Proceedings of 15th International Marine Design Conference (IMDC-2024)
June 2-6, 2024
Amsterdam, the Netherlands

Closing the Gap between Early and Detailed Ship Design Models

Herbert J. Koelman[1,*], Bastiaan N. Veelo[1], Ludmila Seppälä[2] and Paul Filius[2]

ABSTRACT

Conventionally, ship design and engineering are segregated activities, carried out with different software packages that thus each have their own place, qualities and tools. And consequently, a different data model. As a report on ongoing work to bridge that gap, this paper first explores existing neutral data models and standards employed or considered in maritime applications and concludes that none of these is directly applicable. It continues with describing the requirements and derived abstract data model of the SEUS project and its design and engineering applications. A graph database is identified as a potentially useful tool for SEUS data modelling, and a hands-on experiment confirms this presumption.

KEY WORDS

Ship design methodology; Data models; Digital transition.

INTRODUCTION

In the ship design process, the transition from early conceptualisation to detailed design marks a critical juncture — a "tipping point" that involves a change of stakeholders. Early design models emphasize the vessel's fundamental characteristics, naval architectural and functional aspects. Detailed design and production design delve into the intricacies of layout and deal with the generation of production and construction data. This dichotomy in focus leads to disparities in the tools employed and necessitates the alignment of data models.

This paper explores data standards in maritime applications, data models of two particular software systems — PIAS (by SARC) and CADMATIC — and the direction for integrating these models in the SEUS[3] project. SEUS aims at creating a high Technological Readiness Level solution to provide a set of computational tools for shipbuilding, incorporating data flows while the design matures and providing a comprehensive toolset for different stakeholders of the overall shipbuilding process, with access to a single source of true data. However, conventional neutral models or industry standards have not generally demonstrated their suitability for this task, so the research question addressed in this paper is to find a data model for shipbuilding which provides coherence of the data generated along the life cycle. Critical for SEUS are its anticipated impacts — such as a) platform solution for PLM, b) facilitation of digital transformation and c) integration of early design into the overall design process, see Gaspar et al. (2023) —many of which require software integration between design and engineering. However, it is not the first time in the history of mankind that such an endeavor is undertaken, so in the next section the applicability of existing maritime product data standards is investigated.

DATA STANDARDS FOR MARITIME APPLICATION

The authors share a combined 100^{+} years of experience in developing maritime software, where aspects of interfacing and (software) collaboration have played an important role, from time to time. In that context, we have regularly been surprised

[1] SARC, Bussum, The Netherlands.
[2] CADMATIC, Turku, Finland.
* H.J.Koelman@sarc.nl.

[3] See acknowledgement at the end of this paper for SEUS project details.

Submitted: 13 February 2024, Revised: 8 April 2024, Accepted: 1 May 2024, Published: 19 May 2024

Conference paper, DOI: https://doi.org/10.59490/imdc.2024.837 e-ISSN: 3050-4864

by the surprise of others when we have been asked why we don't just use one of the many existing data exchange standards. By way of a (late) response, this will be addressed later in this section. First, however, a brief overview of potentially applicable standards is given, without seeking to be complete.

An Outline of Potentially Applicable Neutral Models and Data Standards

The obvious idea of a neutral model is demonstrated in Figure 1, where without specific provisions the eight independent computer programs A..H need 8 x 7 = 56 interfaces to share their data, while some kind of centralized format only requires eight interfaces.

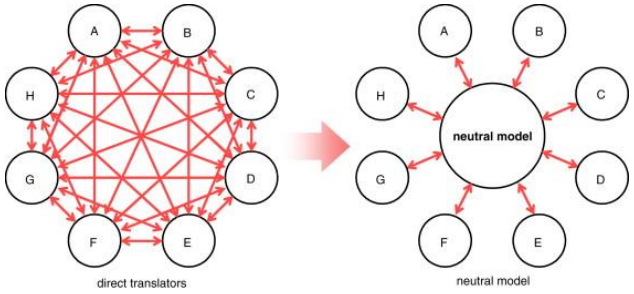

Figure 1: The neutral model saves on interfaces (from Gielingh (2008)).

The paradigm of the neutral model is reflected in several neutral file formats which are used for this purpose. The search for a recent overview of the popularity of the various (file) standards for product information did not yield anything, so we fall back on the survey in Srinivasan (2008). The most frequently used data exchange standards appear to be DXF, IGES and ISO 10303 — commonly known as STEP, STandard for the Exchange of Product model data — with a combined utilization of 60%. The remaining 40% cannot be considered to be suitable for generic exchange of 3D data, for example PDF is listed, very useful, but 'just' a document format. Although quite widely used for data transfer, DXF is merely a drawing exchange and not specifically suitable for the exchange of product model data. IGES dates back to 1980, and, as its name suggests — Initial Graphics Exchange Specification — is more aimed at the exchange of 3D shape than of a product model. Although e.g. in Kirkwood and Sherwood (2020) STEP is recommended above IGES, the latter is still widely in use, for example for exchanging the shape of a ship hull in IGES types 126 or 128, which encode for NURBS curve or surface representation. This demonstrates a phenomenon that shall be encountered in a broader sense later: IGES contains a wide range of some hundred types, for all kinds of mathematical representations of shape, such as lines, curves, planes, surfaces and solids. So, the producer of a computer program that produces an IGES file is free to choose a few favorite representations, there is no need to support all (actually, nobody does).

This leaves STEP as perhaps the most viable alternative for product data exchange, a conclusion which is drawn in many papers, e.g. Kim et al. (2008) where it is proposed to use STEP not only for the exchange of 3D shapes, but to extend it to design intent (e.g. parameters, features, constraints and history). Furthermore, STEP specializes in specific areas of industry for so-called Application Protocols (APs), with maritime application AP215 (Ship Arrangement, see ISO (2004a)), AP216 (Ship Moulded Form, ISO (2003)) and AP218 (Ship Structures, ISO (2004b). Incidentally, there is a short related anecdote to tell: when we ordered the STEP standard from the Netherlands Standardization Institute, a whole package arrived, but without AP215, 216 and 218. On enquiry, they maintained that these had since expired. Fortunately, the purchase could be made at ISO in Switzerland, but the exact status is now not entirely clear to us. Anyway, multiple researchers have formulated a preference for STEP for maritime application, e.g. Whitfield et al. (2011) and Shiplys (2019). Qin et al. (2017) report on some shortcomings of STEP, and they propose some improvement by combining it with Web Ontology Language (OWL) methods.

All these standards aim at a model for shape, with some ambition to grow towards product modelling. They have evolved into extensive books covering all kinds of pre-considered variations, more or less like a dictionary of a human language. Some considered that too limited, because life-cycle support, explicit semantics and relationships between entities are missing. This awareness has gradually led to a new standard, ISO-15926, see ISO (2005), from which a readable overview is presented on https://15926.blog/. The latter reports a twofold goal, a) global semantic interoperability and b) archiving, collecting and integrating plant life-cycle information. This terminology already reveals that this standard is leaning towards process plants, which might limit the applicability in the ship design area. Nevertheless, in the ship-borne integrated piping system as reported in Koelman (2024) the underlying data structure was inspired by ISO 15926. Another maritime

connection is that in a Dutch maritime research program from a decade ago, “Integraal Samenwerken”, a pilot with ISO 15926-11 (see ISO (2023)), was commenced — with the keywords “triple” and “Gellish” — but that did not proceed.

Another interesting industrial standard that has a wider scope than just the product and its shape, is ISO 81346, see ISO (2022). Here the multifacetedness of a product is addressed by assigning different *aspects* to a product, where each aspect can have its own hierarchy and taxonomy. An example is given in Figure 2, which shows three aspects: a) the constructional relations with the components, b) the functional relations and c) the spatial relations (e.g. the location). The application is not limited to these three; other aspects may also be considered, such as financial or logistical. As argued in Leclerc et al. (2022), this framework fits very well with a Systems Engineering approach, a method that is expected to enhance the productivity of the maritime sector (see the acronym *MBSE* in Maritiem Masterplan (2023)).

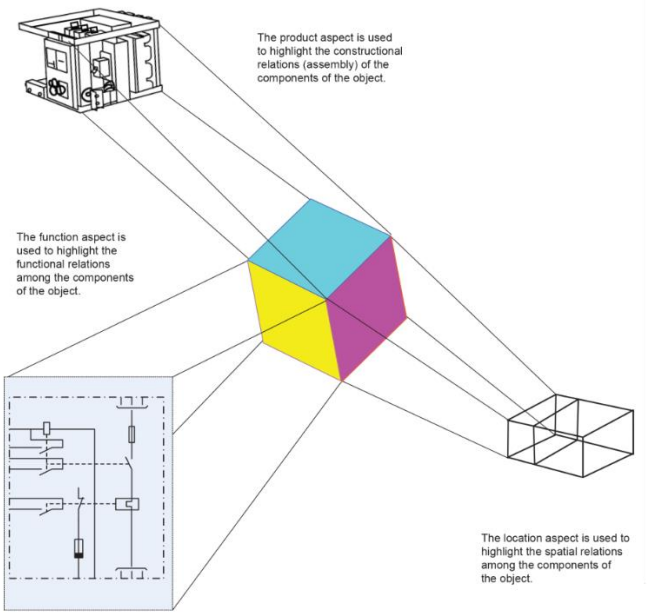

Figure 2: Different aspects of an object (from ISO (2020)).

The discussed standards relate to properties of a product in the design and construction phase. An emerging field is to include operational data as well, where a significant contribution is delivered by sensor data. This application is targeted by ISO 19848 (see ISO (2023)), which has been used by Fonseca et al. (2022) for an experiment with a digital twin of a scale model of a ship. Another emerging standard is the XML-based Open Class 3D Exchange (OCX) (see Zerbst (2023)), however, this is at the edge of our area of applicability, as expressed on https://3docx.org/: “The OCX is a vessel-specific standard addressing the information needs by the classification society”.

This ends a somewhat impressionistic overview of data standards and neutral data formats. A follow-up section discusses applicability and pitfalls, but first the data flow in a typical maritime project is outlined.

Data Flow and Growth in a Typical Maritime Project

Kirkwood and Sherwood (2021) contains an interesting approach to simplify the sustained integration between CAD and CAE. This is motivated by the CAE application being a FEM analysis, and indeed such a representation may contain somewhat less detail than the original CAD model. However, in the maritime world CAE is predominantly seen as the precursor to production, which implies that data is becoming both more detailed and richer. For example, a watertight transverse bulkhead in a ship. In the first instance, this is just a line on a GA plan (or a plane in a 3D CAD model) with the intrinsic property that it is watertight, so it could act as a separator between tanks or compartments. This limited information is sufficient for producing tank capacity tables and damage stability computations. Later in the design process, this bulkhead data is extended with details of plate thicknesses, panels, beams, stiffeners, and perhaps paint details or manufacturing logistics; augmented with data that can be *derived*, such as weight, centroids and cost. And this all belongs to the bulkhead that started as a line in a GA plan. Consequently, when in a design update the bulkhead is shifted one frame position, some of

these properties have to change correspondingly. In a sense this issue is an instantiation of Figure 2, because functional, constructional, financial and logistical aspects are included, each with their own taxonomy and/or codification system. But it is also a matter of Level Of Detail (LOD), for example when at the stage of the General Construction Plan only the main structural elements are included, which are extended with brackets and welds in the final production preparation stage. If one would have the desire to shape such a system in a third generation (3G) programming language — only in RAM, without considering permanent storage and interfacing — a structure with arrays, classes and pointers connecting the different entities, could do the job. This is depicted in Figure 3, where:

- The bulkhead is part of subsystem "bulkheads" (which in turn could be part of the system "hull structure").
- Each bulkhead contains a GUID[4]/UUID[5], which acts as unique and permanent identifier (what is called a Virtual Persistent Identifier in Kirkwood and Sherwood (2021)). A unique identifier from another source (e.g. from a supporting data management system) might be an alternative for the GUID.
- Each bulkhead contains three types of subclassed information:
    1. Topology and geometry.
    2. References to compartments on both sides (in the system "compartments").
    3. List of panels.
- The CAD system manages 1 and 2, never "sees" 3.
- The CAE system uses 1 and 2 and manages 3.

Please understand that this 3G solution is only presented to elucidate the process around and with the data. Practical considerations, such as the lack of a common 3G programming language, prevent its actual implementation.

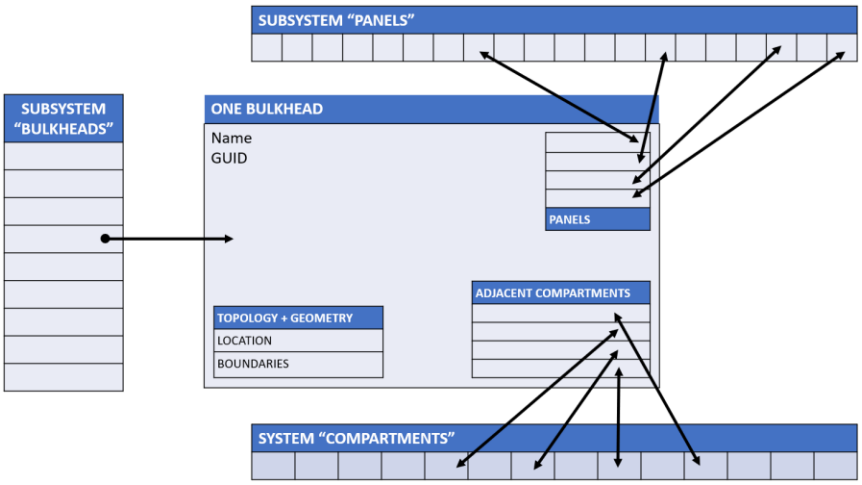

Figure 3: A class-based example of a data structure around a bulkhead, suitable for integrated CAD and CAE

This example illustrates the continuously increasing completeness of information over the whole design and engineering process. This observation is somewhat contrary to the notion of design phases — concept, preliminary, contract and detail — of which it was once useful to distinguish between. The demise of the notion of distinct phases can be witnessed in practice, for example the typical 'preliminary design' requirement of sufficient damage stability is greatly influenced by the presence and position of pipes, valves and ventilation openings; details that are typically addressed in later moments of the design process. Also in literature this trend can be observed, for example a recent state-of-the-art report on ship's design methodology, Erikstad and Lagemann (2022), counts zero entries of the words "design stage". Yet, the title of the present paper addresses a gap, which is not a gap between design phases, but between models. The early design model is a bit more holistic — including non-material aspects such as simulation results — but smaller in size than the detailed model. Bridging the gap between models does imply that the connected tools should be unified for all design activities, indeed the objectives in the earliest stage of the ship design will differ from the later stages, as eloquently motivated in Andrews (2018).

Finally, there is the issue of bidirectional vs. unidirectional flow of data traffic. The ideal with a neutral model has always been that all connected applications are able to read and write from the central storage, as depicted by the bidirectional arrows in Figure 1, at the right. If the information involved is simple, such as an isolated number, then bidirectionality is easy to achieve. However, if complex logic or functionality is involved, then it can happen that not all applications are able to perform modelling changes. For example, the duality between compartments and bulkheads & decks can be managed by one

[4] https://en.wiktionary.org/wiki/GUID
[5] https://en.wikipedia.org/wiki/Universally_unique_identifier

application only, while others can read and use that information without the need for specific tools to address that duality. Another example is hull shape modelling, where one parent application has the specific modelling tools, acting upon a parent representation, while other representations — such as STL, VRML, X3D and 3D-wireframe — at each design change are instantaneously derived and stored in neutral form. Ready to be used by other applications, without them having the tools for shape changes. If different applications are geographically dispersed then unidirectionality might be a bit awkward, but when two applications are open in two windows on the same monitor then it is neither unnatural nor time consuming that only one of the two can be used to make hull shape changes. It might even offer an advantage when an HVAC engineer is not able to change the shape of the bulbous bow.

Applicability of Neutral Models and Data Standards in Ship Design and Engineering

At this point, the paper suffers from self-plagiarism, because what follows is an anecdote that one of the authors surely must have told hundreds of times over the past 20 years. As reported by Owen (1997), STEP has aimed at 'completeness', which led inevitably to 'conversion'. This is illustrated by nine different representations of a circular arc in 2D, e.g. a) centre + radius + start angle + finish angle, b) center, radius, start point, end point, or c) polyline. If an application should want to support all these representations, and it uses one of them internally, then that implies eight conversion algorithms being required. From which item a) can be done with simple math, but item c) already requires some numerical analysis, which may result in round-off errors.
A similar example can be found in STEP AP 216, Ship Moulded Form, which supports three alternative representations: offset table, wireframe and surface. So, if a computer program internally applies a NURBS surface representation, and it receives a STEP file in offset tables, then it should be able to convert. Which is no trivial task in this example, see Koelman and Veelo (2013).

These are examples of the 'variation in representations', which are also addressed in the seminal paper by Gielingh (2008). There it is concluded that while we aimed for the nirvana of Figure 1, due to the large variation in representations, in practice only a subset is supported, leading to a situation of all kinds of ad hoc representation conversions, as depicted in Figure 4, regardless whether these representations are stored or communicated by DXF, IGES or STEP. This is the fate of any neutral model. One way or another underlying representations have to be converted, unless the world (or a workable subset of the world) decides to use a single representation for each and every entity.

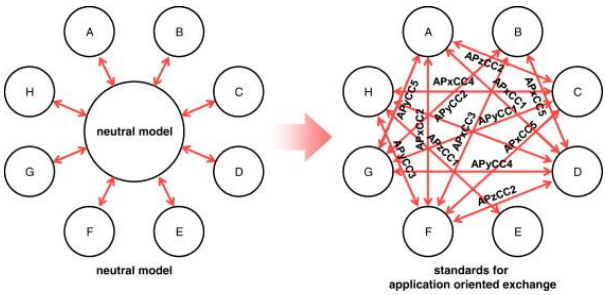

Figure 4: From theory (left) to practice (right); the neutral model doesn't really exist, from Gielingh (2008).

Another issue with standards, such as IGES, DXF and conventional STEP, is that each only contains geometric data as stated by Qin et al. (2017): "A well-known problem of STEP AP 203/AP 214 neutral file-based exchange method is that this method is limited to exchange geometric data, where those nongeometric data related to design intent, such as construction history, parameter, constraint, and features, are completely lost after the exchange". A practical issue of conventional neutral formats is that it does not always work properly, so that their usage may lead to errors and information loss, see Gielingh (2008) on STEP files read into an application and subsequently identified: "significant differences between these files were found: some entities disappeared, others appeared, and others were changed". This observation was confirmed recently in Kurylo et al. (2023).

In particular STEP AP 215 is quite extensive, but still does not cover everything:

- It supports a single permeability per compartment, while in practice intact and damaged permeabilities will differ.
- Three types of block coefficients are specified, but not the one based on the design waterline length, as required for Holtrop's resistance prediction. It is questioned why include such *derived* information as the block coefficient in a standard? Each application should be capable of dividing a volume by length, breadth and draught, specifically in the variant that is relevant for that particular application.

- The height of the sounding pipe is included, but not its shape.
- A tank total volume and corresponding Center of Gravity is included, but not its free surface or a complete tank table provided. Again, here the question is why is easily derivable information stored?
- Although some relationships between entities are supported, there is no generic support for explicit dependencies and relationships between entities.

So, it can be concluded that STEP AP 215 is on the one hand too extensive, while on the other hand not complete. Probably, that will be the fate of any global neutral model. Furthermore, it can considered to be a dictionary, not an ontology. Nevertheless, a lot of effort has been put into developing this standard, and many parameters and entities have been given a name and meaning, so why not use it, at least as inspiration?

The attractive feature of ISO 81346 (ISO, 2022) is that it addresses the multifacetedness of an object explicitly, but it defines a framework more than that it provides an implementable solution. Furthermore, it includes a standard on the coding of the location of objects, however, this is tailored to modeling buildings and is not directly applicable to ships.

Building Information Modelling (BIM) standards have not been discussed in the section on neutral models, because of the large volume, complexity and diversity of standards, national and international. The authors have not studied BIM in detail, yet BIM ISO standards 23386 and 19650 appear to be quite high-level, providing more of the general structure than relevant details. It seems that for many different fields of application this is further elaborated on a national or regional scale, (see www.etim-international.com/, which contains article codings, for example: electrotechnical, HVAC and plumbing components). Although these components are vital in the ship's outfitting phase, their relationship with the design and engineering activities is relatively limited.

TOWARDS THE SEUS DATA MODEL

At this point the conclusion is that of all the existing models and standards, none is directly usable for our purposes. Although, precise purposes have not so far been explicitly formulated, and thus this is addressed in the next subsection.

Requirements for the SEUS Purpose

Minimum requirements can be derived from the analysis and discussion above, and consist of:

- A dictionary of the names and implicit relationships between entities. Although it is perhaps a bit premature to mention a solution at this stage, STEP AP215 provides a useful basis for this, without necessarily needing to adopt everything in it, and with the knowledge that it will need to be extended here and there.
- Support for explicit relationships between entities.
- Support for variations in representations. Since these are uni-directional an object may be defined by one parent application, which derives from each design change other child representations, which are read-only for the other applications of the system.
- Support for multifacetedness, so an entity can be part of multiple taxonomies or other data structures.

Furthermore, there is a feature that, very strictly speaking, is not minimally required, but which could be extremely useful in practice: that of support for *functions*, in addition to *data*. A function is a procedure, a subroutine, a piece of processing software that can be called by all connected applications, implemented as DLL, API or Remote Procedure Call (RPC[6]). The advantage of such a tool has been discussed in Koelman et al. (2015), under the name "request/reply".

Finally, there are some desirable features that are not strictly necessary, but could prove to be useful:

- A tool for documenting the essence and properties of entities and relations. This may be done in a common text editor, typically Word, however, that is not the most user-friendly tool, and for the tabular and reference work at hand, its usefulness is even less. Preferably, the documentation tool is integrated with the data management tool because they both deal with data, structures and relations, either to be understood by a human or by the computer.
- Explicit support for data integrity and authorisation.
- Aspects of system performance, such as processing speed and limits in data sizes. However, this is considered to be self-evident, and it is a bit premature to start quantifying it at this point, but it should not be forgotten in the end.

The ambition of the SEUS project extends beyond the design and engineering phases, which means another similar endeavour to find the data model links for operational data and engineering. This phase is not included in the present paper

[6] https://en.wikipedia.org/wiki/Remote_procedure_call

although the approach can be applied to it in later stages of the project. In PDM terms, the focus of this paper extends to linking “as designed” and “as engineered” data models, with a potential to extend to “as build” and “as in operation”. Hence focusing now on the data models links, means applying later on a similar approach to digital twin platform solutions where different applications can form digital threads.

A First Experiment with Graph Databases

Although it is possible to implement a shared data format that can represent the relations depicted in Figure 3 on the basis of XML or JSON, that would be a laborious undertaking, because every element would need its own GUID and relations would need to be expressed in those terms. It would require a lot of cooperation, extensive formal specifications as well as data validation. The resulting text files would be very substantial and difficult to interpret by human designers, so the advantage of it being in natural language text would diminish. An alternative to sharing data in files is to share a database.

A traditional relational database consists of tables of rows and columns, where each row represents one data entry, and each property of that entry sits in its dedicated column. This enforces the need for the data to be structured: each column contains only data of a particular type and meaning, and each entry in the table has the same number and types of properties. Relations between entries are encoded as a property containing a reference to another row, possibly in another table; this is called a foreign key. An example would be a table of users and a table of orders, where each order has a reference to a user that placed the order. This encodes a “one to many” relation, where a user can place multiple orders. To collect all orders that were placed by a particular user, the entire column must be searched for matches with the user’s key. Because foreign keys are subjected to the rigid structure of this database format, relational databases aren’t very well suited to model systems with many relations or arbitrary relations, such as social networks and financial systems.

A different type of database has emerged that allows modeling completely unstructured data, the so-called graph databases. Different approaches exist, but common to them all is that a graph database describes nodes of information and how various nodes connect to each other. As such, the term “graph” refers to topology and discrete mathematics, not graphics. Depending on the application, a graph database can yield higher performance than can a relational database, it can be easier to query and it gives more freedom in conceptualising a system. Another aspect that is gaining relevance is that a graph may be easier to train using AI than a collection of tables, because semantics are expressed more clearly.

It is important to note that in the end there it is not necessary to use a graph database. A graph database gives the freedom and flexibility that is important for solution exploration. If the experiment is successful, this will to some extent demonstrate that it is implementable. Preferably, there is a concept that can be filled in by the PLM software of the SEUS consortium partner Contact Software.

In early 2024, the popularity ranking site db-engines.com had 41 graph database management systems (graph DBMS) in its ranking, and 13 additional DBMS capable of representing graphs as secondary model. Many of these build on open-source implementations, with some of them offering additional commercial services, which are seen to be more attractive than purely commercial solutions. Without the desire or need to do an exhaustive evaluation of all of these, the following graph DBMS have been considered: Neo4j, NebulaGraph, Memgraph, ArangoDB, Redis, GraphDB and Virtuoso. We have installed and programmed against both Neo4j and ArangoDB, after which the latter came out as the more appropriate DBMS. ArangoDB is performance oriented with features that allow some structuring of the data where that is desired. Nodes and edges in the graph are functionally equivalent to JSON documents, which can include arrays. Relations between nodes can be encoded as edges, which are just like nodes but with mandatory “from” and “to” properties containing node identifications. Edges allow the application of typical graph algorithms such as “shortest path” and facilitate validity guarantees that prevent dangling edges. Alternatively, relations may be encoded by having references as a property of the nodes themselves. In this way, ArangoDB offers a mix between graph database and document store.

Thus, an experiment has commenced representing the case depicted in Figure 3, using ArangoDB. The source code of the experiment is publicly available on GitHub[7]. Communication between the DBMS and the client program happens over HTTP. This allows the database and the client to be at different geographical locations, which is how the experiment was developed. Also included is an option to run the experiment in a Docker container, which simplifies reproduction and demonstrates that network overhead can be reduced when everything runs on the same hardware. In this case the reduction was 60 milliseconds, to under ten milliseconds per query. Evaluation of processing speed at scale is not part of this initial

[7] https://github.com/seus-project/graphexperiment

experiment, but increasing the size of the graph to one million nodes meant the slowdown was measurable but not significant. The VelocyPack binary transport[8] was not utilised, which has the potential to increase throughput.

The main question that this experiment tried to address is how the data of the design model can be sensibly organised in a graph database. It was noted that not all required functionality had to be covered by the database as such, because rules and logic in the governing applications can guarantee database consistency. As the database was not filled directly by an unauthorized system or human, nonsensical relations should in practice be prevented. Similarly, aspects of authorization can be handled by applications.

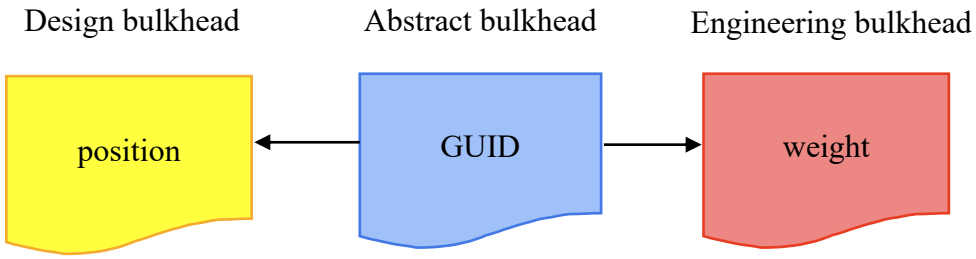

Figure 5: Bulkhead representation.

Considering the viewpoint of the applications, the ship model can be simplified by only considering bulkheads. For design software, a bulkhead primarily defines the partition of space and thereby determines the volumes of compartments. For engineering software, a bulkhead is primarily a structural object. The objective of the experiment was to allow both design and engineering software to work on the same bulkheads without getting in the way of each other. The approach that this experiment took (Figure 5) is to start with a node that represents the abstract concept of a bulkhead, by only containing its identity. Connected to it are separate nodes that represent specializations for design and engineering. This principle is scalable to additional systems, such as cost estimation, construction process logistics and maintenance.

Each specialisation contains properties that are specific to that specialization and can also extend the graph with nodes that are specific for that representation, as is shown in Figure 6.

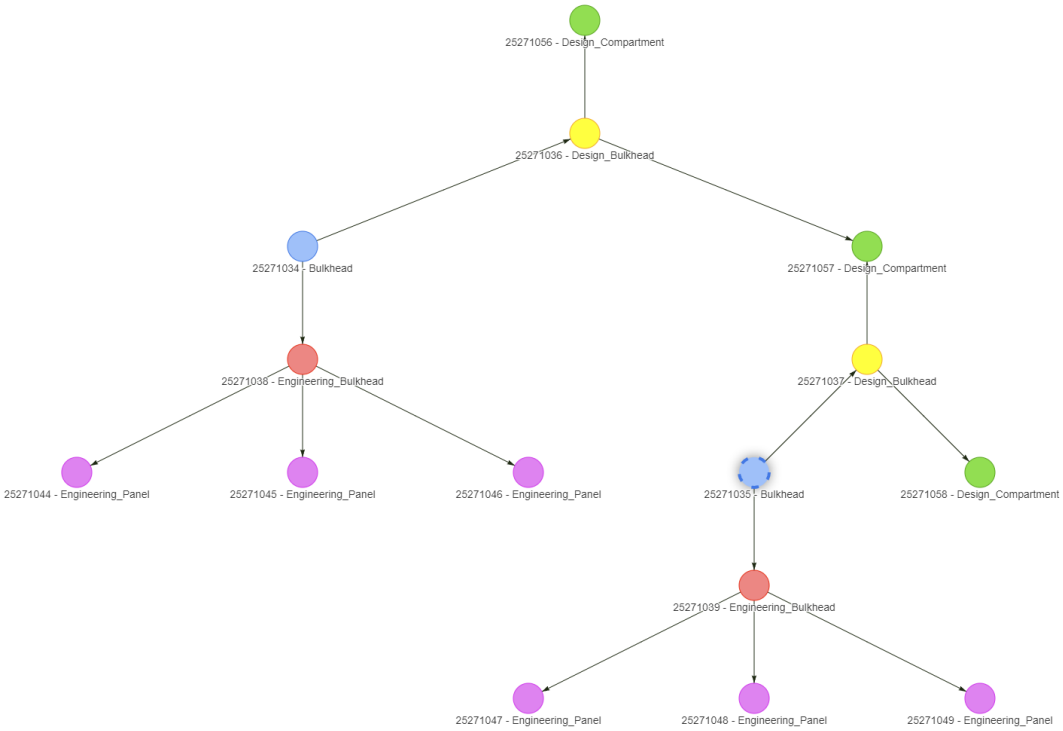

Figure 6: Automatically generated plot of a graph database with two bulkheads and three panels per bulkhead.

[8] https://github.com/arangodb/velocystream

The source code for the experiment contained a procedure[9] that synthetically generated data for a given number of bulkheads, where each bulkhead consists of a given number of panels on the engineering side of the graph, and on the design side of the graph the boundaries of compartments are defined. Figure 6 shows the nodes and connections for two bulkheads, three compartments and six panels. The connections are directional, but they can be traversed in any direction. The following lines of ArangoDB Query Language (AQL) are an excerpt of the source code that traverses the graph and lists the compartment names together with the positions of their bounding bulkheads:

```
FOR c IN Design_Compartment
    FOR db IN INBOUND c Design_BulkheadAdjacentCompartment
        RETURN [ c.name, db.position ]
```

The experiment demonstrated how the output changed when one of the bulkheads was moved to a new position.

In practice, the possibilities of extending the graph with additional information are endless. The design software could include constraints from the constraint management system (see De Koningh et al. (2011)), and any other model data that is currently stored in separate files. The engineering software can store the panel data, such as identification number, dimensions, and position in relation to the appropriate local coordinate system, and connections to other nodes representing other parts and production details. Even versioning can be implemented as part of the graph, where properties of nodes aren't simply updated, but a new version of the complete node is pushed on top of a stack of node versions, where edges between the versions contain the date of the change, approval by superior, etcetera.

In principle, the complete graph is traversable by both engineering client and design client. For example, when the engineering software needs the position of a bulkhead, it could locate the abstract bulkhead, traverse to the corresponding design bulkhead, and read out the position from there. But this requires the engineering software to have knowledge of the topology of the design side of the graph, and it also means that the developers of the design software cannot change their side of the graph without coordinating this with the developers of the engineering software. Interestingly, ArangoDB allows for the installation of so-called Foxx microservices, that can be used to provide a stable API for querying potentially dynamic subgraphs. A Foxx microservice is essentially a snippet of JavaScript that can be uploaded to the graph DBMS after which it can be accessed at a specific URL. When a client accesses that URL, the code in the microservice is executed inside of the DBMS, and the resulting data is returned. Among other things, a microservice can perform a graph query. This way, the developers of a particular system are free to change the layout of their side of the graph if they adapt their microservices accordingly, and other systems can continue using them without change, coordination, or synchronisation.

The experiment demonstrates that with 22 lines of JavaScript[10], a microservice can be implemented by which the position of a bulkhead can be queried by name: thus, `http://localhost:8529/_db/seus/bulkhead_position/B1` produces the output `[10]`. ArangoDB's HTTP API follows the OpenAPI specification and is integrated with Swagger 2.0, meaning that the database server serves its own API documentation together with a web form where the API can be tested interactively. This functionality also covers the microservices. So, the microservice can include documentation that the returned value is the longitudinal distance in metres between the aft surface of the bulkhead and the aft perpendicular, with positive values meaning forward, and negative values aft, of the aft perpendicular.

A microservice can also contain additional logic. Thus the weight of a bulkhead does not need to be a discrete property stored in the engineering representation of a bulkhead, but a value that is determined dynamically, by a microservice. In an early stage of the design, the structure of a bulkhead might still be undetermined, meaning that there are no nodes connected to it that represent panels and stiffeners. In that case, the microservice could return an estimated weight, possibly with a low level of confidence. But as soon as panels have been defined, the plate thickness and dimensions known, the material known, and the stiffeners have been defined, then the weight can be calculated with a high level of confidence. When the weight of a module is requested, the microservice can recurse into all the parts that make up the module and accumulate the weights of the parts.

What is concluded from this experiment is that a graph database has great potential in the implementation of a shared data model. Flexibility and scalability is offered by a graph database, and the extensibility with microservices addresses to some degree the need for documentation (taxonomy) and RPCs. Whether the ship design department and the engineering department are working on the same model in different geographical locations, or all systems are run on the same computer, the HTTP interface of the DBMS means that it is applicable in either situation.

---

[9] https://github.com/seus-project/graphexperiment/blob/v0.1.0/source/app.d#L150
[10] https://github.com/seus-project/graphexperiment/blob/v0.1.0/foxx/bulkhead_position.js

SEUS' DATA MODELS

The purpose of this section is to conceptualise the model of data, relations and services from SEUS. Prior to that the existing data models of the design and engineering software suites are sketched out.

The pre-existing data model of the design software

The basis of the ship design data model is formed by the hull shape, which can have two representations. The most complete is a solid model with closed curved surfaces, in proprietary H-Rep representation, see Koelman (2003). Another contains a wireframe, i.e. cross sections and stem/stern & deck contours, which is sufficient for all computations. The solid/surface model is convertible to PIAS' wireframe, and to IGES, NURBS surface and IGES/DXF 3D curves. The wireframe model can be converted to solid/surface, albeit with human assistance.

The space inside the hull is filled with constituting planes (bulkheads and decks) and compartments (tanks and other spaces), which form a duality: planes shape compartments, while spaces are bounded by planes. This duality is modelled by a proprietary method (see De Koningh et al. (2011)), which is based on Binary Space Partitioning (BSP). These constituting planes divide the internal of the ship hull into convex spaces, which are called subcompartments. Multiple subcompartments can be assigned to be part of a compartment. In this structure the spaces are 'logical' building blocks, while the compartments are physical, i.e. they are watertight. A finer subdivision may be obtained when non-constituting planes are also taken into account; these are not explicitly modelled, however their presence can be effectuated by modelling subcompartments by their corner vertices, typically, but not limited to, eight. These two compartment modelling methods can be mixed, and an average PIAS user applies the plane-based method for larger, systemic, subdivision planes, and the vertices-based method for smaller tanks or voids. Regardless which of the two modelling methods has been used, the final compartment shape is computed by an intersection with the ship's hull. Although this shape is the core property of a compartment, other properties are also stored, such as its name, permeability, design density, location and type of external openings, and location and shape of the sounding pipe(s).
From the viewpoint of hydrostatic and (damage) stability the connections from and to compartments — such as by pipes, internal openings and ducts — are as equally important as their shapes. This forms an integral part of PIAS' data model, but as this is already a topic of another paper on this conference, Koelman (2024), it is not discussed further here.

This comprises more or less the geometrical and topological ship design data, which are shared in many stages of design and engineering. However, an accurate prediction of all kinds of technical properties — such as draught, cargo capacity, power consumption, stability and strength — also depends on the ship's weight and its distribution. Obviously, PIAS supports this, basically with a long list of numbers of components, their weights, three spatial coordinates, and their aft and forward boundaries.

The 'design' software is not only applied during the ship design phase, but also to produce simulation and delivery documents. These include tables of tank capacity, assessment of (damage) stability, longitudinal strength and maneuvering characteristics. Such reports are currently exported to ASCII, XML or Word formats, although many of these computations can be offered in an RPC fashion.

Furthermore, all entities can be equipped with a Virtual Persistent Identifier, possibly a GUID, a concept that has attracted our attention before. This uniquely and permanently identifies an object, which facilitates tracking and processing changes.

The pre-existing data model of the engineering software

The data model of CADMATIC Hull is based on relationships, which directly support the requirement that any change in shape or position will directly affect another element and provide a chain reaction to others. In addition, standards such as end shapes, holes, cutouts, lugs are included as a feature and not as their shape; and its form is recorded in a referenced library. The body and thickness directions are also assigned as a parameter of the object and therefore the CL and reversed frame have a direct influence on the final 2D and 3D presentations. A group of elements (plates and profiles) are recorded as a whole in a sub-database and this part is linked to a parameterised grid definition which also indicates the x, y and z directions or a plane definition defined as a surface. The whole (the ship) is a collection of these sub-databases, which can be flexibly exchanged, this approach facilitates simultaneous work in large models and possibility to replicate database of the project for several physical servers to provide seamless experience for remote work teams and users. This is the basis of the Hull application, from which all presentations are derived, such as 2D cross-sections, 3D views, derived information, such as

weight, length and material, and production data, such as cutting, robot and bending data. All objects have their unique GUIDs, which are used as a link for all relationships.

The CADMATIC applications are Hull, P&ID, Plant Modeller and Piping Isometrics & Spools. All the components, parts, symbols, and design instructions are stored and managed in Library & Project Databases. These databases also include the format control for sheets, listings, and reports. All applications use the same database, ensuring the information remains the same throughout the design project. Access is governed by the COS (CADMATIC Object Storage, see Figure 7) environment, where data is protected from being modified at multiple sites at the same time. Therefore, remote design teams can work in distributed projects using a common model database without conflicts. Projects are split into blocks, general project data, hull line subsets and 2D symbols, and these can all be saved to the COS server separately. In P&ID, the designer describes the process schematically in 2D format using predefined symbols and metadata information. In Plant Modeller and Piping Isometrics & Spools the process diagram is rebuilt in a 3D format to describe the ship in a realistic way using pipes, fittings, equipment, structural components, etc.
While different applications work in a slightly different way, depending on the discipline they serve, the overall project data is consolidated in the COS database. CADMATIC represents so called "intent-driven" CAD solution, which focuses primarily on the shipbuilding nature of the designer's work, see Dush et al. (2017). Each application has their own API to serve the needs of particular integrations for design disciplines, while COS Web API serves the needs of integration with overall project data.

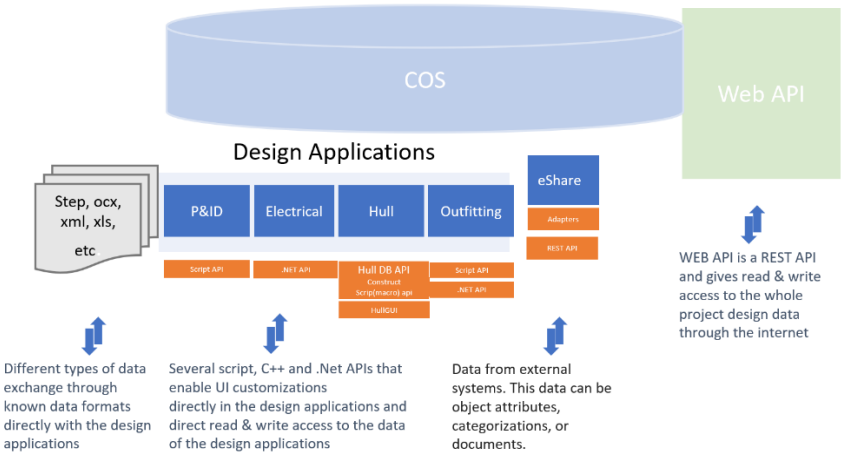

Figure 7: CADMATIC Object Storage structure.

A sketch of the SEUS service warehouse

As a learned lesson, it is worth reflecting on a prior CADMATIC – PIAS interface, see Koelman et al. (2015), which comprised a) direct communication over TCP/IP instead of a shared database, b) data synchronisation on demand, rather than continuously and c) exchange of high-level data entities, based on STEP semantics, which means that in essence the deepest data representations were not shared. For example, both systems had a notion of *deck*, which was shared, while the underlying representation differed quite radically. As such, that software collaboration was impressive to see; with two windows open, showing the same model in the two distinct applications, with one press of a button the changes from one was transferred to the other. Nevertheless, the direct TCP/IP communication had one drawback: the lack of a central permanent storage; if one of the applications was not connected the *synchronisation actions* from the other vaporized. This conclusion, combined with the other analyses and experiments in this paper, led to an envisioned SEUS warehouse with the following functions and services:

1. Storage of data and their relationships are extendable, offering varying Levels of Detail, and multiple facets (i.e. the entities can be part of multiple taxonomies, each of a different kind and with a different purpose).
2. With data semantics based on the maritime STEP application protocols, extended where required.
3. Should this come down to a dictionary, then preferably some integrated documentation system for human use is required.
4. There is easy communication by API or RPC with this storage system, including systems of varying types, such as high-level programming languages and scripting tools.
5. It is extended with a set of system-wide services, among which there is conversion of data representations.
6. Control of access of the data is possible.

To underline that this is not limited to data, this is called a *service warehouse*, in analogy with a physical warehouse, where services are also provided, with or around the goods.

However, the world is littered with abstract plans and grand designs from the past, so how can the feasibility of this SEUS service warehouse be assessed? The first two items combined have been a bottleneck for many years, but the experiment described in this paper suggests that solutions exist. It is now time to sharpen this choice by also involving considerations around SEUS project strategy, involved effort, costs and other aspects of licencing of supporting software. At present how existing software solutions can satisfy the third and fourth requirement is being investigated.

Finally, a word on the *modus operandi* with this service warehouse. It will be obvious that the design and engineering programmes will not be using the service warehouse, so synchronisation between the internal representations and the service warehouse will be required from time to time. The question as to whether this is done automatically (at certain time intervals), or at the instigation of the user, or even restricted to authorised users, is a practical one that it is considered can be answered later. There may be a setting for this if necessary.

CONCLUSIONS

In this paper the background, requirements and desires for the integration of design and engineering shipbuilding program suites have been sketched. After a survey of maritime data standards, it was concluded that none of these are directly applicable for our purposes, mainly because they do not support the inherent multifacetedness of the design and engineering data. The recently emerged category of *graph databases* might perhaps fill this gap, and as a first investigation a practical implementation with a set of bulkheads and compartments has been created and evaluated. The results look quite promising, although non-technical aspects, such as licensing and costs, have not yet been considered. Anyway, these experiments, combined with an analysis of the existing data models of SEUS' design and engineering applications, led to a sketch design of SEUS' central *service warehouse.* The considerations and experiments have shown that viable tools and methods exist to solve significant software integration aspects. Future steps in the SEUS development are intended to be:

- Exploration of attractive alternatives for the functionality offered by graph databases.
- Addressing the integrated support for RPCs.
- The management of the (STEP-based) dictionary, or, alternatively, the integration of dictionary and warehouse software implementation.
- The development of a second demonstration case, with piping, associated components and their connections to equipment and compartments.

CONTRIBUTION STATEMENT

Author 1: Conceptualization; conclusions; sketch of the warehouse; literature review; PIAS background; writing. **Author 2:** Graph databases research and writing; review and editing. **Author 3**: Conceptualization; conclusions; sketch of the warehouse; CADMATIC background; writing. **Author 4**: CADMATIC background.

ACKNOWLEDGEMENTS

The SEUS project has received funding from the Horizon Europe Framework Programme (HORIZON) EU program under grant agreement No 101096224. Info is updated at http://seus-project.eu/. This article reflects only the authors' views, and the European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

Andrews, D. (2018). The Sophistication of Early Stage Design for Complex Vessels. Trans RINA, Special Edition, IJME, 160, (SE 18).

De Koningh, D., Koelman, H.J. & Hopman, J.J (2011). A Novel Ship Subdivision Method and its Application in Constraint Management of Ship Layout Design. Journal of Ship Production and Design, 27(3), 137-145.

Dusch, T., Franke, B., Grau, M. & Zerbst, C. (2017), Intent-driven CAD vs. Mechanical CAD in Shipbuilding – A review and Solution Outline, ICCAS 2017.

Erikstad, S.O. & Lagemann, B (2022). Design Methodology State-of-the-Art Report. 14th International Marine Design Conference (IMDC), Vancouver, Canada, June 28.

Fonseca, Í.A., Gaspar, H.M., de Mello, P.C. & Sasaki, H.A.U. (2022). A Standards-Based Digital Twin of an Experiment with a Scale Model Ship. Computer-Aided Design, 145, 103191.

Gaspar, H.M., Seppälä, L, Koelman, H.J. & Agis, J.J.G. (2023). Can European Shipyards be Smarter? A Proposal from the SEUS Project. COMPIT'23. Drübeck, Germany, May 23-25.

Gielingh, W. (2008). An assessment of the current state of product data technologies. Computer-Aided Design, 40(7), pp. 750-759.

ISO (2003). ISO 10303. Industrial Automation Systems and Integration: Product Data Representation and Exchange. Part 215: Application Protocol: Ship Moulded Form.

ISO (2004a). ISO 10303. Industrial Automation Systems and Integration: Product Data Representation and Exchange. Part 215: Application Protocol: Ship Arrangement.

ISO (2004b). ISO 10303. Industrial Automation Systems and Integration: Product Data Representation and Exchange. Part 218: Application Protocol: Ship Structures.

ISO (2005). ISO-15926-1. Industrial Automation Systems and Integration - Integration of Life-Cycle Data For Process Plants Including Oil and Gas Production Facilities - Part 1: Overview and Fundamental Principles.

ISO (2018). ISO 19848. Ships and marine technology - Standard data for shipboard machinery and equipment.

ISO (2022). ISO/IEC 81346-1. Industrial Systems, Installations and Equipment and Industrial Products - Structuring Principles and Reference Designations - Part 1: Basic Rules.

ISO (2023). ISO-15926-11. Industrial Automation Systems and Integration - Integration of Life-Cycle Data for Process Plants Including Oil and Gas Production Facilities - Part 11: Simplified Industrial Usage of Reference Data Based on RDFS Methodology.

Kahn, M.T.H. & Rezwana, S. (2021). A review of CAD to CAE integration with a hierarchical data format (HDF)-based solution. Journal of King Saud University - Engineering Sciences, Vol 33 (4). pp 248-258,

Kim, J., Pratt, M.J., Iyer, R.G. & Sriram, R.D. Standardized Data Exchange of CAD Models with Design Intent. . Computer-Aided Design, 40(7), pp. 760-777.

Kirkwood, R. & Sherwood, J.A. (2021). Sustained CAD/CAE Application Integration: Supporting Simplified Models. J. Comput. Inf. Sci. Eng. 21(1).

Koelman, H.J. (2003). Application of the H-rep Ship Hull Modelling Concept. Ship Technology Research. 50 (4), pp. 172-181.

Koelman, H.J. (2024). Piping Layout Integrated in Ship Design and Stability Assessment. 15th International Marine Design Conference (IMDC), Amsterdam, Netherlands, June 3-6.

Koelman, H.J., van de Zee, J. & de Jonge, T. (2015). A Virtual Single Ship-Design System Composed of Multiple Independent Components. COMPIT'15. Ulrichshusen, Germany, May 11-13.

Koelman, H.J. & Veelo, B.N. (2013). A technical note on the geometric representation of a ship hull form, Computer-Aided Design 45(11), pp. 1378-1381,

Kuryło, P., Frankovský, P., Malinowski, M., Maciejewski, T., Varga, J., Kostka, J., Adrian, Ł., Szufa, S. & Rusnáková, S. Data Exchange with Support for the Neutral Processing of Formats in Computer-Aided Design/Computer-Aided Manufacturing Systems. Appl. Sci. 2023, 13, 9811.

Leclerc, J-C., Keraron, Y., Fauconnet, C., Chauvat, N. & Zelm, M. (2022). New ways of using standards for semantic interoperability towards integration of data and models in industry. 11th International Conference on Interoperability for Enterprise Systems and Applications (I-ESA 2022), Valencia, Spain, March 23-25.

Maritiem Masterplan (2023). Aanvraag nationaal groeifonds (in Dutch). maritiemmasterplan.nl/wp-content/uploads/sites/3/2023/10/230203_Maritiem-Masterplan_Verkorte-versie-zonder-appendices.pdf

Owen, J. (1997). STEP, an introduction. Information geometeers, Winchester, UK.

Qin, Y., Lu, W., Qi, Q., Liu, X., Zhong, Y., Scott, P. & Jiang, X.. (2017). Status, Comparison, and Issues of Computer-Aided Design Model Data Exchange Methods Based on Standardized Neutral Files and Web Ontology Language File. Journal of Computing and Information Science in Engineering. 17.

Shiplys (2019). Ship Lifecycle Software Solutions (SHIPLYS). D9.7 SHIPLYS Software and its Functionality in Relation to Existing Standards and Potential for Inputs to Future Standards. www.shiplys.com/library/deliverables/d97-shiplys-software-and-its-functionality-in-relation-to-existing-standards-and-potential-for-inputs-to-future-standards/

Srinivasan, V. (2008). Standardizing the specification, verification, and exchange of product geometry: Research, status and trends, Computer-Aided Design 40(7), pp. 738-749.

Whitfield, R., Duffy, A., York, P., Vassalos, D. & Kaklis, P. (2011). Managing the Exchange of Engineering Product Data to Support Through Life Ship Design. Computer-Aided Design 43(5), pp. 516-532.

Zerbst, C. (2023). OCX on the Way from Research to Industry Practice. COMPIT'23. Drübeck, Germany, May 23-25.