessel
	A team of WPP technicians
	A set of WPP inventory spare-parts
Major failure	A small vessel
	A large vessel
	A team of WPP technicians
	A set of WPP inventory spare-parts
	An OEM spare-part
	A Port
Major replacement	Multiple small vessels
	A team of WPP technicians
	A team of OEM technicians
	A large vessel
	A crane vessel
	A set of WPP inventory spare-parts
	An OEM spare-part
	A Port

Table 23: Assumed MO requirements per type of WT failure

The other assumptions made are:

- There are in total four vessel suppliers in the maintenance supply chain, namely the four individual WPPs.
- There are four WPP technician suppliers, namely the four individual WPPs.
- There are four WPP inventory spare-part suppliers, namely the four individual WPPs.
- There are three OEM spare-part suppliers for each kind of OEM provided spare-part, e.g. turbine blades, gearboxes.
- There are three OEM technician suppliers per OEM provided spare-part, e.g. gearbox specialists.
- There are five ports, where a large or crane vessel could be loaded and unloaded with large spare-parts and maintenance equipment.

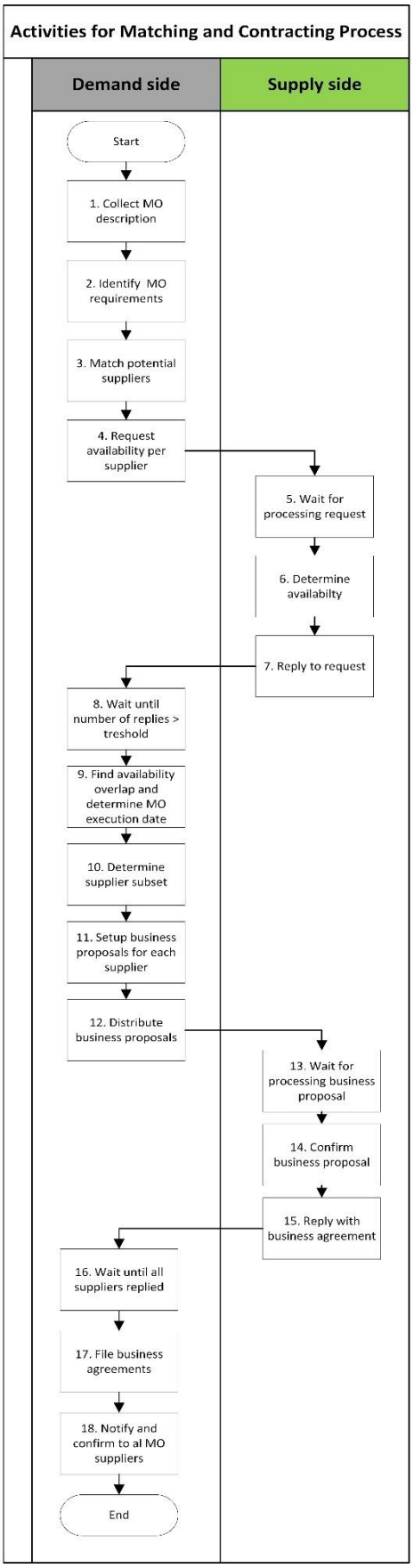


Figure 39: Swimlane diagram of activities related to the supplier matching and contracting

- The total response time for an availability request, or business proposal, of all suppliers combined will be set to 24 hours.
- For both the availability request and the business proposal, a response rate of 100% is assumed. That means all requested potential suppliers have some availability within the presented MO time window, and all suppliers that received a business proposal confirm the proposal 100% of the time. Both assumptions are reasonable because most suppliers offer various assets, and the business proposals are set up with availability and cost knowledge, leading to very high chance of acceptance.

Summarizing on the assumed supply chain size and configuration, for each type of failure the total amount of potential suppliers and required suppliers are:

- Minor failure: 12 potential suppliers, 3 required suppliers
- Major failure: 24 potential suppliers, 6 required suppliers
- Major replacements: 31 potential suppliers, 8 required suppliers

4.3. Case study data

The activity lead time data used for the process lead time and labour time calculations, is determined and reasoned in the table below. The activity IDs match with the activities in the swimlane process diagram of figure 39. Additionally, two variables are introduced:

- *N_pot* = Number of potential suppliers for the MO
- *N_req* = Number of required suppliers for the MO

Activity ID	Manual scenario: lead times [minutes]	Automated scenario: lead times [minutes]	Reasoning
1	1	0	Similar to activity 8 and 9 “info collect” in Blocklab analyses
2	1	0	Similar to activity 8 and 9 “info collect” in Blocklab analyses
3	1	0	Similar to activity 8 and 9 “info collect” in Blocklab analyses
4	5*N_pot	0	Similar to 4 and 31 “request document” in BlockLab analyses
5-8	1440	0	As assumed
9	10*N_pot	0	Similar to activity 2 “system input” in BlockLab analyses, but per potential supplier
10	10	1	Similar to activity 2 “system input” in BlockLab analyses
11	5*N_req	1*N_req	Similar to activity 16 “invoicing” in BlockLab analyses, per required supplier
12	5	1	Similar to activity 18 “share documentation” in BlockLab analyses
13-16	1440	1440	As assumed
17	1*N_req	0	Similar to activity 8 and 9 “info collect” in Blocklab analyses, per required supplier
18	5	0	Similar to activity 18 “share documentation” in BlockLab analyses
Total	2903 + 15*N_pot + 6*N_req	1442+N_req	

Table 24: Process activity lead time determination and reasoning

In the bottom row of Table 24, matching and contracting process lead time formulas are determined for both the manual scenario and the system-of-systems automated scenario. The process labour time formulas for the Asset Manager are easily derived from those, because the waiting times need to be excluded from both.

Process lead time formulas:

- Manual scenario: $t\_lead\_man = 2903 + 15*N\_pot + 6*N\_req$ [minutes]
- Automated scenario: $t\_lead\_auto = 1442 + N\_req$ [minutes]

Demand side process labour time formulas:

- Manual scenario: $t\_labour\_man = 23 + 15*N\_pot + 6*N\_req$ [minutes]
- Automated scenario: $t\_labour\_auto = 2 + N\_req$ [minutes]

4.4 Case Study Results

The derived formulas for the matching and contracting process lead time and labour time are applied to calculate the lead and labour times per failure type and for both the manual and automated scenario. The results of that calculation are found in table 25 below.

	Process lead time [m]			Process labour time [m]		
Failure type	**Manual scenario**	**Automated scenario**	**Time reduction**	**Manual scenario**	**Automated scenario**	**Time reduction**
Minor failures	3101	1445	53.4%	221	5	97.7%
Major failures	3299	1448	56.1%	419	8	98.1%
Major replacements	3416	1450	57.6%	536	10	98.1%

Table 25: Results and improvements on process lead time and labour time for manual and automated scenarios.

A solid lead time reduction is achieved by the automated s-o-s for each failure type. As expected, the time reductions increase with the total number of involved suppliers for a maintenance operation. The time reductions on process labour time by the automated s-o-s are extreme. Although extreme, the results are explainable. For every activity in the matching and contracting process, the automated s-o-s either eliminates the labour or hugely decreases the amount of labour. Summing up all those savings plausibly lead to the extreme results.

In one year, the four combined WPPs at the Dogger bank, totalling up to 1000 offshore wind turbines, have to find the supply for the maintenance demand of 6178 minor failures, 1062 major failures and 264 major replacements. Currently, under the made assumptions, the manual maintenance supply matching and contracting process for each maintenance operation took on average 54.4 hours of lead time, and 6.5 hours of Asset Manager's labour time. The total annual process labour time for the Dogger bank WPPs amounts to 32530 hours.

For similar maintenance demand, the developed automated system-of-systems design decreased the process lead time on average 56% to 24.1 hours, and decreased the Asset Manager's process labour time on average 98% to 0.13 hours. The total annual process labour time for the Dogger bank WPPs operating on the new system-of-systems amounts to only 700 hours. Under the assumption of an hourly rate of €80 for an Asset Manager, the new system-of-systems saves annually 31830 process labour hours, or €2.5 million on process labour cost.

4.5 Answer to research questions

The following research questions have been answered in this chapter.

6. What are relevant KPIs for the automated process?

For both the current state manual process, and the novel designed automated process, the determined KPIs are process lead time and process labor. Process lead time influences the speed of maintenance mobilization, that limits OWT downtime. Process labor reflects the amount of manual work in the process, negatively impacting information processing speed, solution quality, and overall WPP operational performance.

Chapter 5: Discussion

This chapter reflects on the developed system-of-system design. First the verification of the design is discussed, then the design validation and finally the implications of the design.

5.1 Design verification

The developed system-of-systems design is verified in three ways. First shall be verified if the s-o-s still suits the theoretical definitions and architectural principles as developed by Maier. Second, design verification on the basis of programming shall be discussed. And last, the s-o-s design is verified through evaluation of enabling the defined user stories.

Maier describes two important properties of the component systems that make up a system-of-systems; operational independence and managerial independence. Operational independence means that if the s-o-s would be disassembled, the component systems must be able to fulfil customer-operator purposes on their own. To some extend this is true for the developed s-o-s, although each system will lose some of its functionalities. The Blockchain System is operated and managed on its own, but it lost the functionality of synchronizing connected systems for automated maintenance demand supply matching. The Demand System and Supplier System remain operated and managed individually, but in terms of functionality fall back to an ERP system with blockchain and peer-to-peer system communication connectivity.
In regard of the four design principles, the development of the system-of-systems design meets them all. First of all, all individual systems are set up as stable intermediate forms, independent of each other. The second principle of policy triage was met because the included existing components such as the ERP systems, Baseline Protocol elements and the Ethereum blockchain network were left unaltered. For the s-o-s design, elements and data objects were only added, or were given content in case of the Verifier contract. The third principle “leverage at the interfaces” was met through elaborate iterative design of the interfacing Blockchain System, and in particular the Verifier contract. The Blockchain System, together with the Messenger service can be seen as the interface between all the connected Demand and Supplier systems. For the Messenger service, the data object “MO Object” was developed for efficient and easy transport of MO related data. The fourth and last principle of “ensuring cooperation” is arguably met through the design of the automated Workflow. Technical cooperation is not ensured by the Workflow, but incentivized cooperation on entity level is. For all parties, it is much more efficient and cost-effective to operate via the s-o-s design instead of the current manual operations. Besides that, although the Blockchain System can be seen as merely the enabler of automated cooperation between Demand and Supplier Systems, all three systems are of equal importance of the system-of-systems, and the s-o-s would be useless without one of them.

The second design verification is made through evaluation of the programming. Although we only programmed a bare essential implementation of the s-o-s, on one single machine, it was exactly this process that got to the crucial details to enable the automated Workflow. Without it, the design would just be a conceptual, potentially infeasible, design. Now we are sure that the designed automated Workflow completely works in an operational enterprise setting.

The last design verification is through evaluation of enabling the defined user stories. Although not all user stories were specifically designed for, via literature on existing components included in the design, we can assume that they shall be enabled in a finalized enterprise ready instance of s-o-s. The user stories and the evaluation to what extend they are achieved are captured in the table on the next page.

User story	Enabled via	Evaluation
US1	MO Extraction Module	Technically completely enabled by the MO Extraction module
US2	MO Extraction Module, Workflow database	Technically completely enabled by the MO Extraction module and the MO object stored in the Workflow database
US3	MO Extraction Module, and ERP system, ERP connector	Matching suppliers to requirements technically enabled by the Extraction Module. Communication with an ERP system theoretically enabled by the ERP connector, however was not designed and programmed in detail.
US4	Messenger service, Blockchain System, Commit Generator, ERP system, ERP connector, Privacy Service	Technically enabled with a dummy ERP system. Theoretically the ERP connector should enable communication with the ERP system.
US5	Matching Module, ERP System	Technically enabled by the Matching Module with a dummy ERP system. Theoretically enabled if the ERP connector and ERP system allow to search on supplier services and capabilities
US6	Workflow database, ERP system, ERP connector	Theoretically enabled via document templates in the Workflow database that are filled with MO Object data, presented to the user via the ERP system and the ERP connector
US7	ERP system, Vault service, Messenger Service, Privacy Service	Theoretically enabled, the Vault service is used for signing, the ERP system and ERP connector for visualisation and control, the Messenger service for distribution.
US8	ERP system, ERP connector	Not enabled. Theoretically, notifications and maintenance schedule updates are enabled by the ERP system and connector, but no system element was specifically designed for this.
US9	ERP system, ERP connector	Theoretically enabled, there are multiple ways of stopping the automated Workflow; via a programmable button in the UI, simply start working the manual way, delete the s-o-s code.
US10	Blockchain system, Commit Generator, Privacy service, Messenger service	Technically completely enabled. Privacy service generates ZKProof for Genesis Commit, publish Commit via Blockchain System, send relevant data via Messenger service and let Commit Generator verify request
US11	ERP system, ERP connector	Theoretically enabled, there are multiple ways of stopping the automated Workflow; via a programmable button in the UI, simply start working the manual way, delete the s-o-s code.
US12	Workflow database, ERP system, ERP connector	Theoretically enabled via document templates in the Workflow database that are filled with MO Object data, presented to the user via the ERP system and the ERP connector.
CS1	Blockchain System	Technically completely enabled. Theoretically maximum size of the supply chain is related to the maximum number of transactions per time unit the used blockchain platform can process.
CS2	S-o-s, Baseline Protocol	Technically completely enabled, a new user only has to download and install the software for either the Demand or Supplier system. Via a JSON Web Token, the user's system is completely configured according the s-o-s settings.
CS3	S-o-s	Technically completely enabled, results of the Case Study revealed a 80% labour reduction is achieved.
CS4	S-o-s	Theoretically enabled. The s-o-s is operated through an existing, and thus familiar, ERP system. The user only has to be able to do a few manual actions which can be enabled via an easy to use UI.
CS5	S-o-s, Blockchain system	Theoretically enabled. Both by the Blockchain System and how the s-o-s is designed, it is extremely hard to act maliciously.

Table 26: Evaluation of the user stories

5.2 Design validation

The validity of the design is evaluated through the results of the case study. Purely on the basis of the results, the design achieved its goal of automating the matching and contracting of maintenance supply with demand to ensure operational feasibility. While doing so, it proved to unlock tremendous time savings in terms of process lead time and process labour time, on average respectively 56% and 98%. Particularly the latter 98% savings in process labour time is almost too good to be true. Considering the fact that the automated system-of-system design eliminates many manually executed process activities and hugely decreases the amount of labour for the remaining activities, makes the 98% labour time savings more plausible. It could be argued that mistakes have been made in determining the process activity lead times, however their values are based on elaborate process analyses that include a total breakdown of timed activities related to the information exchange in international shipment processes and export processes from NL to the UK. On top of that, each determined activity lead time can be logically justified. For additional validation, annual WPP maintenance simulations could be executed with varying supply chain sizes, configurations, and supplier availabilities.

5.3 Design implications

From a practical business perspective, how should the design be regarded? The design is a pay-per-use system-of-systems that automates the processes of matching potential suppliers to maintenance operation requirements; requesting their availability and cost information; aggregating received supplier availabilities to find overlap within the scheduled time window; selecting suppliers for the MO according set preferences, and facilitates the contracting process via preparation of templated business proposals, their distribution, and digital signing.

Because of the pay-per-use infrastructure, no large capital investments have to be done in additional hardware. Also, because the ERP connectors are supposed to connect with every level of enterprise management system, from advanced SAP and Dynamics365 to simple Google Sheets or Excel, the s-o-s allows for participants with limited financial resources, such as SMEs, to be involved in the automated Workflow.

The s-o-s consists of maintenance Demand Systems and maintenance Supplier Systems, that use the Ethereum public blockchain network and a peer-to-peer messenger service to synchronize the state of the Workflow between systems of the supply chain participants. Using the Ethereum blockchain means that for every verification of an executed Workstep, and thus the publication of a Commit, a transaction fee has to be paid in digital currencies. This is considered to be the main obstacle for adoption of developed design.

Commercially sensitive data never leaves the enterprise premises. All sensitive data processing happens locally, whereafter only minimally disclosing pieces of availability data, or cryptographically transformed Workflow data are shared in the system-of-systems.

The s-o-s is operated through the user interface of existing ERP systems, meaning that no additional system needs to be operated and employees remain operating on the systems they trust and are familiar with.

The integration of new supply chain participants is quick and easily done by simply installing the Demand or Supplier System software and configure the system through a JSON Web Token sent by the s-o-s admin via email. The new participant only needs to create a public blockchain address and a messenger service ID. The admin adds the blockchain address to the OrgRegistry contract, while the other participants add the new messenger service ID to their respective systems. Hereafter, the new participant is fully integrated in the s-o-s and is enabled to collaborate in the automated Workflow.

Chapter 6: Conclusion and recommendations

6.1 Conclusion

Because of the renewable energy targets of the European Union, installed offshore wind capacity is projected to grow over a tenfold of current capacity for 2050. Therefore, the offshore wind maintenance demand, and its supply chain in terms of number of available suppliers, assets, resources, and products, has to increase proportionally. On top of that, due to energy production optimisation, the average size of an offshore wind turbines increases. Larger wind turbines consist of larger, heavier parts that have to be maintained at higher altitudes, which increases the MO complexity and the number of requirements needed to successfully execute a maintenance operation. Because both the number of MOs and the number of MO requirements increases, the amount of communication and information processing for coordinating these operations also increases.

To achieve the renewable energy targets, it is important that the cost of offshore wind energy is kept low and competitive. The price of offshore wind energy relies heavily of the operational performance of a WPP. OWT downtime and O&M costs are dominant factors that have a negative influence on the operational performance of a WPP. To keep OWT downtime and O&M cost low, it is of importance that maintenance is mobilized as quickly as possible, and in a cost-effective manner.

For current state maintenance mobilization, MOs are scheduled automatically, whereafter asset management teams have to manually match, contract and coordinate a variety of suppliers for each MO. Key activities in this complex process are:

- Linking suitable suppliers to each MO requirement;
- Requesting their cost and availability for MO lead time within the scheduled time window;
- Finding overlap in supplier availabilities for MO lead time and scheduled time window;
- And contracting them for feasible MO execution.

Key pieces of information for these activities are the commercially sensitive maintenance and availability schedules, all residing in supply chain participant ERP systems. Key performance indicators for this process are process lead time and process labor. Lead time is important to the maintenance mobilization speed that limits OWT downtime. Process labor reflects the amount of manual work in the process, negatively impacting information processing speed, solution quality, and overall WPP operational performance.

To accommodate the huge increase in MOs and related information processing, manual execution of the matching, contracting and coordination of suppliers for each MO is considered to be a bottleneck. The human information processor has limited capacity that shall result in slower and less cost-effective maintenance mobilization, which are both important to the WPP operational performance. To overcome this bottleneck, the solution is found in designing an automated system-of-systems consisting of maintenance demand and supplier systems, that together execute a matching, contracting and coordination workflow for each scheduled MO. Since the workflow requires trustworthy collaboration of many independently owned ERP systems and secure processing of commercially sensitive information, blockchain technology selected as part of the solution. As a result, the following research question was defined.

How to design a technical feasible decentralized system-of-systems that enables automated matching and contracting of maintenance supply for scheduled demand through privacy preserving processing of commercially sensitive data?

Through the answering of a series of sub-questions, the answer to the defined research question was found.

1. *What is the theoretical framework to define and describe envisioned system-of-systems?*

System-of-systems theory as defined by Maier and Veeke et al., and blockchain technology as described by Nakamoto and Buterin. S-o-s theory is proposed to be extended with an additional definition for describing passive, standardized, pieces of information consumed by other parts in the s-o-s, that can be separated from the element class. These pieces of information are defined as "Data objects", and are vital to the operational success of an s-o-s. In this research data objects appear in the form of data structure templates (schedule templates, hash (Commit) designs, MO Object) or as unique identifiers (hashes, blockchain addresses). In physical systems, they appear as serial numbers, stickers, tags, and general signalling.

2. *What is the most suitable design method for envisioned system-of-systems?*

A design method based on the agile blockchain system engineering method "ABCDE" of Marchesi et al. complemented with Baseline Protocol design features of defining the Workgroup and design of the Workflow.

3. *How will the design be verified?*

Design verification comes from:

- Teaming up with experienced blockchain engineer to program core functionalities.
- Compliance to architectural principles of s-o-s theory.
- Evaluation of realisation of defined user stories as part of the ABCDE method.
- Regular consults with lead developers and solution architects of BP

4. *How will the design be validated?*

Through a case study where its performance in terms of process lead time and process labor, is compared to current state matching, contracting and coordinating suppliers for scheduled maintenance demand. After definition of the process activities, formulas are defined to calculate lead time and process labor in both current state and designed automated scenario. Activity lead time data needed to develop the formulas comes from two process analyses for information exchange related to international transport. For a maintenance demand based on the work of Carroll et al. and an assumed supply chain size for the four WPPs located in the Dogger Bank, the impact on lead time and process labor for three types of maintenance operations is calculated.

5. *What are the activities automated by envisioned system-of-systems in current state matching and contracting process?*

- Linking suitable suppliers to each MO requirement for each scheduled MO
- Creating and distributing availability and cost requests to each linked suitable supplier
- On supplier side, calculating and returning supplier asset/resource/product availability based on MO time window and MO lead time

- Finding overlap in received supplier availabilities for every MO requirement within MO time window, and determine earliest date of feasible MO execution
- Calculating all possible available supplier subsets and propose the best option according given WPP optimisation preferences, for determined MO execution date
- Entering supplier and MO information in templated business proposals and handling the distribution of them

6. *What are relevant KPIs for the automated process?*

For both the current state manual process, and the novel designed automated process, the determined KPIs are process lead time and process labor. Process lead time influences the speed of maintenance mobilization, that limits OWT downtime. Process labor reflects the amount of manual work in the process, negatively impacting information processing speed, solution quality, and overall WPP operational performance.

7. *What are the implications of the final design?*

Developed s-o-s design supports any type of supply chain configuration, and any type of maintenance strategy. It allows for multiple Demand and multiple Supplier Systems to collaborate, using an existing pay-per-use digital infrastructure. The s-o-s is operated through existing company ERP systems, and can be used in parallel to current state manual operations. Since the ERP connector integrates with a broad range of ERP systems, from Microsoft Excel to SAP, even SMEs can participate in the automated workflow. The main bottleneck for adoption is expected to be company holding of and payment with digital currencies, although that also opens the door to immediate financial settlement, decentralized finance (DeFi) and compatibility with machine-to-machine and autonomous operation concepts such as Industry 4.0

8. *How is trustworthy processing of commercially sensitive data enabled?*

Privacy preserving processing of commercially sensitive data is achieved through the following:

- Local processing of maintenance and availability schedules, whereafter minimally disclosing results are shared directly with involved supply chain participants via p2p messenger service.
- All data, including the business proposals and agreements, processed by Blockchain System first undergoes local cryptographic hashing and zero-knowledge proof transformation before submitted on the public network.

After combining the answers to all sub-questions into on, the answer to defined research question is:

By following a design approach based on ABCDE method, complemented met Baseline Protocol design features, described and guided by an extended theoretical framework of system-of-systems theory a s-o-s design can be developed for automated matching of maintenance demand with supply. Technical feasibility is ensured through programming of core functionalities and consulting experts. Automation is enabled through a designed s-o-s workflow executed by designed Demand and Supplier Systems, enforced by Blockchain System. Privacy preserving processing of sensitive information is enabled by local processing of all sensitive data and only sharing minimally disclosing results directly peer-to-peer, or cryptographically transformed on the blockchain network.

Limitations to this research is the lack of involvement of WPP asset managers or suppliers in the offshore wind maintenance supply chain, therefore the defined user stories and their evaluation are solely based on the working experience of the researcher. Also, the resulting impact of the design on

process lead time and process labor are based on a single case study, for which the supply chain size was assumed and only a happy flow was considered.

The academic contribution of this research is the addition of the definition of “Data Objects” to the s-o-s theory for defining and describing vital, standardized, informational parts of an s-o-s that are consumed by other parts of the s-o-s. The industry contribution is a novel generic s-o-s and workflow design for automated decentralised contracting and coordination of suppliers for scheduled operations, that supports any size, type and configuration of supply chain.

6.2. Recommendations for further research

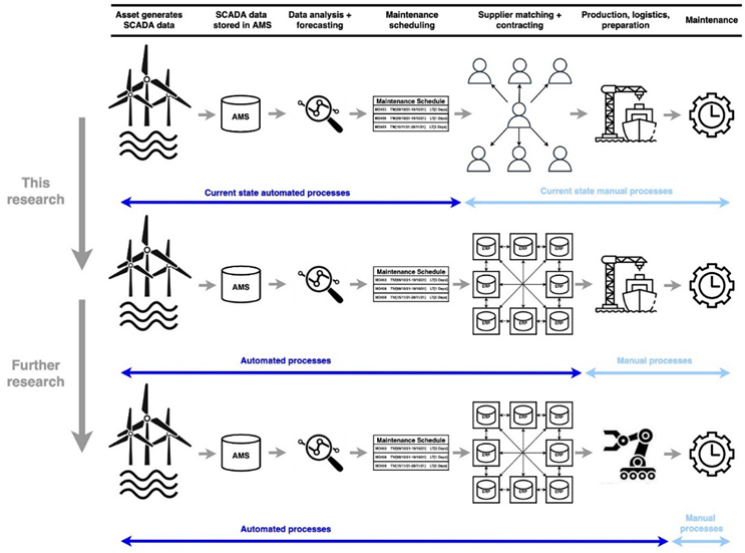
Based on the limitations of this research and the implications of the developed design, the recommendations for further research are given in two categories. First, routes to improve the current research are recommended. Second, recommendations on how to extend and increase adoption of current research are presented.

Recommendations to improve current research:

- Inspire asset managers and supply chain participants with developed novel s-o-s architecture and automated workflow, and involve them in the development of additional user stories and workflows. Then, perform research into advanced applications of zero-knowledge proofs to enable and further extend developed user stories and workflows.
- Research into methods of automated searching in digital records of supplier information with the goal to link their offered assets, products and resources to detailed maintenance operation requirements for a large variety of maintenance operations.
- Perform quantitative simulations on WPP annual operational performance, for differently located WPPs, and sized and configured supply chains to determine the best applicability of developed s-o-s design.

Recommendations to extend current research and increase adoption:

- Theoretically, with current state automated maintenance scheduling, developed design for automated contracting and coordination, and financial settlement in digital currencies, OWTs could be enabled to control, order and pay for their own maintenance. It would be interesting to research autonomous operation scenarios and their requirements, that could be enabled with current research.
- Research on the feasibility of maintenance operation marketplaces enabled by developed s-o-s design, and its impact on the maintenance industry.



Bibliography

[1] R. Wiser, M. Hand, J. Seel, and B. Paulos, “Reducing Wind Energy Costs through Increased Turbine Size: Is the Sky the Limit? Berkeley Lab study,” 2016.

[2] J. Carroll, A. Mcdonald, and D. Mcmillan, “Failure Rate , Repair Time and Unscheduled O & M Cost Analysis of Offshore Wind Turbines,” *Wind Energy*, vol. 19, no. 6, p. 214, 2015.

[3] C. Stock-Williams and S. K. Swamy, “Automated daily maintenance planning for offshore wind farms,” *Renew. Energy*, vol. 133, no. April, pp. 1393–1403, 2019.

[4] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008.

[5] N. Yvas, A. Beije, and B. Krishanamchari, *Blockchain and the supply chian*. 2019.

[6] R. L. Ackoff, “Towards a system of systems concepts,” *Manage. Sci.*, vol. 17, no. 11, 1971.

[7] H. Veeke, J. Ottjes, and G. Lodewijks, *The Delft Systems Approach*, vol. 53, no. 9. 2008.

[8] M. W. Maier, “Architecting Principles for Systems-of-Systems,” *Syst. Eng.*, vol. 1, no. 4, 1999.

[9] V. Buterin, “A next-generation smart contract and decentralized application platform,” *Etherum*, no. January, pp. 1–36, 2014.

[10] S. S. Panda, B. K. Mohanta, U. Satapathy, D. Jena, D. Gountia, and T. K. Patra, “Study of Blockchain Based Decentralized Consensus Algorithms,” in *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 2019, vol. 2019-Octob, no. December, pp. 908–913.

[11] M. Du, X. Ma, Z. Zhang, X. Wang, and Q. Chen, “A review on consensus algorithm of blockchain,” in *2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017*, 2017, vol. 2017-Janua, pp. 2567–2572.

[12] E. Piscini and L. Kehoe, “Blockchain & Cyber Security. Let ’ s Discuss,” 2018.

[13] J. Garzik and Bitfury Group, “Public versus Private Blockchains Part 1: Permissioned Blockchains,” *https://bitfury.com/docs/page:3*, vol. 1. pp. 10–11, 2015.

[14] M. Rauchs, A. Blandin, K. Bear, and S. McKeon, “2nd Global Enterprise Blockchain Benchmark Study,” 2019.

[15] Baseline Protocol Community, “Baseline Protocol,” *https://docs.baseline-protocol.org/*, 2020. .

[16] P. Rook, “Controlling Software Projects.,” *Softw. Eng. J.*, vol. 1, no. 1, pp. 7–16, 1986.

[17] L. Marchesi, M. Marchesi, and R. Tonelli, “ABCDE -Agile Block Chain dApp engineering,” *arXiv*, vol. 1, no. November, 2019.

[18] Green Port Hull; BVG Associates, “Job Roles in Offshore Wind,” 2017.

[19] M. Cohn, *User Stories Applied for Agile Software Development*. 2009.

[20] J. Steendijk and W. W. A. Beelaerts van Blokland, “Optimization of Maintenance Operations for Offshore Wind Farms,” in *Hamburg International Conference of Logistics (HICL) – 28*, 2019, no. September, pp. 55–82.

Appendices

Appendix A – Detailed Design Iterations

Design Iteration 1: Element adjustment: Output of verifier contract	
Before	Use Verifier Contract to calculate the availability of a supplier after receiving the Time Window and Task Length input of the WPP and the Availability Calendar input of the supplier.
Reasoning	Good way to use sensitive data because is only handled by associated party and pseudo-anonymously given to trustworthy contract on decentralized network
Involved user story	US4: Ability to request supplier availability
Problem	Zeroknowledge Proof-based Verifier Contract is only able to calculate if a given input is either true or false. It cannot be used as a general calculator that outputs any unknown result from given inputs. Additionally, the Verifier Contract only takes the input given by a single party.
When was problem encountered	Tests in Remix (online IDE, also for testing smart contracts) with simplified initial contract design revealed that the output of the calculation inside the contract always has to be known upfront. Therefore, an unknown output can never be calculated.
How solution was presented	Additional detailed analysis of Zeroknowledge-Proof literature and examples, complemented with various tests in Remix together with Hamza(online IDE, also for testing smart contracts)
After	Do the availability calculation off-chain, outside of verifier contract as it is unable to provide the necessarily functionalities. Led to the design of the availability module

Design Iteration 2: Element adjustment: Generic verifier contract per MO	
Before	Create unique verifier contract for each individual work step in order to verify compliance with specific work step related business logic
Reasoning	Splitting up verification for each individual work step keeps the verification scheme simple and maximises security as more process details can be taken into account.
Involved user story	US4, CS4, CS5
Problem	Managing the routing of the work step data to the related verifier contract is very hard and not incorporated in Baseline Protocol. Additionally, each deployment of a contract adds significantly to the operational costs
When was problem encountered	When brainstorming about the routing to the various contracts and the absence of routing information in the Baseline literature
How solution was presented	Discussions on the matter in the Baseline Slack with Sam Stokes and Brian Chaimberlain led to the insight that it is more economical to deploy only one contract and more simple too, as it would become very complex to create a mechanism that is able to listen to a contract deployer, determine which contract belongs to which workstep and to update the smart contract routing of each connected system in the network. It would also lead to the need of a human or automated contract deployer for each MO.
After	Design a Verifier Contract that is generic and able to verify the logic of each work step

Design Iteration 3: Element adjustment: Reusable generic verifier contract	
Before	Create generic verifier contract for each individual MO to verify compliance with the MO related business logic
Reasoning	Splitting up verification for each MO maximises security as more MO details can be taken into account.
Involved user story	CS4, CS5
Problem	Routing, design and deployment of an MO-related Verifier Contract increases operational costs, design complexity and process labour
When was problem encountered	In the discussions with the Baseline developers regarding previous iteration.
How solution was presented	In the discussions with Sam Stokes and Brian Chaimberlain about the Verifier Contract setup they gave the advice that it was best practise to just create one reusable generic verifier contract as it make the system design much less complex and cheaper to operate
After	Design a generic Verifier contract that is reusable for all the handled MOs

Design Iteration 4: Element adjustment: Verifier contract hashing algorithm	
Before	The Verifier Contract was designed to verify the proof of correct formulation of a multi-stage hashing operation, where two initial batches of combined data are hashed separately, whereafter the resulted hashes are combined and hashed again for the final result. The security layers to protect the sensitive business data are as follows; 1. Sensitive data, such as business proposals or contracts, are hashed first. Only the suppliers involved in the MO are in possession over this shared data with the WPP 2. Those hashes are combined with less sensitive data, such as company names, and are hashed again according a generic formula only the supply chain members know of. The resulting hash is used as Commitments, which will be emitted on the public network after successful verification. Reconstruct the original data from a multi-staged hashing operation is currently impossible. 3. The final security layer is the Verifier Contract itself. To be able to generate a successful verification (i.e. proof of compliance and execution with the process), one first has to get possession of the right data, then has to get possession over the generic formula and formulate the multi-staged hashing operation the right way. From that combination of data and hashes, a Zero-knowledge proof (zk-SNARKS) has to be generated via a generic arithmetic circuit only possessed by the supply chain members. That ZKP has to be fed to the Verifier Contract via an Ethereum address that is listed in the OrgRegistry Contract which holds the addresses of all the connected supply chain members. In summary: a correctly formulated ZKP representation of a combination of data and hashes is only allowed to be send to the Verifier Contract by an OrgRegistry listed address, and by successful verification a process step-related Commitment – also built from a correctly formulated combination of hashes and data - is emitted on the public network. The hashing algorithm used for these operations is the SHA256 algorithm, which produces a 32-character fixed length 256-bit hash.
Reasoning	Well-known, widely used and secure hashing algorithm.
Involved user story	CS5
Problem	Using SHA256 for multi-stage hashing operations in an on-chain Verifier Contract resulted in unnecessary long processing times, therefore expensive operations, and extreme code bloat when deploying the contract on the network.

When was problem encountered	During deployment and testing of the Verifier Contract.
How solution was presented	Discussions with Hamza revealed he was aware of the fact that within the Baseline Protocol development community they were also looking at alternatives for the slow and expensive SHA256 hashing algorithm. One of the promising alternatives was the Pedersen algorithm, which was known for its efficiency and therefore very suitable for use in zk-SNARKS circuits.
After	The SHA256 hashing algorithm in the formula was replaced with the Pedersen hashing algorithm, which turned out to be a factor of 10 to 100 times faster

Design Iteration 5: Added data objects: Genesis Commit and Selection Commit	
Before	The initial set of Commits consisted of the Availability, Proposal, Contract, and Final Commit.
Reasoning	The initial set covers all the essential process steps that occur in the sourcing and contracted process, therefore it seemed sufficient
Involved user story	US4, US10, CS5
Problem	Iteration 4 led to the unlimited ability for the WPP to request availability data from the suppliers, as the Availability Module can be requested on demand by the WPP. Additionally, lead time in-between the supplier selection and the actual distribution of a business proposal could cause a supplier to become unavailable again by external business requests.
When was problem encountered	Trough Iteration 4, the problem of unlimited data mining came in existence. The insight for impermanent availability of the supplier came through a realization in one of our calls with the Baseline developers
How solution was presented	It was my own idea to add a Genesis Commit to mark the start of a new sourcing process for an MO and simultaneously act as a monetized motivation (in terms of transaction costs) for a data request. The solution to claim the availability for a supplier by notification about their selection came as an idea in the meeting with Kyle and Daven
After	The Availability Commit was replaced by the Genesis Commit as a monetized motivation to perform a single availability request, that is automatically authorized by system of the supplier. The Selection Commit is also incorporated and will be published after the most optimal set of suppliers is selected by the system to perform the MO. Backed by this commitment, a supplier can safely clear his Availability Calendar until he receives a business proposal from the WPP

Iteration 6: Added data object: MO data object for MO data storage	
Before	Initial idea was to hold a database in each system to store process and MO related data, that could be retrieved whenever needed.
Reasoning	Provision of necessary, basic functionalities
Involved user story	US1-US10
Problem	Such a configuration results in a lot of requests to the database, and also made management and ordering of the database unnecessary complex
When was problem encountered	During integrated testing we found that we needed to add a lot of effort to store and retrieve the right data at each intermediate step
How solution was presented	Hamza proposed to combine all the various data objects into one major object where all the data related to one MO, is stored and passed around between the system elements
After	For each MO handled by the system, a MO specific data object is continuously populated with MO related data throughout the entire process. That object can be easily shared with connected suppliers through the messenger service, in order for the suppliers to have the right data to reconstruct passed Commits

Iteration 7: Added element: Commit Generator	
Before	To be able to verify the correctness of published Commitments, the idea was to use the Commitment Manager which is a part of the standard components of the Baseline Protocol
Reasoning	As an essential feature of the protocol, functionality should be provided to verify published commitments
Involved user story	US4, US10, CS3, CS5
Problem	Commitment Manager does not provide the functionality of verification of Commitments
When was problem encountered	During a virtual run through the process, realized was that there is no function that provides verification of commitments that reached the local Merkle Tree
How solution was presented	We came up with the simple solution to create an off-chain copy of the Verifier Contract in order to be able to regenerate the published Commitments
After	A Commitment Generator was created and added to each system stack to enable verification of published Commitments. The system of a supply chain member is now able to automatically generate a Commitment based on data received in the MO data object. Whereafter the resulting Commitment can be compared to the original one published on the network and stored in each Merkle Tree

Iteration 8: Added element: Data Formatter	
Before	Data formatting, if required, will be done at the particular system element when it is needed, or at its predecessor.
Reasoning	The expected amount of formatting is considered rather limited
Involved user story	US1-US10
Problem	It turned out that in- and outbound communication via the messenger service required a lot of reformatting each time, affecting various components that interact directly with the messenger service. Message payload data needs to be mapped and stored in the right field of the MO data object and vice versa
When was problem encountered	During integrated testing we found out that we needed to add a lot of mapping and formatting code at each element interacting with the messenger service
How solution was presented	In discussions with Hamza on how to handle this problem
After	An element called Data Formatter was added to each system stack for the mapping and formatting of in- and outbound data from the MO object to the messenger service and vice versa

Appendix B – zk-SNARK program for Verifier Contract

Program function: Verify proof of correct hash formulation of a variety of inputs.
Program language: Zokrates' DSL (Domain specific language)
Link: https://github.com/Meuko/baseline-ganache/blob/master/examples/bri-1/base-example/src/zkp/src/stateVerifierP.zok

```
import "hashes/pedersen/512bit" as pd
import "utils/pack/u32/pack128" as pack128
import "utils/pack/u32/unpack128" as unpack128

def main(private field State, private field MJ_ID, private field
        SupplierID, private field DocHash1, private field DocHash2, private
        field ContractH1, private field ContractH2, private field LC1,
        private field LC2, private field NC1, private field NC2) -> (field,
        field):

    field a = if (State == 1 || State == 2 || State == 5) && ContractH1 ==
        State && ContractH2 == State && SupplierID == State && DocHash1 ==
        State && DocHash2 == State then 0 else 1 fi
    field b = a * ContractH1
    field c = a * ContractH2
    field d = a * SupplierID
    field e = a * DocHash1
    field f = a * DocHash2

    field g = if (State == 3 || State == 4) && DocHash1 != ContractH1 &&
        DocHash2 != ContractH2 then 1 else 0 fi
    field h = g * b
    field k = g * c

    u32[16] preHash1 = [...unpack128(State), ...unpack128(MJ_ID),
        ...unpack128(d), ...unpack128(LC1)]
    u32[16] preHash2 = [...unpack128(e), ...unpack128(f), ...unpack128(h),
        ...unpack128(LC2)]

    u32[8] Hash1P = pd(preHash1)
    u32[8] Hash2P = pd(preHash2)

    u32[16] NewHash = [...Hash1P[0..4], ...Hash1P[4..8], ...Hash2P[0..4],
        ...Hash1P[4..8]]

    u32[8] NewHashP = pd(NewHash)
    field[2] NewHashField = [pack128([...NewHashP[0..4]]),
        pack128([...NewHashP[4..8]])]

    assert(NewHashField[0] == NC1)
    assert(NewHashField[1] == NC2)

    return MJ_ID, State
```

Appendix C – Detailed program of MO Extraction Module

To save space, only a link to the location of the program is provided.

https://github.com/Meuko/baseline-ganache/blob/master/examples/bri-1/base-example/src/mods/extract/extract.ts

Appendix D – Detailed program of Commit Generator

To save space, only a link to the location of the program is provided.

https://github.com/Meuko/baseline-ganache/blob/158713d7e57a88ba40c7d8621b47e04951d3172f/examples/bri-1/base-example/src/index.ts#L874

Line 874 – line 1040

Appendix E – Detailed program of Matching Module

To save space, only a link to the location of the program is provided.

https://github.com/Meuko/baseline-ganache/blob/master/examples/bri-1/base-example/src/mods/allign/allign.ts

Appendix F – Detailed program of Availability Module

To save space, only a link to the location of the program is provided.

https://github.com/Meuko/baseline-ganache/blob/master/examples/bri-1/base-example/src/mods/avail/avail.ts

Appendix G – System-of-systems Workflow design on element level

See attached PDF.

Appendix H – Research paper "Design of a Supply Chain Coordination System-of-systems"

See attached PDF.

Design of a Supply Chain Coordination System-of-Systems

Ir. G.J.W. Frijters
Dept. Maritime and Transport Technology
Delft University of Technology
Delft, The Netherlands
g.j.frijters@gmail.com

Dr. Ir. W.W.A. Beelaerts van Blokland
Dept. Maritime and Transport Technology
Delft University of Technology
Delft, The Netherlands
d.l.schott@tudelft.nl

A. Beije, MSc
Director Blockchain Fieldlab B.V. (BlockLab), co-author of "Blockchain and the supply chain"
Rotterdam, The Netherlands
aljojsa@blocklab.nl

Dr. Ir. D. Schott
Sect. .....Dept. Maritime and Transport Technology
Delft University of Technology
Delft, The Netherlands
w.w.a.beelaertsvanblokland@tudelft.nl

Dr. J.M. Vleugel
Dept. Transport and Planning
Delft University of Technology
Delft, The Netherlands
j.m.vleugel@tudelft.nl

***Abstract*—The renewable energy targets agreed upon in the Paris Agreement and European Green Deal lead to an enormous expected growth of currently installed offshore wind energy production capacity in European waters. Over a tenfold of the current capacity is planned to be installed before 2050. In support, due to efficiency, the size of offshore wind turbines also increases. Therefore, the number of offshore wind maintenance operations and their complexity is expected to grow. Asset managers are burdened with the complex manual task of finding supply for the increasing maintenance demand, to ensure performance of the wind power park and compliance with power purchase agreements. The current state matching and contracting of suppliers is a manually executed step in a series of automated maintenance planning steps, that requires the processing of commercially sensitive information. Manual execution is determined to be a bottleneck for effectively accommodating the increased maintenance demand and complexity. Solution quality, process lead time and process labor are expected to be improved when automating the human information processor. This paper describes the design process of an automated decentralized supply chain planner, that transforms the maintenance schedule into a feasible, definitive maintenance planning, by matching and contracting suppliers for scheduled maintenance operations and determine its definitive date, while By following a design approach based on the ABCDE – agile blockchain Dapp engineering method, merged with Baseline Protocol design features, a system-of-systems architecture and automated system-of-systems Workflow was designed, verified and validated.**

Keywords—Offshore wind maintenance, decentralized supply chain planning, blockchain technology, Baseline Protocol, system-of-systems, ABCDE – agile blockchain Dapp engineering

I. Introduction

The European energy system is undergoing a transition supporting the fulfilment of the objectives of the Paris Agreement and the European Green Deal. One of the key elements in these climate change combatting policies is the complete decarbonization of the energy sector. Offshore wind power is one of the attractive renewable energy sources for replacement of the current polluting sources. In 2019, 20 GW of energy production capacity was installed in European waters, which is projected to grow more than a tenfold for 2050, targets varying between 230 and 450 GW [1]. Additionally, wind experts forecast continued evolutionary growth in average offshore wind turbine (OWT) size, because it helps lowering the cost of wind energy [2]. A low cost of wind energy is important for the renewable energy transition.

Due to the growing OWT size, larger and more specific assets are required for offshore maintenance operations [3], while simultaneously the demand for these maintenance operations increases with the installed energy production capacity. The growth trends combined lead to an increase of multi-party maintenance operations, that need to be planned.

The current maintenance strategy of choice for the remotely located OWTs is sensory-based predictive maintenance, because its goal is to optimise the energy production plan and economic maintenance plan [3]. The predictive maintenance strategy is an advanced and highly automated strategy, that includes the automated scheduling. Based on the outcome of weather and asset health prediction models, fed with actual weather and condition monitoring data, maintenance schedules are automatically generated on a daily basis [4]. Afterwards, the manual execution takes over, where the asset manager analyses the maintenance schedule and starts searching for suppliers of assets and resources for each requirement per maintenance operation, by phone or email [4]. Depending on the configuration of an offshore wind power park (WPP), maintenance assets and resources are supplied by the WPP itself, a shared pool of nearby WPPs, or by an external supplier, that all are part of the offshore wind maintenance supply chain. Regardless of the supplier, an asset manager has to manually communicate over the availability and cost information of each maintenance requirement to ensure feasibility of the scheduled maintenance operation. A high-level view of the maintenance organisation activities and the level of process automation is illustrated in figure 1 on the next page.

A. Problem Definition

Manual matching and contracting of suppliers for each requirement for each scheduled maintenance operation is believed to be a bottleneck in accommodating the huge

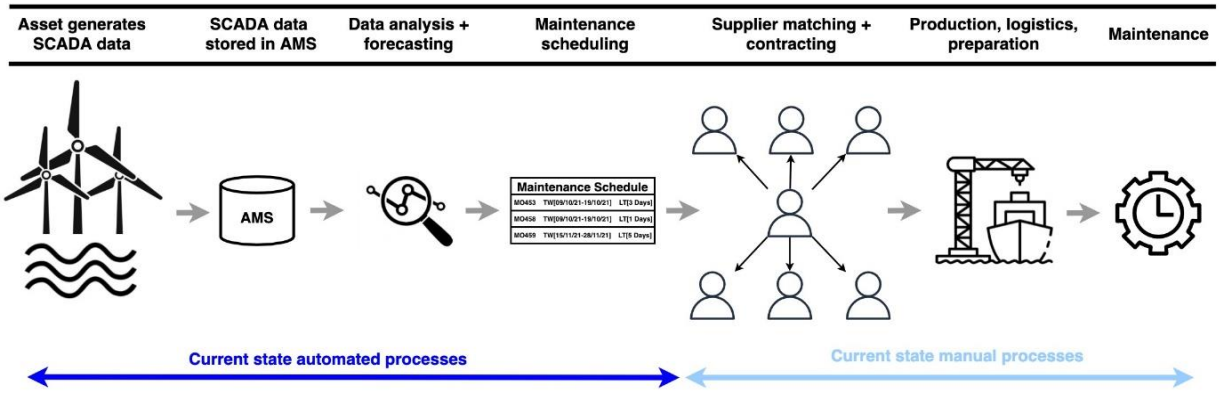


Figure 1: High-level view of OWT maintenance organization activities, with indication of automation level

expected increase of demand for multi-party maintenance operations, while maintaining the WPP's operational performance. The average number of human interactions per operation, and the total number of operations increases, while available lead time remains constant. Speed and capacity limitations of the human information processor are expected to lead to sub-optimal solutions for the energy production plan and economic maintenance plan. Therefore, the solution is searched in digital automation of the supplier matching and contracting process, similar to the preceding maintenance organisation activities.

The difficulty with matching and contracting of suppliers, is that it requires the processing of commercially sensitive information, i.e. maintenance demand schedules, supplier asset or resource availability schedules and their cost rates. These schedules are typically stored in protected company ERP systems. Company security officers are not keen on automating multi-party supply chain processes on the basis of exchanging commercially sensitive information from their protected ERP systems. The problem is that there is a lack of a system of connected systems that enables trustworthy processing of private data to automate the final stage of maintenance operation planning; the matching and contracting of suppliers for maintenance operations.

B. Envisioned Solution

The conventional solution is to design a centralised system, owned by a dominant supply chain participant or external service provider, that hosts the system on their own, or in a cloud based infrastructure. This is considered undesirable for a number of reasons:

- Single point of failure for system that supports multi-party supply chain operations.
- Control over automated supply chain process for single dominant supply chain participant, cloud provider, or external service provider.
- Potential access to an abundance of commercially sensitive supply chain data for the parties that own and host the system.
- Numerous system integrations to setup and maintain between suppliers and every WPP that develops their own centralised system.

Because of forementioned reasons, the envisioned solution shall include the use of an innovative technology known for its ability to automatically achieve decentralised consensus over a shared object, between distrusting systems in a public peer-to-peer network; blockchain technology [5]. Since the emergence of Bitcoin in 2008, blockchain technology has gained popularity due to its unique capabilities as a decentralised digital currency. Ethereum, a second generation blockchain, allows for deployment of automatically enforced programmable contracts called "smart contracts" [6]. Smart contracts combine protocols with user interfaces to formalize and secure algorithmically specifiable relationships over computer networks [7]. Meaning business logic can be programmed into, and enforced by smart contracts.

The applicability of blockchain technology to supply chain processes is covered in the work of [8]. By removing the single point of failure and the presence of a central authority, supply chain resilience increases. Public blockchain networks offer data security and cost-effective transmission of transactions in peer-to-peer networks, allowing for single, one time system integration for direct business-to-business interaction. Public blockchains are transparent and publicly accessible, therefore increasing the supply chain visibility and traceability. And, via the use of smart contracts real-time settlement and automation of the information and financial flow is enabled. The resulting efficient and effective data flow is proven to be essential for efficient supply chain coordination and responsiveness [8].

Not only in theory is the applicability of blockchain technology to supply chain processes validated, also in practise. Successful examples such as Naviporta [9], TradeLens [10] and the North-America Coca-Cola bottling supply chain [11] all show that an effective application of blockchain can reduce process lead times and labour costs, specifically for supply chain coordination processes that involve a high amount of intercompany communication and data processing.

Taking the features of blockchain technology into consideration, the envisioned solution is expected to effectively apply the technology by:

- using the public network as a single-integration common frame of reference for connected supply chain systems;
- using smart contracts to enable decentralised supply chain process automation between connected systems.

Assumed is that the commercially sensitive maintenance demand and maintenance supply schedules of all participants in the offshore wind maintenance supply chain are stored in

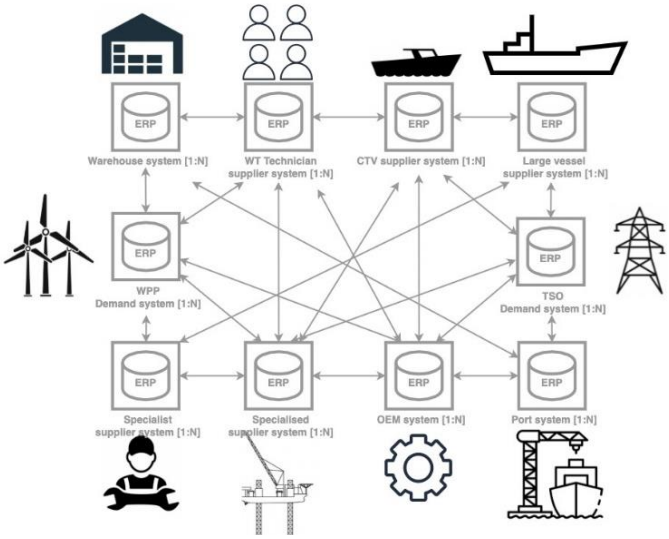


Figure 2: Visualization of envisioned peer-to-peer supply chain system-of-systems interaction for maintenance supply chain coordination.

company ERP systems. By designing a system that connects these systems and force them to cooperate in an automated data exchange workflow, while remaining the data privacy, most of the manual coordination can be eliminated for each scheduled maintenance operation. This is expected to result in less manual communication for the entire maintenance supply chain, less manual coordination and associated data processing for the WPPs, and faster secured and more cost-effective supply for the maintenance demand. Therefore, the envisioned solution (see figure 2) helps the WPPs maintain their operational performance by finding maintenance supply for rapidly increasing demand while respecting the energy production optimised maintenance schedule.

Because business managers are still hesitant to adopt blockchain based enterprise solutions, technical feasibility and data privacy for the developed design is considered important. The main research question that is answered in this research is determined to be:

How to design a technical feasible decentralized system-of-systems that enables automated matching and contracting of maintenance supply for scheduled demand through privacy preserving processing of commercially sensitive data?

II. Research Methodology

This research is approached from a system engineering perspective. Therefore, to answer the research question, first a literature analysis is performed into the subjects of systems-of-systems theory, blockchain technology and systems engineering. The result of the analysis is used to determine a definition framework and a design approach to create a technical feasible design. The design shall be validated via a case study, which is explained in a separate section.

A. Literature analysis

The Delft Systems Approach was developed as an extension to existing systems theory, which lacked the perspective of the researcher [12]. The Delft System Approach provides tools to identify, define and describe parts and properties of a system. A system is composed of elements, which are the smallest identifiable parts. The interaction between elements is referred to as relationships. Subsystems are partial collections of elements whereby all the original relationships between these elements remains unchanged. A system can have a state, which is the value of all its properties at a given time. An event occurs when the value of a property of an element changes. And when one event leads inevitability to other events, this is referred to as activity.

Maier was the first researcher in 1998 to examine, the then commonly used term, system-of-systems and the meaning of it in detail [13]. He defines collaboratively integrated systems as "systems-of-systems" (s-o-s), given it includes two distinguishable characteristics:

A system-of-systems is an assemblage of components which individually may be regarded as systems, and which possesses two additional properties:

- *If disassembled, the component systems must be able to usefully operate independently.*
- *Managerial independence of component systems, meaning they are separately acquired, integrated, and maintained.*

Moreover, Maier identified four architectural principles to give the definition more body. The first principle is the principle of stable intermediate forms for component systems, that originated from civil engineering. The second principle is policy triage, which forces to think very carefully what to control in an s-o-s design, while respecting the s-o-s properties. The third principle is leverage at the interfaces, meaning to focus on the interfaces between the operational and managerial independent components. The last principle is ensuring cooperation between the component systems of an s-o-s. Definitions for each granularity level of s-o-s components can be captured in a pyramid, as illustrated in figure 3. S-o-s are made out of systems, that consist of subsystems, that consist of elements.

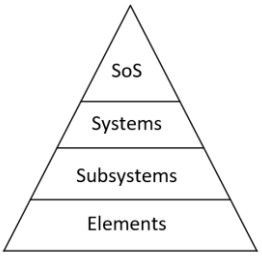


Figure 3: Definitions for describing s-o-s components at different granularity levels

The second subject of literature analysis is blockchain technology. The essentials and applicability to supply chain processes are already given, so remaining relevant features and projects are explained here.

For this research it is important to know that a blockchain technology consists of a peer-to-peer network of nodes that all possess a ledger, also they possess the tools to gain consensus among nodes, in a decentralized manner, about the actual state of the ledger. The ledger can be regarded as a record of which blockchain addresses own which digital tokens. Other than that, smart contracts are also represented by a blockchain address and can fed with data via a

blockchain transaction. The smart contract then automatically executes the programmed set of rules and the broadcasted output will be used to update the ledger accordingly. On a public network, transactions and the state of the ledger is fully transparent. Transactions are captured in blocks, which are added to an immutable append-only chain of blocks. Once deployed on the network, smart contracts are just as immutable except for what the programmed functionalities allow for.

While analyzing literature on blockchain technology, the Baseline Protocol was encountered. Set up as an Oasis open-source project, the Baseline Protocol (BP) is combines advances in cryptography, messaging, blockchain technology to deliver secure and private business processes, event ordering, data consistency, and workflow integrity at low cost [14]. It is a middleware that connects ERP systems to a common frame of reference; the public blockchain network. In the BP, the following elements are included; an ERP connector, privacy service to work with zkSNARKS, peer-to-peer messenger service, vault service to digitally sign documents and a blockchain client to communicate with a blockchain network. BP elements placed on the blockchain are: a Shield smart contract that acts as gatekeeper and workflow state synchronizer, a OrgRegistry smart contract that holds a list of involved supply chain participants and their blockchain addresses, and a Verifier smart contract that is able to verify received zero-knowledge proofs according zkSNARK transformed business logic. With these elements, BP synchronizes connected systems to the current state of a workflow (shared business process) they execute together. A connected system creates a zero-knowledge proof out of the data of an executed workstep, and send the proof to the Shield contract. The Shield contract verifies if the sending address is listed in the OrgRegistry, and if so it forwards the proof to the Verifier contract. Once the proof is verified, the Shield contract emits a successful event and broadcasts a Commit to the connected systems, that are now enabled to update their state of the workflow. A Commit is a hash combination of data that proves the execution of a workstep, that is used as message to update the state of the workflow in the network.

Zero-knowledge proofs (ZKP) are a family of probalistic protocols, first described in 1989[15]. They are defined as proofs that convey no additional knowledge other than the correctness of the proposition in question, which makes them extremely useful to perform computations on sensitive data on a public network. One particular family of ZKP is described as zero-knowledge succinct non-interactive arguments of knowledge, a.k.a. zkSNARKS. It is an efficient application of ZKP, particularly useful in systems where running computations is costly. They are non-interactive because it allows for a verifying party to include the verification scheme into, for example, a smart contract that is deployed on the blockchain network [16]. A proving party can at any time upload a ZKP to verify its correctness and compliance to the programmed scheme. The Ethereum blockchain is one of the early adoptors of zkSNARKS, and ZoKrates is the toolbox that shall be used to develop these data privacy preserving verification schemes [16].

The third and final subject analysed in literature is systems engineering, with the goal to provide the necessary design methods for developing an s-o-s. First was looked at the widely adopted V-model, proposed by Rook in 1986 [17]. Although originally created for software development, the model was and is applied for the development of high-end systems in any domain. It's a comprehensive design method that addresses the entire lifecycle of an engineered system, including maintenance and product phase-out. For this research a more agile and suitable implementation of the V-model was searched for, that predominantly focusses on the design phase of the system.

During analysis, the Agile Block Chain Dapp Engineering (ABCDE) method was encountered. Created by Marchesi et al. [18], the method was specifically developed out of a lack of disciplined, organized and mature development processes for blockchain based products. Agile practices were included to cope with misunderstood or changing system requirements, which can be the case for developing novel system designs as in this research. The iterative and incremental approach of Scrum is also included in the method, to speed up the development and increase the quality of the design. According the method first, the goal of the system, the actors and user stories are defined. Then the design phase is split into development of first the blockchain parts, and second the non-blockchain ("off-chain") parts. Those parts are programmed and tested, and are integrated in a final design synthesis. All of the design steps in the ABCDE method are similar to first steps until the maintenance phase as described in the V-model.

B. System-of-systems theory extension

Via the literature analysis it was discovered that an additional system part distinction can be made for information processing s-o-s. According the system and s-o-s theory, the definition "Elements" is used to described the smallest parts of a system. However, standardized, passive objects of data consumed by the other elements in the s-o-s, can be distinguished from this group. These parts shall be defined as "Data objects" and are vital to successful operation of an information s-o-s, and therefore also need to be designed separately. In the context of this research the data objects are the maintenance and availability schedule templates, Commit hashes, blockchain addresses, message structures and other data structures (e.g. JSON schemes). Not only in information s-o-s data objects can be distinguished from the elements, in physical s-o-s data objects appear in the form of serial numbers, tags, stickers, and general signaling etcetera. The definition is therefore considered to be generic applicable to all kinds of s-o-s and is proposed to be added as an extra definition granularity level for describing parts of a s-o-s, illustrated in figure 4.

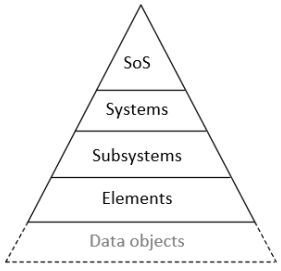


Figure 4: Proposed additional definition granularity layer "Data objects" for describing system-of-systems parts

C. Design approach

Via the literature analysis, an approach is determined to develop the design of the envisioned system-of-systems. The ABCDE method will be the main design method, complemented with design features of the Baseline Protocol. The merged design method is determined to be:

1. Define goal of s-o-s (ABCDE step 1)
2. Identify actors (ABCDE step 2)
3. Define initial s-o-s architecture (BP design feature)
4. Define user stories (ABCDE step 3)
5. Define initial workflow (BP design feature)
6. Split design on blockchain (ABCDE step 4)
7. Design blockchain system (ABCDE step 5 and 6)
8. Design off-chain systems (ABCDE step 7 and 8)
9. S-o-s design integration (ABCDE step 9)

The scope of the design is an element level s-o-s architecture design and a shared workflow design executed by elements of the component systems, both in UML format. The envisioned supply chain coordination process involves three distinguishable systems; the maintenance demand system in possession of the maintenance schedule, the maintenance supply system in possession of the availability schedules, and the blockchain system that enforces the two other systems to cooperate via the automated workflow. Therefore, the three component systems of the s-o-s that are separately designed are:

- (Maintenance) Demand system
- (Maintenance) Supplier system
- Blockchain system

The determined design scope can be captured in the definition pyramid for s-o-s (see figure 4). Since the scope includes three component systems, the definition pyramid becomes a three-sided pyramid where each side represents one system. The data objects consuming workflow that forces s-o-s elements to cooperate, can be visualized in the heart of the pyramid on element level. For each of the component systems; subsystems are defined, elements are defined and designed, and data objects are defined and designed. The complete design scope is visualized in figure 5.

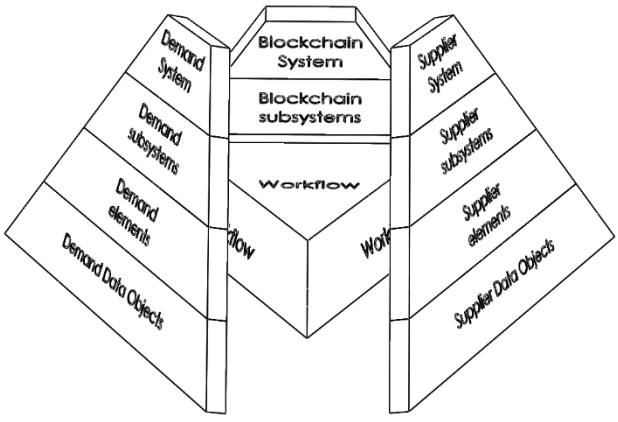


Figure 5: Visualization of design scope for envisioned s-o-s

Verification of the design is done via four methods. First, by teaming up with a blockchain developer at BlockLab and programming essential functionalities, technical feasibility of the design is ensured. Second, lead developers and system architects of the Baseline Protocol are regularly consulted during the design process resulting in design iterations. Third, an assessment on the realization of defined user stories is performed. Fourth and last, compliance to s-o-s properties and architectural principles is assessed.

The design is validated via a case study on coordination process lead time and labor time. Lead time is important because fast maintenance mobilization limits OWT downtime, and it is considered an indication of process efficiency. Process labor time is important as a measure of automation level, and potential operational cost reductions. How the impact of developed design on these two parameters is measured is explained in the next section.

D. Case study

The business case selected for this case study is coordination of the annual maintenance demand for the group of four WPPs at Dogger Bank, namely Creyke Beck A and B, and Teesside A and B. Together they consist of 1000 OWTs. The supply chain size and maintenance demand, that determines the amount of communication and information processing for maintenance coordination, is based on the work of Steendijk and Beelaerts van Blokland [19]. The annual failure rates that generate the scheduled maintenance demand is based on the research of Carroll et al. [20], resulting in 6178 minor failures, 1062 major failures and 264 major replacements for the 1000 OWTs. Due to the large distance to shore, each of the four WPPs owns a fleet of large, small and crane vessels, a warehouse for high-frequent spare-parts and employs teams of OWT technicians. To complete the case study, the following assumptions are made:

- WPPs let other WPPs charter their vessels.
- WPPs outsource their OWT technicians to each other.
- WPPs sell each other spare-parts from their warehouses.
- For any MO, always one supplier will be able to supply any MO requirement in order to fulfil the demand.
- All possible MO required assets and resources are registered in ERP systems.

Furthermore, it's assumed that there are 3 original equipment manufacturers (OEM) and 5 ports located in the Dogger Bank service area. An overview of the assumed supply chain size is given in table 1.

Type of MO	MO requirements	No. of suppliers
Minor failure	Small vessel (e.g. CTV)	4
	Team of OWT technicians	4
	OWT spare-parts	4
Major failure	Small vessel (e.g. CTV)	4
	Team of OWT technicians	4
	OWT spare-parts	4
	Large vessel (e.g. SOV)	4
	OEM spare-part	3
	Port of loading/unloading	5
Major replacement	Small vessel (e.g. CTV)	4
	Team of OWT technicians	4
	OWT spare-parts	4
	Large vessel (e.g. SOV)	4
	OEM spare-part	3
	Port of loading/unloading	5
	Team of OEM technicians	3
	Crane vessel (e.g. HLV)	4

Table 1: Assumed requirements per type of MO and assumed no. of available suppliers in service area.

To assess the impact of the designed s-o-s, formulas to calculate the process lead time and process labor time for each type of MO are derived for both the current state manual coordination process and the designed automated workflow. Values for coordination process activity lead times come from two undisclosed internal process analyses performed at BlockLab. One analysis measures the process activity lead times of the communication and information processing for international shipment of a logistics service provider. The other analysis measured similar activity lead times for an NL to UK import process. One constant that has to be assumed for these formulas is the supplier response time, which is set to 24 hours. Process activities that are completely automated by the developed design are assumed to have no lead time and labor time.

III. Findings

Implementation of the described design approach led to research findings that are explained in this section. The first finding is the extension of the s-o-s theory with an additional definition for describing and distinguishing system parts, data objects. The second finding is the scoped set of system and s-o-s designs, including blockchain, demand and supplier system designs, and the final s-o-s architecture and s-o-s workflow design. The last finding is the impact of the designed s-o-s on lead time and amount of labor spent on the coordination process.

A. System-of-systems theory extension

As explained earlier in section II-B, via this research it was discovered that an additional layer of definition granularity is needed to properly describe information s-o-s, and to some extend s-o-s in general. “Data objects” are the smallest, standardized parts in an s-o-s, consumed by other s-o-s elements. They’re vital to its operational success. For this research the data objects are the maintenance and availability schedule templates, Commit hashes, blockchain addresses, and other data structures (e.g. JSON schemes). In physical systems, they appear as serial numbers, stickers, tags, and general signaling. Some examples are; numbered buttons on a remote in a home entertainment s-o-s, or road signs in transport s-o-s, and the aisle number markings in a warehousing s-o-s.

B. System design: Blockchain System

According the applied design method, the Blockchain System is the first component system to be designed. Its main goal is to securely synchronize the connected demand and supplier systems to the actual state of the workflow that is initiated for every MO. Second, it protects the Supplier Systems from Demand System data harvesting.

The final design for the Blockchain System, achieved through 5 documented design iterations, is illustrated in figure 6. What specifically is designed for the Blockchain System is the element Verifier contract, and the data objects Commits. The function of each Commit is explained in the final s-o-s design.

The design exploits the existing Ethereum blockchain infrastructure, which means it is a pay-per-use, always accessible and fully decentralized infrastructure that securely synchronizes the Demand and Supplier Systems. The synchronization mechanism is adopted from the BP, and works as follows:

1. One of the connected systems in the network executed a workstep in the workflow on their system, and synchronizes the network. The system creates a verifiable ZKProof of the data that proofs the execution of the workstep, and the Commit message that is used to synchronize the other systems in the network. Both are sent to the Shield contract.
2. Upon receival of the ZKProof and the Commit, the Shield contract verifies if the sending address is listed as supply chain participant in the OrgRegistry contract
3. If so, the Shield contract forwards the ZKProof to the Verifier contract.
4. The Verifier contract verifies the ZKProof for correct formulation of hashed workstep data in privacy preserving manner (detailed program in Appendix A). If correct, the Verifier contract emits a verified event to the Shield contract.
5. The Shield contract, where each of the connected systems is listening to, broadcasts the Commit throughout the network. Each system stores the Commit in their respective blockchain database. Now the entire supply chain network is validly synchronized to the actual state of the workflow, in a secure, data privacy preserving manner.

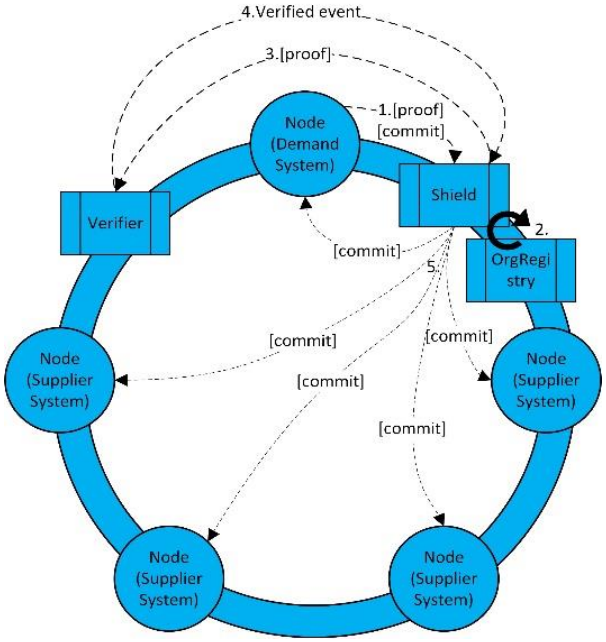


Figure 6: Blockchain system design illustrated via an arbitrary supply chain network of demand and supplier systems

C. System design: Demand System

The Demand System is designed for the asset managers searching for supply for scheduled maintenance demand. The goal of the Demand System is to automatically match suppliers to each requirement of scheduled MOs and to allow for standard business procedure of proposing and negotiating business agreements between two supply chain participants, while remaining synchronized to running workflows in the s-o-s. The final design, illustrated in figure 7, was reached

through 3 documented design iterations. Specifically designed for the Demand System are the elements MO Extraction Module, Matching Module, Commit Generator, Data Formatter, and Workflow Database. The designed data objects vital to the operational success of the system are the maintenance schedule template, MO objects for efficient distribution of MO related data. The remaining elements are adopted from BP.

The Demand System achieves its goal through multiple steps. First, it automatically links suitable suppliers in the ERP system to each MO requirement for each MO in the maintenance schedule. Second, it automatically sends out availability and cost requests for the given time window and MO lead time, via the Messenger Service to the suitable suppliers. Third, supplier availability overlap and determination of the earliest day of feasible MO execution is calculated from the received availabilities. Feasible MO execution means at least one supplier for each MO requirement is available. Based on the earliest day, all possible supplier subsets are generated and according the set WPP optimization preferences, an optimal supplier subset is selected. Fourth, for the selected suppliers, the Demand System automatically prepares business proposals that only have to be reviewed and signed by the asset manager before distribution. Once all required suppliers have signed the busines proposal, all suppliers are contracted for the particular MO and is ready for execution. The Demand system is operated through existing company ERP systems and is able to support multiple workflow execution.

D. System design: Supplier System

The Supplier System is designed for the asset, resource and product planners in the maintenance supply chain. The goal of the Supplier System is to allow for assets, resources or products to be automatically matched to MO requirements and to allow for standard business procedure of negotiation and acceptance of business proposal.

The final design, illustrated in figure 8, was achieved without any design iteration. Because most elements could be adopted from previous designs, only the Availability Module had to be specifically designed.

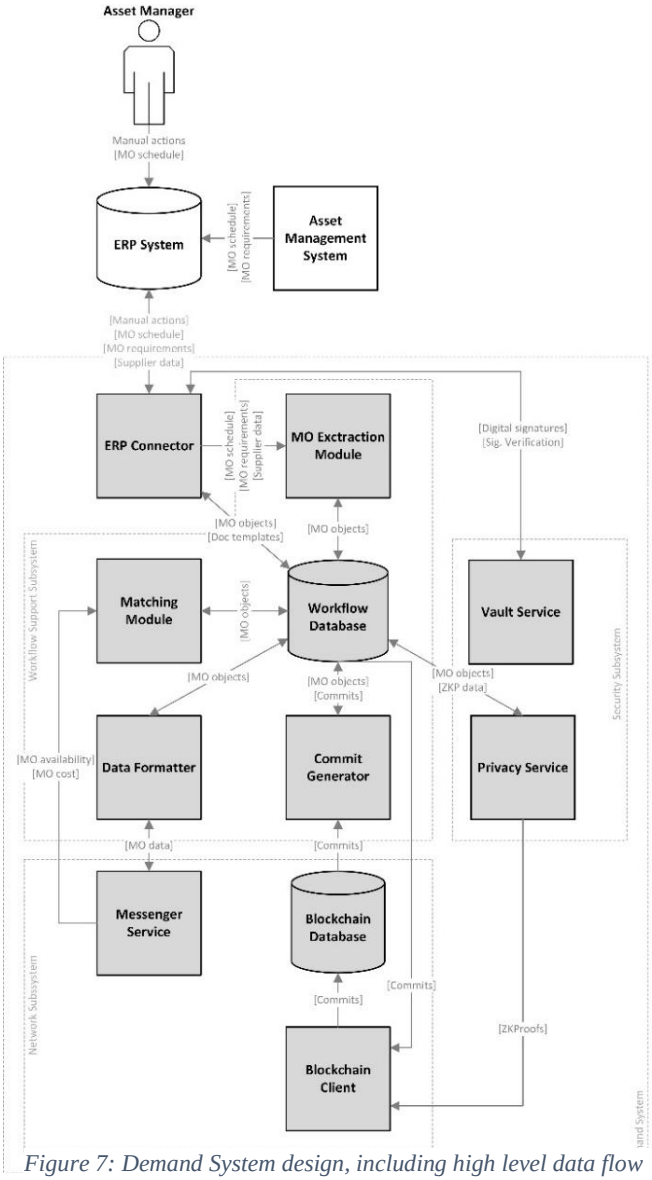


Figure 7: Demand System design, including high level data flow and subsystem definition

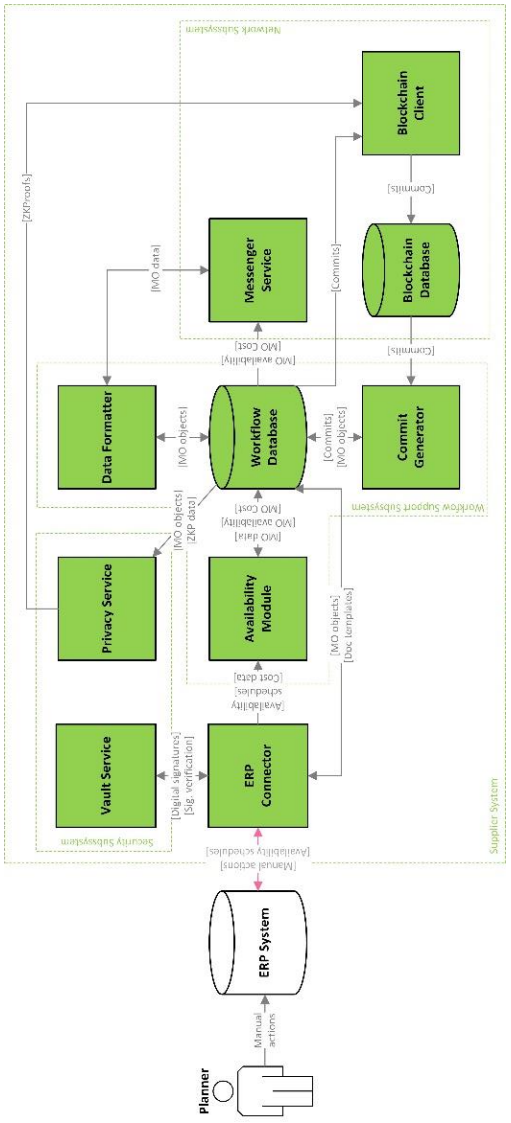


Figure 8: Supplier System design, including high level data flow and subsystem definition

The Supplier System achieves its goal via multiple steps. First, after receiving an availability request via the Messenger Service, the system takes the associated Commit in the Blockchain Database and compares it to a Commit generated from data in the received MO Object. If equal, the Supplier System has received a validated request and allows the Availability Module to calculate and return the asset/resource/product availability for given time window and MO lead time, from the availability schedule in the ERP system. If the supplier eventually gets selected by the Demand System, the system allows for standard business procedure negotiation of the terms of received business proposals, where every proposal or agreement is registered as Commit on the blockchain. The Supplier System is protected from free data harvesting because for every executable availability request, first a Commit has to be published on the blockchain. Commits are only published after successful verification of the ZKProof, which costs about 1.6 million gas, or 8 times the cost of a regular token transaction on Ethereum [16].

E. S-o-S Design: Architecture and Workflow

The three designed systems; Blockchain System, Demand System and Supplier System, are integrated in the final s-o-s architecture design as illustrated in figure 9. The s-o-s allows for multiple Demand and Supplier Systems, collaborating simultaneously on different uniform workflows, with the goal to automatically match and contract suppliers to every MO requirement in every scheduled MO. The Blockchain System enforces collaboration between all connected Demand and Supplier systems on the workflows, and controls the legitimate synchronization via publishing Commits on the network. The underlying message and data of each Commit reaches the right systems via the Messenger Service peer-to-peer network. The only manual interaction in the s-o-s is giving confirmation on the selected supplier subset, and reviewing and digitally signing of business proposals, all other manual communication and information processing is done by the s-o-s.

The final uniform workflow that is executed by various combinations of systems of supply chain participants, for the matching of MO demand with supply, is explained on a high-level in table 2 on next page. A detailed workflow, on element level can be found in TU Delft repository.

F. Case Study

As explained earlier, through a case study the developed design can be validated. Its performance in terms of process lead time and amount of process labor is compared to the current state manual coordination. The determined formulas for the key performance indicators are:

Process lead time:

- *LT_cur_state = 2903 + 15*N_suit + 6*N_req [min]*
- *LT_sos_design = 1442 + N_req [min]*

Amount of process labor:

- *PL_cur_state = 23 + 15*N_suit + 6*N_req [min]*
- *PL_sos_design = 2 + N_req [min]*

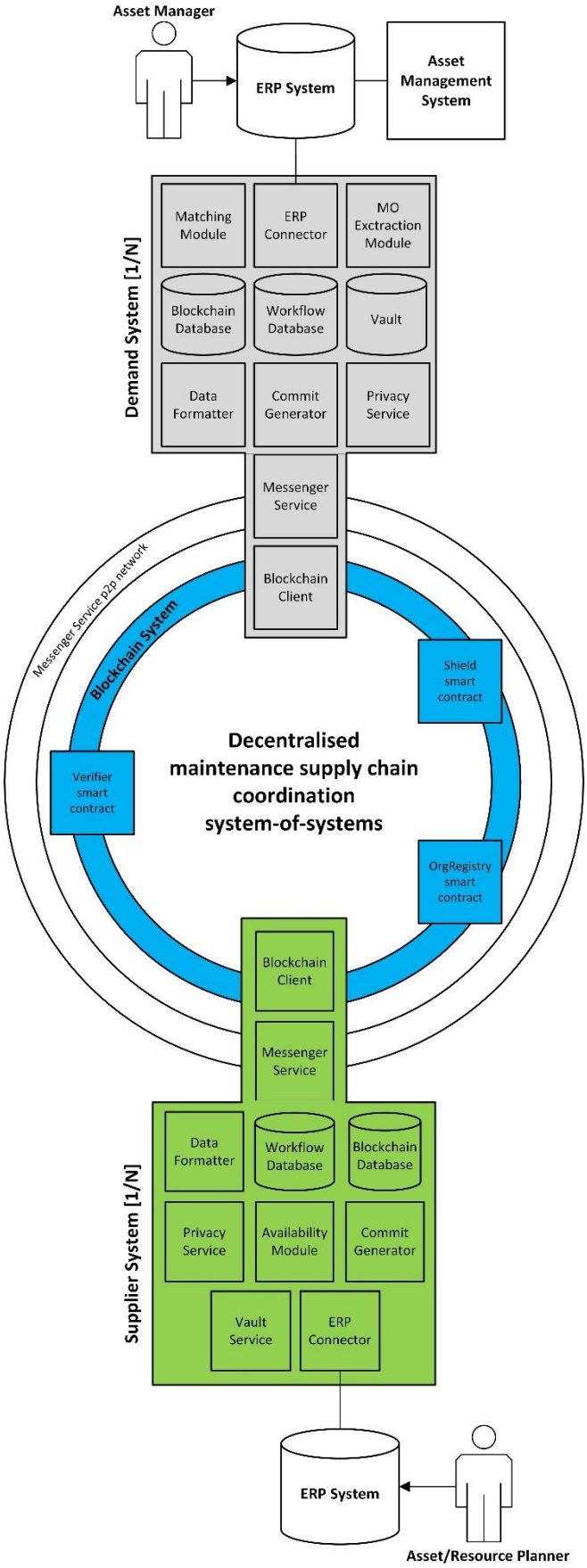


Figure 9: Final supply chain coordination s-o-s architecture design. Colored items are the designed systems according design scope

W.S.	Commit	P2P message	Explanation
1	Genesis	Availability request	After linking suppliers to each MO requirement for a particular MO, a Demand System first creates a Genesis Commit, whereafter the availability request follows to all linked suppliers. Commit is also protection against data harvesting. This marks the beginning of the workflow.
2	Selection	Supplier selection	Once asset manager confirms selected optimal supplier subset, a Selection Commit is published as an incentive for each requested supplier to either block/unblock their asset/resource availability.
3	Business proposals	Business proposal	Once asset manager has reviewed and signed business proposals for every supplier for the MO, each of the proposals is notarized on the blockchain via this Commit
4	Business agreements	Business agreement	Once a supplier reviewed and signed a business proposal, the resulting business agreement is notarized on the blockchain via this Commit
5	Finalization / cancelation	Finalization / Cancelation	Once all suppliers are succesfully contracted, a Final Commit is published so that each supplier knows preparation for MO can begin. Likewise, if one participant publishes a Cancelation Commit all participants know that the MO is (temporary) cancelled.

Table 2: High-level workflow design, in worksteps with related Commits and messages

Where,*N_suit* is the total number of suitable suppliers capable to supply for the particular MO, and *N_req* is the required number of suppliers for a feasible MO execution.

Based on assumed supply chain size of the Dogger Bank, the following results are obtained from the case study.

Case study results: process lead time			
MO type	**Current state [min]**	**S-o-s design [min]**	**Reduction**
Minor failure	3101	1445	53.4%
Major failure	3299	1448	56.1%
Major replacements	3416	1450	57.6%

Table 3: Case study results on coordination process lead time

Case study results: amount of process labor			
MO type	**Current state [min]**	**S-o-s design [min]**	**Reduction**
Minor failure	221	5	97.7%
Major failure	419	8	98.1%
Major replacements	536	10	98.1%

Table 4: Case study results on amount of process labor

IV. Conclusion

For the conclusion of this research, the defined research question is revisited:

How to design a technical feasible decentralized system-of-systems that enables automated matching and contracting of maintenance supply for scheduled demand through privacy preserving processing of commercially sensitive data?

Mentioned s-o-s is designed with the merged design approach as described in section II-C, which consists of the ABCDE method complemented with the two Baseline Protocol design features, and guided by architectural principles from system-of-systems theory. For the developed designs, existing Ethereum blockchain infrastructure, BP elements and their synchronization mechanism were adopted. To fulfil the goal of the s-o-s and component Blockchain, Demand and Supplier Systems; additional elements and data objects had to be designed. For the ability to define and describe all vital parts of the s-o-s, the systems-of-systems theory definition framework was extended with an additional definition for passive, standardized, pieces of information consumed by the other s-o-s parts. These pieces of information are defined as “Data Objects” and are vital to the operational success of an s-o-s. This is also the academic contribution of this research, next to the novel generic decentralized s-o-s design for supply chain coordination.

Technical feasibility was ensured by programming and testing of essential functionalities. Automated matching and contracting of supply is enabled through the Blockchain System enforced workflow. Privacy preserving processing of commercially sensitive data is achieved through the following:

- All sensitive data processing happens on client side, after which only minimal disclosing results are returned directly via a p2p messenger service.
- Sensitive data as input for the Blockchain System is also first cryptographically transformed in either a ZKProof or a Commit hash, before the data leaves the system.

Design verification was performed through regular consults with lead developers and solution architects of BP that led to the 8 documented design iterations, and assessment of compliance with defined user stories and s-o-s architectural principles. For the user stories, 9/17 were technically enabled, 7/17 theoretically enabled and 1/17 not enabled. Regarding architectural principles, all four were met. Stable intermediate forms are the component systems, policy triage is applied by only adding complementing elements, leverage is applied at the interfaces because most effort went into the Blockchain System design, that also ensures cooperation together with the workflow.

Design validation was performed through a case study based on the maintenance supply chain of four combined WPPs at the Dogger Bank. On average, coordination process lead time was reduced by 56% by designed s-o-s, which is beneficial for increasing maintenance demand and OWT down time. The amount of process labor was reduced by 98% on average, which is beneficial for solution quality and for overall operational performance.

V. Appendices

A. Proof of correct formulation program as input for zero-knowledge proof mechanism used in Verifier contract

```
import "hashes/pedersen/512bit" as pd

import "utils/pack/u32/pack128" as pack128

import "utils/pack/u32/unpack128" as unpack128


def main(private field State, private field MJ_ID,
private field SupplierID, private field DocHash1,
private field DocHash2, private field ContractH1,
private field ContractH2, private field LC1,
private field LC2, private field NC1, private field
NC2) -> (field, field):

        field a = if (State == 1 || State == 2 ||
        State == 5) && ContractH1 == State &&
        ContractH2 == State && SupplierID == State
        && DocHash1 == State && DocHash2 == State
        then 0 else 1 fi

        field b = a * ContractH1

        field c = a * ContractH2

        field d = a * SupplierID

        field e = a * DocHash1

        field f = a * DocHash2


        field g = if (State == 3 || State == 4) &&
        DocHash1 != ContractH1 && DocHash2 !=
        ContractH2 then 1 else 0 fi

        field h = g * b

        field k = g * c


        u32[16] preHash1 = [...unpack128(State),
        ...unpack128(MJ_ID), ...unpack128(d),
        ...unpack128(LC1)]

        u32[16] preHash2 = [...unpack128(e),
        ...unpack128(f), ...unpack128(h),
        ...unpack128(LC2)]


        u32[8] Hash1P = pd(preHash1)

        u32[8] Hash2P = pd(preHash2)


        u32[16] NewHash = [...Hash1P[0..4],
        ...Hash1P[4..8], ...Hash2P[0..4],
        ...Hash1P[4..8]]


        u32[8] NewHashP = pd(NewHash)

        field[2] NewHashField =
        [pack128([...NewHashP[0..4]]),
        pack128([...NewHashP[4..8]])]


        assert(NewHashField[0] == NC1)

        assert(NewHashField[1] == NC2)


        return MJ_ID, State
```

VI. References

[1] BVG-Associates, “Our energy, our future,” 2019.

[2] R. Wiser, M. Hand, J. Seel, and B. Paulos, “Reducing Wind Energy Costs through Increased Turbine Size: Is the Sky the Limit? Berkeley Lab study,” 2016.

[3] Z. Ren, A. S. Verma, Y. Li, J. J. E. Teuwen, and Z. Jiang, “Offshore wind turbine operations and maintenance: A state-of-the-art review,” *Renew. Sustain. Energy Rev.*, vol. 144, no. March, 2021.

[4] C. Stock-Williams and S. K. Swamy, “Automated daily maintenance planning for offshore wind farms,” *Renew. Energy*, vol. 133, no. April, pp. 1393–1403, 2019.

[5] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008.

[6] V. Buterin, “A next-generation smart contract and decentralized application platform,” *Etherum*, no. January, pp. 1–36, 2014.

[7] N. Szabo, “Formalizing and securing relationships on public networks,” *First Monday*, vol. 2, no. 9, pp. 1–27, 1997.

[8] N. Vyas, A. Beije, and B. Krishnamachari, *Blockchain and the supply chain - concepts, strategies and practical applications*. 2019.

[9] Port of Rotterdam, “Singapore and Rotterdam successfully complete trial with electronic bill of lading,” *https://www.portofrotterdam.com/en/news-and-press-releases/singapore-and-rotterdam-successfully-complete-trial-with-electronic-bill-of.* .

[10] TradeLens, “An interconnected ecosystem of supply chain partners,” *https://www.tradelens.com/*. .

[11] Kyle Thomas and P. Services, “Baselining the North America Coca-Cola Bottling Supply Chain,” *https://provide.services/news/baselining-the-north-america-coca-cola-bottling-supply-chain*. .

[12] H. Veeke, J. Ottjes, and G. Lodewijks, *The Delft Systems Approach*, vol. 53, no. 9. 2008.

[13] M. W. Maier, “Architecting Principles for Systems-of-Systems,” *Syst. Eng.*, vol. 1, no. 4, 1999.

[14] Baseline Protocol Community, “Baseline Protocol,” *https://docs.baseline-protocol.org/*, 2020. .

[15] S. Goldwasser, S. Micali, and C. Rackoff, “The knowledge complexity of interactive proof systems,” *SIAM J. Comput.*, vol. 18, no. February, 1989, pp. 186–208, 1989.

[16] J. Eberhardt and S. Tai, “ZoKrates - Scalable Privacy-Preserving Off-Chain Computations,” *Information Systems Engineering, TU Berlin*. pp. 1–8, 2018.

[17] P. Rook, “Controlling Software Projects.,” *Softw. Eng. J.*, vol. 1, no. 1, pp. 7–16, 1986.

[18] L. Marchesi, M. Marchesi, and R. Tonelli, “ABCDE -Agile Block Chain dApp engineering,” *arXiv*, vol. 1, no. November, 2019.

[19] J. Steendijk and W. W. A. Beelaerts van Blokland, “Optimization of Maintenance Operations for Offshore Wind Farms,” in *Hamburg International Conference of Logistics (HICL) – 28*, 2019, no. September, pp. 55–82.

[20] J. Carroll, A. Mcdonald, and D. Mcmillan, “Failure Rate , Repair Time and Unscheduled O & M Cost Analysis of Offshore Wind Turbines,” *Wind Energy*, vol. 19, no. 6, p. 214, 2015.

Supplier System 1 to N
Messenger service
Blockchain Client
Mongo DB (Merkle Tree)
Vault
Commitment Generator
Availability Module
ERP System
WF System
Extraction module
Alignment and selection
Commitment generator
Privacy service
Vault
Mongo DB (Merkle Tree)
Blockchain client
Messenger service
Shield contract
Dvgbg contract
Verifier contract
Blockchain System
AMS
ERP SYSTEM
AVAILABILITY MODULE
COMMITMENT GENERATOR
VAULT
MERKLE TREE
BLOCKCHAIN CLIENT
MESSENGER SERVICE
EXTRACTION MODULE
ALIGNMENT AND SELECTION
PRIVACY SERVICE
SHIELD SERVICE
Dvgbg contract
Verifier contract
Page 1
Page 2
Page 3
Page 4