Copy No. _____

Defence Research and Development Canada | Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Determination of the Most Efficient Database Format for Ship Analysis Models

Tom MacAdam, John Wallace and Michael Lichodzijewski
Martec Limited

Prepared by:
Martec Limited
1888 Brunswick Street, Suite 400
Halifax, Nova Scotia B3J 3J8

Contract Project Manager: Tom MacAdam, 902-425-5101 x239
Contract Number: W7707-088100
Contract Scientific Authority: Malcolm Smith, Defence Scientist, 902-426-3100 ext 383

The scientific or technical validity of this Contract Report is entirely the responsibility of the contractor and the contents do not necessarily have the approval or endorsement of Defence R&D Canada.

Defence R&D Canada – Atlantic

Contract Report
DRDC Atlantic CR 2012-003
April 2012

Canada

This page intentionally left blank.

Determination of the Most Efficient Database Format for Ship Analysis Models

Tom MacAdam
John Wallace
Michael Lichodzijewski
Martec Limited

Prepared By:
Martec Limited
1888 Brunswick Street, Suite 400
Halifax, Nova Scotia B3J 3J8

Contract Project Manager: Tom MacAdam, 902-425-5101 x239
PWGSC Contract Number: W7707-088100
CSA: Malcolm Smith, Defence Scientist, 902-426-3100 x383

The scientific or technical validity of this Contract Report is entirely the responsibility of the Contractor and the contents do not necessarily have the approval or endorsement of Defence R&D Canada.

Defence R&D Canada – Atlantic

Contract Report
DRDC Atlantic CR 2012-003
April 2012

Principal Author

Original signed by Tom MacAdam

Tom MacAdam

Software Development Team Leader

Approved by

Original signed by Neil G. Pegg

Neil G. Pegg

Head, Warship Performance

Approved for release by

Original signed by Calvin V. Hyatt

Calvin V. Hyatt

Chair, Document Review Panel

© Her Majesty the Queen in Right of Canada, as represented by the Minister of National Defence, 2012

© Sa Majesté la Reine (en droit du Canada), telle que représentée par le ministre de la Défense nationale, 2012

Abstract

This report sets out a high-level proposal for a data schema capable of storing ship model data suitable for various types of life-cycle management engineering analyses. A critical analysis of two existing ship model data formats, RMGScript, serving global finite element analysis, and Ship Structure Exchange, serving ship design assessment, was undertaken as the basis for the schema proposed. The new schema combines the strengths of both existing formats and sets forth guidelines for implementing extensions with minimal disruption on applications that make use of it.

Résumé

Le présent rapport établit une proposition de haut niveau pour un schéma de données capable d'emmagasiner des données concernant un modèle convenant à divers types d'analyses techniques de gestion de cycle de vie. Une analyse critique de deux formats de données de modèle existants (RMGScript : sert une analyse générale par éléments finis, et Ship Structure Exchange : sert une évaluation de conception de navire) a été entreprise comme base du schéma proposé. Le nouveau schéma combine les forces des deux formats existants et explique des lignes directrices pour mettre en œuvre des extensions en perturbant le moins possible les applications qui utilisent cela.

This page intentionally left blank.

Executive summary

Determination of the Most Efficient Database Format for Ship Analysis Models

Tom MacAdam; John Wallace; Michael Lichodzijewski; DRDC Atlantic CR 2012-003; Defence R&D Canada – Atlantic; April 2012.

Introduction or background: This work is the continuation of a longstanding collaboration between DRDC Atlantic and Martec Limited concerning research and development of software solutions for ship life cycle management (LCM). As part of this collaboration, Martec previously explored the suitability of various commercial-ship single product model (SPM) tools for supplying data for various types of LCM engineering analyses. This work builds upon those findings by proposing a data format capable of acting as the translation intermediary between SPM and LCM analysis tools. The proposal is influenced heavily by an in-depth analysis of two existing ship data formats–the RMGScript format used by Martec's Trident Modeller application, and the ship structure exchange (SSX) format used by the Lloyd's Register DIME application.

Results: A new data schema is format that amalgamates the strengths of both the RMGScript and SSX data formats. The new format favours industry-standard formats for storage of complex geometric data and parametric definitions of simpler entities; it could be implemented using standard extensible markup language (XML) technology, so as to leverage the wealth of existing infrastructure and associated technologies based upon it. A mechanism is proposed to allow the format to evolve with minimal disruption to applications that already use it.

Significance: The proposed basis for a modelling data format, if adopted could significantly reduce the through-life modelling effort required for a new class of ship. With the proposed data format, all relevant information about a ship would be stored in a format from which it can be accessed in a wide range of engineering analysis applications. This would eliminate modeling redundancy and ensure model changes are centralized and applied consistently to all analyses undertaken. The proposed data format has potential application in life cycle management support for the new ship classes being considered for the RCN under the National Shipbuilding Procurement Strategy.

Future plans: This work sets forth the data storage requirements and methodology for the new data format. It is recommended that future work cover implementation of the requirements in a formal XML schema definition (XSD) document.

Sommaire

Determination of the Most Efficient Database Format for Ship Analysis Models

Tom MacAdam; John Wallace; Michael Lichodzijewski; DRDC Atlantic CR 2012-003; R & D pour la défense Canada – Atlantique; avril 2012.

Introduction : Ce travail est la suite d'une collaboration de longue date entre RDDC Atlantique et Martec Limited concernant la recherche de solutions logicielles pour la gestion du cycle de vie de navire et le développement de telles solutions. Dans le cadre de cette collaboration, Martec a examiné divers outils pour modèle de navire commercial afin de déterminer s'ils convenaient à la fourniture des données pour divers types d'analyses techniques de gestion de cycle de vie. Ce travail s'appuie sur ces découvertes et propose un format de données capable d'agir comme intermédiaire de transposition entre les outils d'analyse de gestion de cycle de vie et de modèle unique. La proposition est influencée grandement par une analyse en profondeur de deux formats de données existants concernant les navires : le format RMGScript utilisé par l'application Trident Modeller de Martec, et le format Ship Structure Exchange (SSX) utilisé par l'application DIME de la Lloyd's Register.

Résultats : Un nouveau schéma de données est un format qui amalgame les forces du format de données RMGScript et du format de données SSX. Le nouveau format favorise les formats respectant la norme de l'industrie pour le stockage de données géométriques complexes et de définitions paramétriques d'entités plus simples; il pourrait être mis en œuvre à l'aide de la technologie XML (langage de balisage extensible) standard de façon à exploiter la richesse de l'infrastructure existante et des technologies connexes qui sont associées à cette infrastructure. Un mécanisme est proposé pour permettre au format d'évoluer en dérangeant le moins possible les applications qui l'utilisent déjà.

Portée : La base proposée pour un format de données de modélisation, si on l'adoptait, pourrait diminuer de façon significative l'effort de modélisation incessant requis pour une nouvelle classe de navires. Grâce au format de données proposé, tous les renseignements pertinents concernant un navire seraient stockés dans un format permettant d'y accéder par une panoplie d'applications d'analyse de type technique. Cela éliminerait la redondance au niveau de la modélisation et permettrait d'assurer que les modifications au niveau du modèle sont centralisées et appliquées uniformément à toutes les analyses effectuées. Le format de données proposé pourrait être appliqué dans le soutien à la gestion du cycle de vie pour les nouvelles classes de navires considérées pour la Marine royale canadienne dans le cadre de la Stratégie nationale d'approvisionnement en matière de construction navale.

Recherches futures : Ce travail explique la méthodologie et les exigences en matière de stockage de données pour le nouveau format de données. Il est recommandé que les travaux à venir portent sur la mise en œuvre des exigences dans un document de définition de schéma XML officiel.

Table of contents

Abstract i
Résumé i
Executive summary iii
Sommaire iv
Table of contents iv
List of figures ix
Acknowledgements x
1 Introduction 1
1.1 Background 1
2 Data Requirements for Analysis 3
2.1 DND Engineering Analysis 3
2.2 LR Rule Analysis 3
2.2.1 RulesCalc 3
2.2.1.1 Ship Data 3
2.2.1.2 Geometric Data 3
2.2.1.3 Material Properties 4
2.2.2 ShipRight 4
2.2.2.1 Ship Data 4
2.2.2.2 Geometric Data 4
2.2.2.3 Material Properties 5
2.2.2.4 Stiffener Profiles 5
2.3 Discussion 5
3 Existing Databases 8
3.1 Trident Modeller 8
3.1.1 The RMGScript Schema 9
3.1.1.1 Object Reference Syntax 9
3.1.1.2 Attribute Reference Syntax 9
3.1.1.3 Anonymous Object Syntax 9
3.1.1.4 Inline Math Operators 10
3.1.1.5 Inline Units Specification 10
3.1.2 Example 10
3.2 DIME 13
3.2.1 The DIME Schema 14
3.2.2 Example 19
4 Translating from SPM Into Existing Databases 22
4.1 Trident Modeller 22
4.1.1 Converting to the Native Schema 22

4.1.2 Converting to the Extended Schema ... 23
4.1.3 Converting to a Combination of the Native and Extended Schema ... 24
4.2 DIME ... 26
5 Critical Analysis of the Existing Databases ... 29
5.1 Trident Modeller ... 29
5.1.1 Advantages ... 29
5.1.2 Disadvantages ... 29
5.2 DIME ... 30
5.2.1 Advantages ... 30
5.2.2 Disadvantages ... 30
6 Plan For The Way Ahead ... 32
6.1 Storage Mechanism ... 32
6.2 Ship Metadata ... 33
6.3 Ship Geometry ... 33
6.3.1 Plating ... 33
6.3.2 Stiffeners ... 34
6.3.3 Libraries ... 35
6.3.4 Penetrations ... 35
6.3.5 End Connections ... 35
6.3.6 Brackets ... 36
6.3.7 Equipment and Other Non-structural Objects ... 36
6.3.8 Spaces ... 36
6.4 Extensibility ... 36
6.5 Miscellaneous ... 37
7 Extensibility Mechanism ... 38
References ... 47
Annex A .. DND LCM Analysis Data Requirements Summary ... 49
A.1 Global Structural FEA ... 49
A.1.1 Geometric Data ... 49
A.1.2 Section Properties and Material Data ... 50
A.1.3 Mass Data ... 50
A.2 Detailed Structural FEA ... 50
A.2.1 Detailed Geometric Data ... 50
A.2.2 Section Properties and Material Data ... 50
A.2.3 Mass Data ... 50
A.2.4 Boundary Conditions ... 51
A.3 Hydrodynamics ... 51
A.3.1 Ship Geometry and Weight Distribution ... 51
A.3.2 Hull Section Parameters ... 52
A.3.3 Appendages ... 53

A.3.4 Environmental Parameters 53
A.4 Radar Signature 54
A.4.1 Geometric Properties 54
A.4.2 Material Properties 54
A.4.3 Others 54
A.5 Infrared Signature 54
A.5.1 Geometric Properties 54
A.5.2 Non-Geometric Properties 55
A.5.3 Supplementary Data 55
A.6 Electrical Potential Signature and Cathodic Protection 55
A.6.1 Geometric Data 55
A.6.2 Material Data 55
A.6.3 Cathodic Protection System 55
A.6.4 Supplementary Data 56
A.7 Magnetic Signature 56
A.7.1 Geometric Data 56
A.7.2 Material Data 56
A.7.3 Magnetic Field Sources 56
A.8 Low Frequency Acoustic Signature 57
A.8.1 Geometric Data 57
A.8.2 Supplementary Data 57
A.9 High Frequency Acoustic Data 57
A.9.1 Geometric Data 57
A.9.2 Input Power Data 58
Annex B .. RMGScript Language Entities 61
B.1 RMGScript XML Entities 61
B.1.1 Collection Entities 61
B.1.2 Reference Value Entities 61
B.1.3 Geometric Entities 61
B.1.4 Property Zone Entities 62
B.1.5 Auxiliary Data Entities 62
B.1.6 Planar Intersection Entities 62
B.1.7 Material Type Entities 62
B.1.8 Section Type Entities 63
B.2 RMGScript Attribute Value Syntax 63
B.2.1 Units Specifiers 63
B.2.2 Object Reference Syntax 63
B.2.3 Anonymous Object Syntax 63
B.2.4 Inline Arithmetic Operators 63
B.2.5 Values 64
B.2.6 Points 64

B.2.7 Vectors 64
B.2.8 Planes 65
B.2.9 Conics 65
List of symbols/abbreviations/acronyms/initialisms 66
Distribution list 67

List of figures

Figure 1: The RMGScript example displayed in Trident Modeller. 12
Figure 2: The DIME process flow. 13
Figure 3: The DIME schema top level entities. 14
Figure 4: The ShipDefinition3D entity. 15
Figure 5: The Structure3DAssets entity. 16
Figure 6: The Boundary entity. 17
Figure 7: The Trans3DInstances entity. 18
Figure 8: DIME example geometry in DIME application. 21
Figure 9: RMGScript model of the CPF. 25
Figure 10: RMGScript model of a mega-yacht. 26
Figure 11: DIME model of the MCDV imported from SmartMarine. 27

Acknowledgements

The authors are grateful to members of Lloyd's Register Marine Product Development, specifically Paul Roberts for his assistance. In addition, thanks are extended to members of the Martec Software Development Team, particularly Eric Teng, for insight regarding DIME data storage.

1 Introduction

1.1 Background

Much research has been done on development of software tools to support rapid engineering analysis in support of in-service life cycle management (LCM) of the Royal Canadian Navy (RCN) fleet. Martec Limited and Defence R&D Canada – Atlantic (DRDC Atlantic) have collaborated on a number of related projects, commencing in the late 1990s.

The first such collaboration was under the Improved Ship Structures Maintenance Management (ISSMM) project, which saw the development of a software tool capable of generating a geometric model of the Canadian Patrol Frigate (CPF); applying various types of damage and degradation scenarios; and seamlessly performing a variety of engineering analyses using that data. The tool, termed the ISSMM Software Tool (IST), was demonstrated to be effective at cutting the time to perform various types of damage assessments on the CPF geometry.

The IST's weakness, however, was its structural modeling and storage functionality. The IST not only aimed to provide LCM analysis features, but also provided a suite of tools to allow the user to generate the vessel geometric model upon which the analyses were performed. The vast majority of the project effort was consumed by development of these geometric modeling tools, and despite this, they remained in the end the most brittle part of the application (i.e. most likely to be the cause of bugs, least able to adapt to new features, etc.). As such, one important legacy of the IST project was a realization that providing tools for generating ship models was very difficult and costly.

Following on the ISSMM project, Martec and DRDC collaborated on a Defence Industrial Research (DIR) project to explore leveraging a commercial ship model database to supply the vessel geometry for engineering analysis tools such as the IST. The goal was to choose a commercial off the shelf (COTS) product that could provide a structural modeling and storage (database) functionality that could then be accessed by a wealth of engineering analysis applications. As such, the commercial vessel database would serve as a single product model (SPM) and, if supplied during vessel design or production (e.g. by the ship builder), could be leveraged throughout the lifetime of the vessel to serve LCM requirements.

In parallel with the DIR project, another collaborative development was underway to enhance the geometric modeling capabilities of an application called the Submarine Structural Analysis Suite (SubSAS). The SubSAS application had a shared heritage with the IST in that both had used the same approach to generating vessel geometry (with the exception that SubSAS focused on submarines). Just as the IST had experienced problems with its geometry creation routines, so had SubSAS. However, because of an immediate requirement to support a modeling effort that was underway, an alternative modeling technology was devised for SubSAS. This technology was termed Relational Meshable Geometry Script (RMGScript).

RMGScript proved successful at supporting the modeling that was underway with SubSAS at the time. A generic offshoot of the SubSAS program, termed the Trident Modeller, was then developed to allow RMGScript to be used for ship models. The first full ship model created with

the Modeller was a mega-yacht, though subsequently a partial model of the CPF was devised for proof-of-concept purposes. Towards the latter half of the DIR project, consideration was given to whether or not the Trident Modeller fit within the LCM solution that was being proposed, and it was found to be useful primarily for global finite element mesh generation.

Martec Limited was acquired by Lloyd's Register (LR) at about the midpoint of the DIR project. This event presented the opportunity not only to broaden the LCM analysis types under consideration to include classification rules assessments, but to also consider leveraging existing LR software assets that were related to SPM model storage and translation. It turned out one application in particular was applicable – the Data Interface Management Engine (DIME). In short, the DIME application was concerned with obtaining ship data from commercial modeling packages and supplying it to LR design assessment packages. Although the goals were nearly identical to those of the DIR, limitations with the DIME data schema were identified stemming from the fact that it was solely focussed on the analysis types required by LR. Nonetheless, work within the latter part of the DIR project culminated in successfully linking both the DIME and Trident Modeller as a proof-of-concept example of a working solution to service both commercial and naval LCM requirements, albeit in a limited capacity.

This project continues the effort to develop a database solution capable of meeting the LCM requirements of DND. Specifically, this project will explore the database formats for both the DIME and Trident Modeller and assess them against the data required for a variety of LCM analysis types. As it is known that neither can fulfill the requirements on their own, a new database format will be proposed that builds on the strengths of each, and augments each as necessary. The specific outputs of this work will be a blueprint for implementing the first version of the database format, as well as guidelines for how to extend the format in a manner that allows it to evolve with minimal impact on applications that make use of it.

2 Data Requirements for Analysis

This Section details the identified data requirements for the various engineering analysis packages deemed to be of interest to DRDC as well as LR/Martec.

2.1 DND Engineering Analysis

The data requirements for all of the analysis types under consideration as part of this project were detailed in a 2006 report compiled as part of the joint DRDC/Martec DIR project to explore commercial tools for creation and storage of vessel models [1]. A summary of the data requirements for the analysis types, as given in [1], is included in Appendix A. For full details, the reader is directed to refer to the original report.

2.2 LR Rule Analysis

2.2.1 RulesCalc

Lloyd's Register provides software tools that allow for checking a ship design against the various ship rules that are required as part of ship classification. RulesCalc [3] is one such software tool. It can be used as a stand-alone tool or can be used in conjunction with some ship design systems. As most aspects of a ship design must be compared against the ship rules, data requirements can be quite extensive covering not only ship configuration, hull, design, scantling details, material specifications, but also such things as equipment details.

2.2.1.1 Ship Data

A variety of basic data are required by RulesCalc. Examples are:

Ship particulars (overall length, breadth, displacement, maximum speed, draft, etc.)

Ship registration data (type of ship, ship designation, hull designation, etc.)

Classification data (service area, etc.)

Structural arrangement (deck camber, bow details, transom details, etc.)

2.2.1.2 Geometric Data

RulesCalc does not make use of finite element or boundary element models. Instead, ship geometry is described in terms of profiles, cross sections and connection details.

Ship profile

Cross section scantling details including section properties, loading details, exposure to weather, etc.)

Connection details

Member profiles (stiffener cross section shapes)

2.2.1.3 Material Properties

RulesCalc requires only basic isotropic material properties suitable for basic strength calculations:

Material type and grade

Young's modulus

Poisson's ratio

Ultimate tensile strength

Yield stress

Density

2.2.2 ShipRight

ShipRight Structural Design Assessment (SDA) [4], unlike RulesCalc, provides design assessments using a finite element analysis-based approach. ShipRight provides a modeling tool to allow a geometric model to be created which is then meshed in a semi-automated manner to produce the analysis input. Geometric models are at a 'global' level of detail – i.e. major structure is included such as deck and bulkhead plating, stiffeners, cutouts, etc., but minute details are omitted.

ShipRight geometric models consist of some ship particulars, a hullform definition, and a variety of longitudinal and transverse structures.

2.2.2.1 Ship Data

The ship particulars required include:

General particulars (name, LR number, owner, etc.)

Operational information (condition, maximum speed, etc.)

Principal dimensions (length overall, length between perpendiculars, moulded breadth and depth, etc.)

2.2.2.2 Geometric Data

In terms of the geometry modeled, ShipRight's focus is on modeling the parallel midship compartments of various merchant vessel types (including oil tankers and bulk carriers). As such, the program provides interfaces for parametrically defining the variety of components that comprise these regions [7]. For example, there are interfaces to model the main deck, inner bottom, side stringers, double bottom floors, etc.

2.2.2.3 Material Properties

Material properties for the included structure must contain the following:

Grade

Strength level

Yield stress

Ultimate stress

Poisson's ratio

Young's modulus

Mass density

2.2.2.4 Stiffener Profiles

Stiffener profiles must be defined to include:

Cross sectional area

Centroid

Moment of inertia I_{xx}

Moment of inertia I_{yy}

Product of inertia I_{xy}

Moment of inertia $I_{xx'}$

Moment of inertia $I_{yy'}$

Principle axes angle

Torsion constant

2.3 Discussion

In summary, although there are similarities among the data required by each analysis type, there are also significant differences. In terms of similarities, it can be said that most analyses (perhaps with the exception of hydrodynamic analysis) require overall vessel geometry consisting of the hull, decks, girders, bulkheads, stiffeners, and major openings. All geometry must have associated data such as thickness and basic strength material properties. Building upon that, many analyses require geometry for appendages including the keel, rudder, propeller, sonar domes, masts, etc. Furthermore, some analyses require more specific features including connection details (detailed structural finite element analysis (FEA) for fatigue assessment), weapons, sensors, insulation and exhaust configuration (infrared signature), non-structural members such as engines and generators (magnetic signature) and location of tanks (acoustic signature).

All of the analyses, with the possible exception of detailed structural FEA, require a non-manifold surface representation of the vessel geometry, in contrast to a manifold solid representation. The former is typically modelled with triangular/quadrilateral shell elements in FEA, the latter with tetrahedral/hexahedral volumetric elements. Detailed structural FEA can be required to use either type, depending on a number of factors including the type of analysis being performed, the accuracy required, the size of the model, and even the preference of the analyst. For instance, a crack initiation analysis may require a high-fidelity volumetric mesh while a crack propagation analysis may be possible with a lower-fidelity shell mesh. Where volumetric meshes are required for some detailed FEA, it is reasonable to assume by definition they will be limited in their extent to only a small region of a vessel (i.e. a particular detail such as a cast penetration coaming or valve).

Some analysis types require inclusion of data relating to equipment (engines, generators, weapons, sensors, etc.). In most cases such equipment can be included in an idealized form, serving only to contribute its effect on the overall model in a simplified manner. For instance, an engine may only require its overall mass, location and attachment characteristics be specified instead of its actual shape. Furthermore, stiffening members can optionally be modeled in a simplified FE representation consisting of only a line with associated properties (i.e. a *beam* representation).

Virtually all of the analysis types require specialized input data particular to the specific analysis. Some of this data can be associated with model geometry (e.g. paint properties), while some are completely unrelated to geometry (e.g. ship speed, radar source). Some data are particular to a single analysis and some are used by many analyses. We consider such issues and recommend guidelines limiting the scope of what will be stored in the SPM database being proposed:

The database should store mainly data relating to the vessel itself – i.e. its geometry and physical properties. If a particular geometric feature of a vessel is not supported by the database, one can assume it is reasonable to add the feature using the same approach used by the wealth of existing geometric features, and hence it should be added by straightforward extension of the database. Likewise, if new data merely augments data belonging to an existing entity, then it too should be included by extending the database.

The database should mainly store data that do not change between analyses. Data that can be reused between many analyses is more likely a characteristic of the vessel, whereas ephemeral data particular to a single analysis (e.g., loads) is not. Storing the latter in the SPM database risks significantly complicating the schema and database size, risking reluctance to adopt or make use of the proposed solution. It is worth noting that some provision will have to be made for describing entities that allow for variable degrees of fill – e.g. tanks, stores, weapons stowage, etc. While these entities can change their fill level between analyses, they are, nonetheless, permanent characteristics of the model.

Analysis data falling outside of the scope of the SPM database will be provided by the particular analysis tools themselves. Most such tools have specialized interfaces that support rapid input of the highly specialized data that is particular to their concerns. In that way, they can be considered perhaps a better source for this data than the SPM itself. Indeed, even for data that is to be considered part of the SPM, some analysis tools may have very efficient interfaces for entering that data, and consideration should be made to using the tool in this role and providing a means to then supply the data back to the SPM.

The proposal set forth in this report will outline a set of guidelines for a database format that is capable of storing the most pertinent vessel metadata, structural entities and their associated properties, originating and exported from a third-party SPM system, as directed by the requirements of the analyses considered as well as the guidelines proposed above. As it is recognized that the format will undoubtedly have to grow, a rigorous extension mechanism will also be proposed to accommodate this growth in a controlled manner.

3 Existing Databases

Since the DIME and Trident Modeller have been identified as existing applications that partly meet the functionality required by the database to be proposed, they are studied in detail in order to demonstrate their implementation towards a potential contribution to the current proposal. As such, this Section discusses the current state of the DIME and Trident Modeller databases, focusing on the details, as well as historical motivation and evolution, of their designs.

3.1 Trident Modeller

The Trident Modeller was spawned as a generic offshoot of the SubSAS submarine modeller – both programs support the same geometric modeling capabilities, and differ only in their high-level analysis features. The common modeling technology upon which both programs are based was developed pragmatically to allow for the modeling of vessel geometry specifically to support global FEA. As such, emphasis is on modeling geometry pertaining to overall vessel strength, at a relatively coarse level of detail. Entities exist to model planar plates, conic plates, planar-web stiffeners and stiffeners following a non-planar trace curve.

The Trident Modeller geometric model is created using a script referred to as RMGScript [5]. RMGScript is an extensible markup language (XML) serialized domain specific language (DSL) for defining non-manifold geometric models in a parametric, relational manner. RMGScript allows geometry to be accurately and concisely defined in a very efficient manner. The fact that the file format is XML-based ensures it is human-readable, easily manipulated and highly portable.

Normally, RMGScript does not store geometry in a canonical form typical of most computer aided design (CAD) packages. Its *native* schema does not include entities for defining non-uniform rational B-spline (NURBS) curves and surfaces, nor high-level model topology (i.e. boundary representations, or breps, defining connections between active sub-portions of NURBS curves and surfaces). Rather, RMGScript defines objects in a parametric form that gets interpreted and converted to NURBS-based geometry by the application upon loading the file. As such, the RMGScript file can be likened to a script of instructions for building geometry rather than a database of actual geometry.

That said, RMGScript has an extension to its native schema (for purposes of distinction it will be referred to as the *extended* schema) that allows for canonical geometry and topology to be included in its model. This capability was developed to allow for including geometry exported from CAD packages within RMGScript models as a basis for creating native RMGScript geometry (e.g. import a CAD representation of a faired hull form but use RMGScript to define decks, bulkheads, etc.). The extended schema currently defines two entities – one for importing plating geometry stored in the industry-standard initial geometry exchange specification (IGES) format, and one for importing complex stiffeners with trace curves defined in IGES format.

3.1.1 The RMGScript Schema

The overall RMGScript DSL is defined in two parts. First, a typical XML schema defines a set of allowed elements and their corresponding attributes. These elements are interpreted by the Trident Modeller (the interpreter in this case) and converted to object instances within the application. The geometric constructs supported by the Trident Modeller correspond to elements in the XML schema, as do other object types such as material definitions, reference variables, etc.

The second half of the RMGScript language comprises a syntax that can be used when specifying XML element attribute values. As this syntax is used within the definition of attribute values, it exists apart from the XML schema – i.e. it is not validated by the XML parsing portion of reading in an RMGScript file (with the exception, of course, that the attribute value text cannot render the XML invalid – i.e. special characters must be escaped, etc.). It is still validated by the interpreter, however, which will catch any syntax errors present. This portion of the language does not define object instances, but rather allows for setting the characteristics of object instances.

While the RMGScript XML schema (i.e. entities and attributes) is a fairly straightforward example of using XML to serialize in-memory objects (i.e. objects to elements, properties to attributes), the main utility of RMGScript comes from the attribute value syntax. This syntax provides a variety of constructs that empower the language to be a highly efficient way to define fully relational geometric models. These constructs are described as follows.

3.1.1.1 Object Reference Syntax

Object reference syntax is used in RMGScript to allow object properties to be set with reference to other objects and their properties. Leveraging this syntax not only streamlines modeling, but also enables a model to be built relationally. Simply put this means that dependency relationships between objects are maintained by the system and used to recursively propagate changes made to a given object to all of its dependants (i.e. change object x and have various other dependent objects react to that change). Objects reference syntax consists of wrapping an object name in square brackets (e.g. “[other object name]”), specifying that the system should resolve (look up) that object when the model is being parsed.

3.1.1.2 Attribute Reference Syntax

In addition to being able to refer to existing objects, RMGScript also provides a mechanism to refer to particular attributes of existing objects. In this case the object reference syntax can be extended using the dot operator and referring to a particular attribute on the referenced element (e.g. “[other object name].other_attribute”). The dot operator can be sequenced to refer to nested attributes as well (e.g. “[My Object].pl.o.x to refer to the x component of the origin of the plane belonging to “My Object”).

3.1.1.3 Anonymous Object Syntax

As a modeling convenience, RMGScript supports the inline creation of objects within the definition of other objects. This effectively results in a temporary instance of an object being created to be used in helping define another. The temporary object is referred to as an

anonymous object because it has no name, does not persist beyond its use in defining another object, and cannot be referenced using the reference object syntax. Anonymous object syntax involves wrapping an anonymous object type declaration in curly brackets (e.g. "{ANONOBJ(PARAM1,PARAM2)}"). Anonymous object declarations can be nested and can include references to other objects (e.g. "{FWDPLANE({POINT([FR12].x,0,0)})}").

3.1.1.4 Inline Math Operators

RMGScript supports the use of simple inline mathematical operators (i.e. +, -, *, /) wherever a value type is expected. This often allows values to be entered "as read" from design drawings rather than requiring the user to pre-calculate resultant values before input. Not only does this speed modeling by avoiding a manual calculation step, but it promotes defining objects relationally (i.e. very easy to relate one value to another with some offset), and makes a model much easier to verify. The mathematical operators not only apply to scalar values, but can also be applied to points and vectors, effectively supporting simple vector arithmetic.

3.1.1.5 Inline Units Specification

RMGScript also allows units to be specified explicitly inline with value declarations. This has the benefit of allowing the modeller to enter values directly as they are specified in drawings rather than doing any manual unit conversion. This is especially helpful if design drawing switch units in specific places. The supported units postfix operators are "in", "f", "mm" and "m" indicating inches, feet, millimetres and meters respectively. For example, to specify a value in inches, one would enter v="56.5in". The value would then be converted to the default unit system being used by the model (mm).

3.1.2 Example

An example RMGScript model script is shown in Listing 1. This script defines a small region of a ship model – i.e. a single deck and a few transverse and longitudinal stiffeners. The ship hull form is imported from an IGES file using an IGESEntity, a member of the extended RMGScript schema. The remainder of the model is defined relating to the hull form using object reference syntax (e.g. the Trim entity on line 20). Numerous examples of anonymous object syntax are shown (e.g. lines 18, 21, 22). The geometry resulting from loading this RMGScript file into Trident Modeller is shown in Figure 1.

```
1  <Submarine name="Simple Example" tol_comp="1" tol_calc="0.1" units="mm">
2
3    <Collection name="Materials">
4      <Material_Iso name="Steel" YoungsModulus="207000"
5                    PoissonsRatio="0.3" Density="7.85e-09" YieldStress="310"/>
6    </Collection>
7
8    <Collection name="Sections">
9      <Section_T name="150x100T" d="150" tw="10" wf="100" tf="20"/>
10   </Collection>
11
12   <IGESEntity name="Hull" igesfile="Hullform.iges" m="[Steel]"
13               t="25.4" ashierarchy="1"/>
14
15   <Collection name="Deck 1 Group">
16
17     <Collection name="Deck 1 Plates">
18       <PlanarPlate name="Deck 1 Plate" pl="{DOWNPLANE(10000)}"
19                    t="12.7" m="[Steel]">
20         <Trim objref="[Hull]"/>
21         <Trim objref="{AFTPLANE(100000)}"/>
22         <PlateRefPt p="{POINT(50000,0,10000)}"/>
23       </PlanarPlate>
24     </Collection>
25
26     <Collection name="Deck Beams">
27       <PlanarWebStiffener name="DB 1" pl="{AFTPLANE(40000)}" e1="[Hull]"
28                           e2="[Hull]" s1="[150x100T]" m="[Steel]">
29         <AttachBase objref="[Deck 1 Plate]" normside="1"/>
30       </PlanarWebStiffener>
31       <PlanarWebStiffener name="DB 2" pl="{AFTPLANE(45000)}" e1="[Hull]"
32                           e2="[Hull]" s1="[150x100T]" m="[Steel]">
33         <AttachBase objref="[Deck 1 Plate]" normside="1"/>
34       </PlanarWebStiffener>
35       <PlanarWebStiffener name="DB 3" pl="{AFTPLANE(50000)}" e1="[Hull]"
36                           e2="[Hull]" s1="[150x100T]" m="[Steel]">
37         <AttachBase objref="[Deck 1 Plate]" normside="1"/>
38       </PlanarWebStiffener>
39       <PlanarWebStiffener name="DB 4" pl="{AFTPLANE(55000)}" e1="[Hull]"
40                           e2="[Hull]" s1="[150x100T]" m="[Steel]">
41         <AttachBase objref="[Deck 1 Plate]" normside="1"/>
42       </PlanarWebStiffener>
43       <PlanarWebStiffener name="DB 5" pl="{AFTPLANE(60000)}" e1="[Hull]"
44                           e2="[Hull]" s1="[150x100T]" m="[Steel]">
45         <AttachBase objref="[Deck 1 Plate]" normside="1"/>
46       </PlanarWebStiffener>
47     </Collection>
48
49     <Collection name="Deck Longitudinals">
50       <PlanarWebStiffener name="DL 1" pl="{PORTPLANE(-3000)}" e1="[Hull]"
51                           e2="{AFTPLANE(100000)}" s1="[150x100T]" m="[Steel]">
52         <AttachBase objref="[Deck 1 Plate]" normside="1"/>
53       </PlanarWebStiffener>
54       <PlanarWebStiffener name="DL 2" pl="{PORTPLANE(-1500)}" e1="[Hull]"
55                           e2="{AFTPLANE(100000)}" s1="[150x100T]" m="[Steel]">
56         <AttachBase objref="[Deck 1 Plate]" normside="1"/>
57       </PlanarWebStiffener>
58       <PlanarWebStiffener name="DL 3" pl="{PORTPLANE(0)}" e1="[Hull]"
59                           e2="{AFTPLANE(100000)}" s1="[150x100T]" m="[Steel]">
60         <AttachBase objref="[Deck 1 Plate]" normside="1"/>
61       </PlanarWebStiffener>
62       <PlanarWebStiffener name="DL 4" pl="{PORTPLANE(1500)}" e1="[Hull]"
63                           e2="{AFTPLANE(100000)}" s1="[150x100T]" m="[Steel]">
64         <AttachBase objref="[Deck 1 Plate]" normside="1"/>
65       </PlanarWebStiffener>
```

```
      <PlanarWebStiffener name="DL 4" pl="{PORTPLANE(3000)}" e1="[Hull]"
                          e2="{AFTPLANE(100000)}" s1="[150x100T]" m="[Steel]">
        <AttachBase objref="[Deck 1 Plate]" normside="1"/>
      </PlanarWebStiffener>
    </Collection>

  </Collection>

</Submarine>
```

Listing 1 – RMScript example Script File.

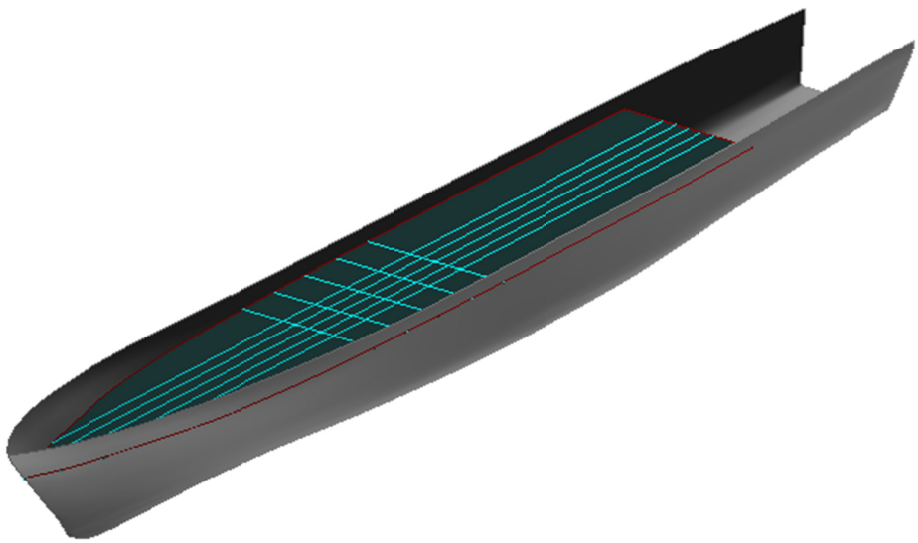

Figure 1: The RMGScript example displayed in Trident Modeller.

The full list of RMGScript entities as well as anonymous object types, etc. is included in Annex A.

3.2 DIME

The Data Interface Management Engine (DIME) [2] is part of the LR Interface Toolkit designed to allow users to move data from their preferred commercial ship design package to the various Lloyd's design assessment tools (e.g. RulesCalc, ShipRight). The DIME consists of an XML data schema and collection of XML transformation stylesheets (XSLT) as well as a graphical user interface (GUI) application. The DIME application imports and stores its data in an XML file formatted according to its schema and uses a combination of the transformation stylesheets as well as custom routines to transform data for input into the LR design assessment tools. The GUI application allows the data being transferred from the commercial ship design packages to the LR design assessment tools to be validated and augmented with any required data that might be missing. The intended flow of data through the DIME is shown in Figure 2 (from [2]).

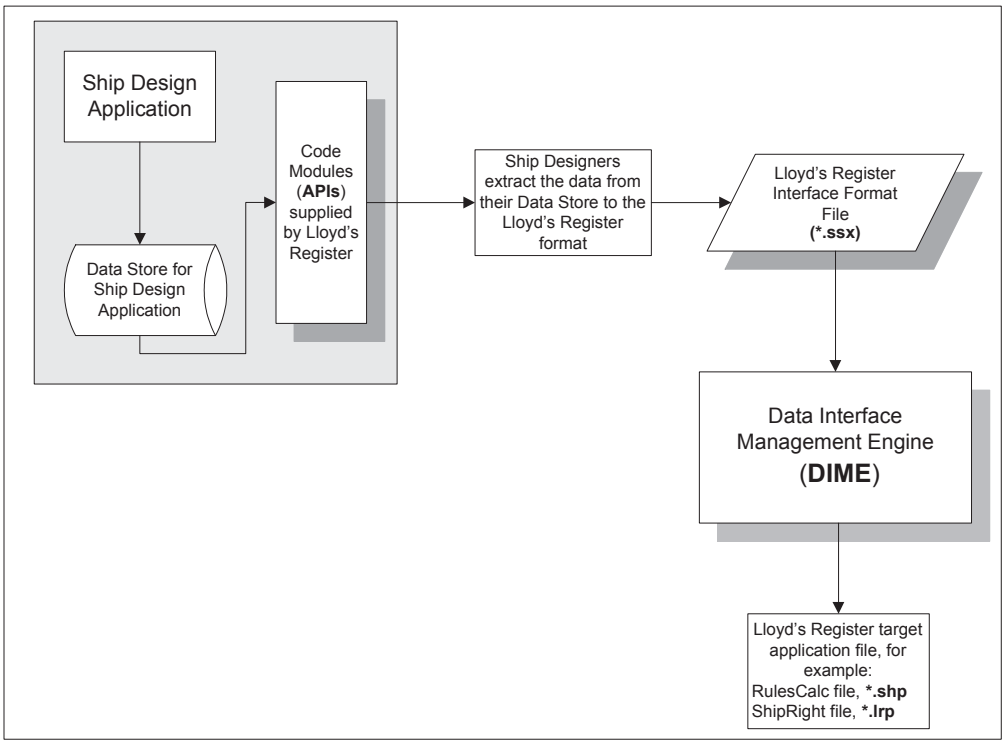

Figure 2: The DIME process flow.

The DIME data format (stored in ship structure exchange or SSX files) evolved from the specific data requirements of RulesCalc and ShipRight. In its earliest form, it mainly dealt with transverse sections of ships, as that was of main concern to RulesCalc in particular. The transverse section could be built up within the application, or be automatically extracted and populated while importing data from an SPM (e.g. Napa, Tribon or SmartMarine) or CAD application. Longitudinal structure could be defined by extruding transverse structure and defining strakes,

stiffener groups, etc., and collectively both representations were used to populate input files for RulesCalc and ShipRight.

Subsequently, however, an alternative representation for geometry was proposed that was more generic and capable of storing both transverse and longitudinal structure. This representation became known as "assets and instances" (see [6]), as one of its main tenets was that geometric assets could be defined once and reused (by applying geometric transformations) in numerous locations throughout a vessel. While this shared instancing methodology was perhaps never fully realized, the asset schema was adopted as the new means to define all 3D geometry (transverse and longitudinal) in DIME. It is this schema that will be the focus for the remainder of this Section.

3.2.1 The DIME Schema

The top level entities in the DIME schema are shown in Figure 3.

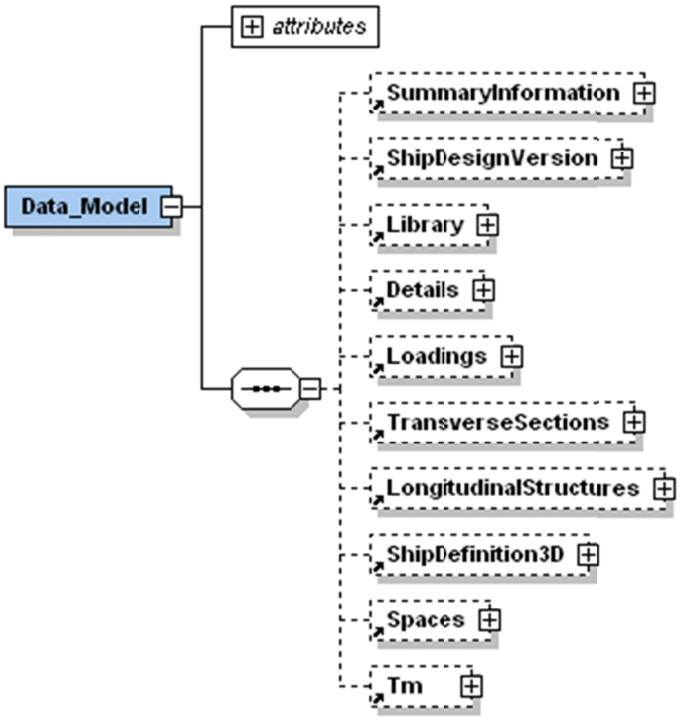

Figure 3: The DIME schema top level entities.

The SummaryInformation and ShipDesignVersion entities each contain metadata pertaining to the model file such as its creation date, originating application and schema version. The Details

entity contains general data pertaining to the ship such as the principal particulars, ownership, build and classification data, and some global model data such as frame spacing. The Library entity groups definitions for materials, stiffener profiles, end connections, openings and brackets. Presently the schema supports steel and aluminium materials and various stiffener profiles including flatbar, tee, angle, bulb and general sections. End connections, openings and brackets are supported through a list of pre-configured, parameterized types (though openings and brackets also support entering general shapes).

The key entity containing the geometric definition of the ship is the ShipDefinition3D entity, shown in Figure 4 (one might be inclined to think TransverseSections and LongitudinalStructures form part of the definition, but instead these entities are used to set up data for passing on to RulesCalc; they will not be discussed further). The ShipDefinition3D entity consists of two categories of data referred to as *assets* and *instances*. Roughly speaking, assets are meant to define a dictionary of common structural configurations that occur repeatedly within a vessel. Assets are intended to be combined with a locating transformation into instances of geometry comprising the actual vessel model. As such, assets can be considered building blocks that do not really exist until they are instanced in a specific location within the model.

The motivation for separating assets and instances in this manner was mainly to minimize the size of DIME data files. It was thought that while the assets are heavy-weight entities (i.e. lots of defining parameters), the instances could be kept lightweight by simply referring back to the assets for their shape definition. As XML is, by its nature, a verbose file format, it was thought this approach could save significant storage and open up the possibility of using the files in, for example, web services-based application scenarios.

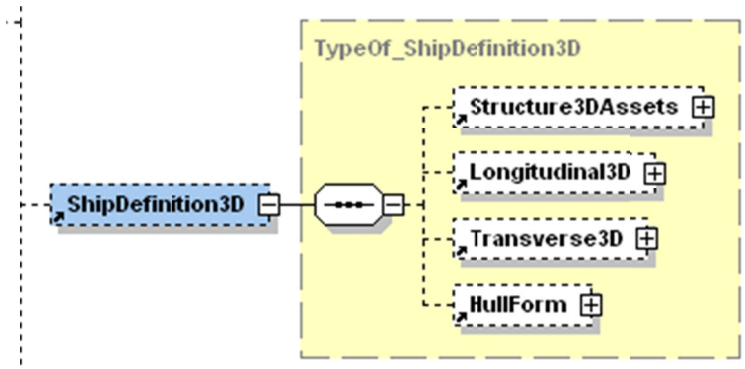

Figure 4: The ShipDefinition3D entity.

Within the ShipDefinition3D entity, assets are aggregated in the Structure3DAssets entity as a collection of Structure3D entities (Figure 5). The geometric definition of the asset is defined by the Boundary sub-entity (Figure 6). This entity supports a mostly canonical definition of the geometry consisting of a series of 3D NURBS curves defining the exterior boundary of the object (Contour3D). In most cases, the boundary curves are collectively planar, leading to a planar plate definition, but a nonplanar collection of boundary curves can also be used that results in a ruled surface shape. Optionally, references to other plates or "pseudo lines" (also 3D NURBS curves)

can be used to augment the boundary curves that trim the plate (these are captured by the ReferencePanels entity).

The geometric asset defined by the Structure3D entity and its Boundary child can be subdivided into a collection of plates. The plate boundary definition is nearly the same as the definition of the overall asset's boundary; except that it does not support references to other plates or pseudo lines as trim boundaries (i.e. it just supports the Contour3D entity). The plates, stored in the Plate3DCollection, are where the physical properties (i.e. the material and thickness) are assigned to the model (the material by referencing entities in the Library collection). The asset may also contain penetrations or support stiffeners or brackets. These are captured by the Hole3DCollection, Stiffener3DCollection and Bracket3DCollection, respectively. Penetrations and brackets reference shapes in the Library collection and position themselves on the asset using a vector transform. Stiffeners are positioned using another Contour3D NURBS curve (i.e. representing the curve of intersection between the stiffener and the asset) and refer back to the Library for their profile and material.

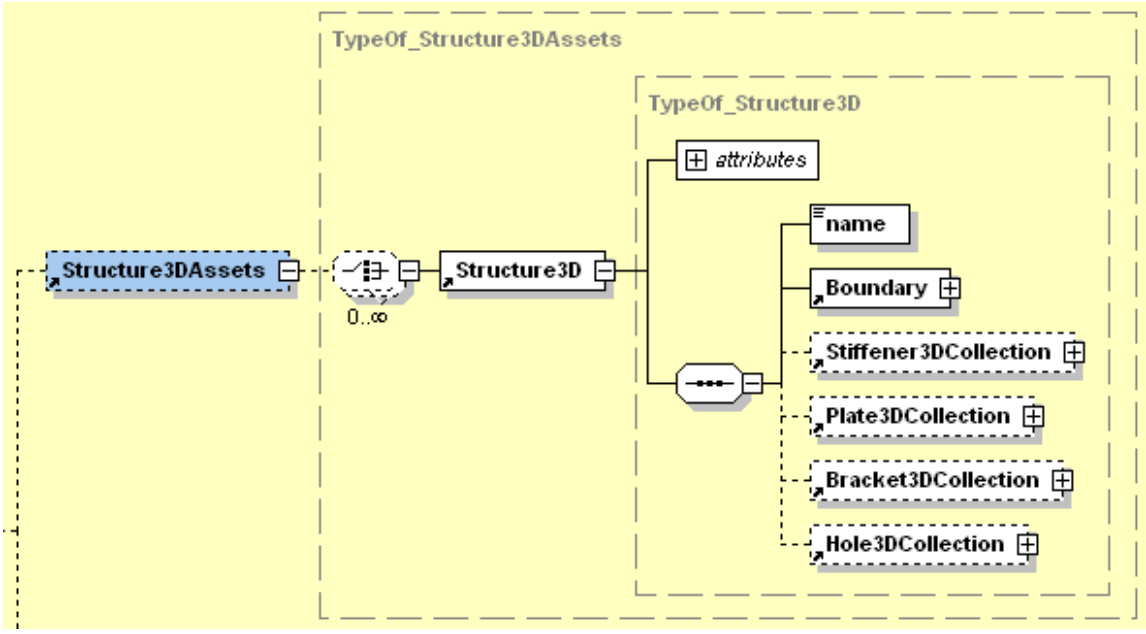

Figure 5: The Structure3DAssets entity.

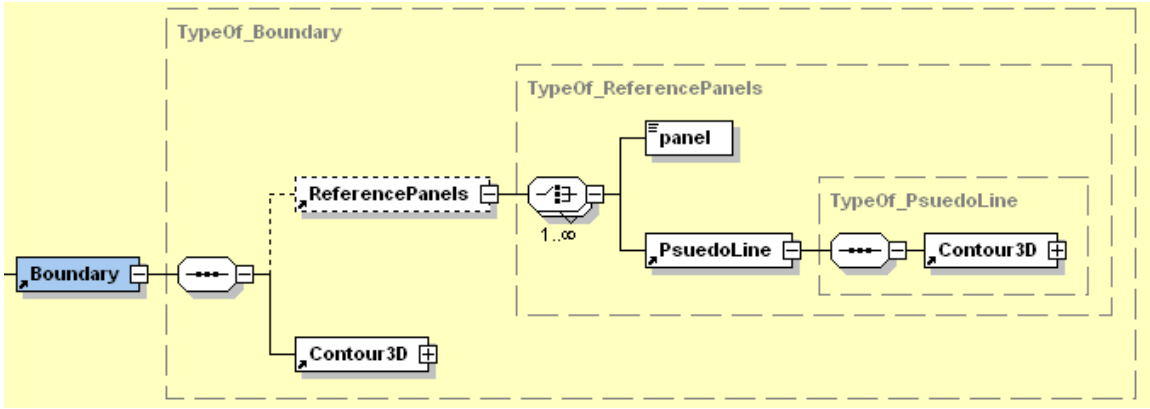

Figure 6: The Boundary entity.

Instances are kept below the Longitudinal3D and Transverse3D child entities of the Structure3DAssets entity (Figure 7 shows the Trans3DInstance; the Long3DInstance is analogous). As intended, instances are lightweight objects relative to assets. The refStruct entity ties the instance back to the asset definition by name, and the Transform entity implements the locating transformation (it is worth noting that simple vector translation is the only transformation presently supported; one could envision rotations and possibly even scaling to be also useful). The Plates collection allows plate function to be set per instance rather than on the overall asset, affording a bit more flexibility to reuse assets in different contexts at the potential expense of repeatedly entering the same values if all instances share the same function.

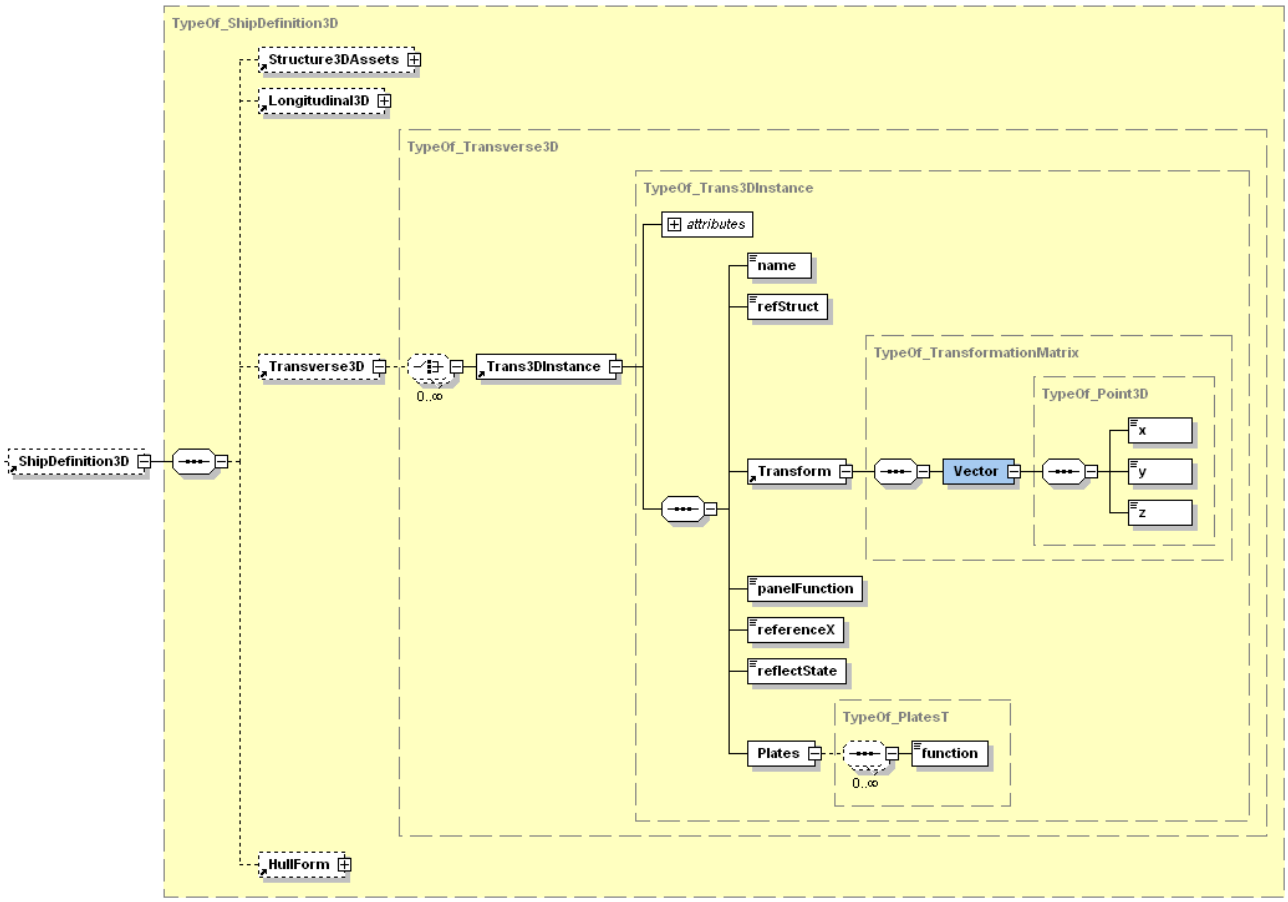

Figure 7: The Trans3DInstances entity.

A few other noteworthy entities in the schema include:

- The HullForm child entity of the ShipDefinition3D – this would allow a hullform to be built up based on standard lines of form. The entity supports defining plates and stiffeners on the hullform using the same approach as used by the Structure3D assets.

- The Spaces child entity of the root Data_Model entity – this is used to define compartments and spaces to be specified to rules applications. This entity allows volumes to be bounded either by referencing other geometric entities, or by building up bounding surfaces using Contour2D/3D objects.

- The Loadings child entity of the root Data_Model entity – supports specifying loading condition data to be used by the rules applications.

3.2.2 Example

An example of a populated sub-portion of a DIME data file is shown in Listing 2. In this excerpt, part of the ShipDefinition3D entity is shown, namely a single asset and instance representing a double bottom floor plate (encircled in Figure 8). The asset defines the overall geometry using both a Contour3D object as well as a collection of panel references. The asset is then divided into two sub-plates (port and starboard). Stiffeners are also specified (only one shown in the listing). The instance refers back to the asset by name (line 85) and assigns a panel function. The transform vector is 0, reflecting the fact that the asset is already positioned in its actual location (i.e. it will not be referred to by multiple instances in this case).

```
1   <ShipDefinition3D>
2
3     <Structure3DAssets>
4
5       <Structure3D id="TP_47B5667D_0000000E">
6
7         <name>TP_47B5667D_0000000E,BASIC-FX130_1</name>
8
9         <Boundary>
10          <ReferencePanels>
11            <panel>TP_47B5667D_0000000D</panel>
12            <panel>TP_47B5667D_00000003</panel>
13            <panel>TP_47B5667D_0000000D</panel>
14          </ReferencePanels>
15          <Contour3D>
16            <order>2</order>
17            <CPList3D>
18              <NPt3D> <x>130000</x> <y>0</y> <z>0</z> </NPt3D>
19              <NPt3D> <x>130000</x> <y>15930</y> <z>0</z> </NPt3D>
20              <NPt3D> <x>130000</x> <y>15930</y> <z>2299.5</z> </NPt3D>
21              <NPt3D> <x>130000</x> <y>-15930</y> <z>2299.5</z> </NPt3D>
22              <NPt3D> <x>130000</x> <y>-15930</y> <z>0</z> </NPt3D>
23              <NPt3D> <x>130000</x> <y>0</y> <z>0</z> </NPt3D>
24            </CPList3D>
25          </Contour3D>
26        </Boundary>
27
28        <Stiffener3DCollection>
29          <Stiffener3D id="STF_47B5667D_000000C5">
30            <profile>PRO_47B56663_00000002</profile>
31            <material>MAT_47B56663_00000001</material>
32            <Contour3D>
33              <order>2</order>
34              <CPList3D>
35                <NPt3D> <x>130000</x> <y>840</y> <z>2040</z> </NPt3D>
36                <NPt3D> <x>130000</x> <y>840</y> <z>450</z> </NPt3D>
37              </CPList3D>
38            </Contour3D>
39            <angle>90</angle>
40            <flip>true</flip>
41          </Stiffener3D>
42        </Stiffener3DCollection>
43
44        <Plate3DCollection>
45          <Plate3D id="TRP_47B5667D_0000002C">
46            <thickness>13</thickness>
47            <material>MAT_47B56663_00000001</material>
48            <Contour3D>
49              <order>2</order>
```

```
50              <CPList3D>
51                <NPt3D> <x>130000</x> <y>0</y> <z>0</z> </NPt3D>
52                <NPt3D> <x>130000</x> <y>15930</y> <z>0</z> </NPt3D>
53                <NPt3D> <x>130000</x> <y>15930</y> <z>2299.5</z> </NPt3D>
54                <NPt3D> <x>130000</x> <y>0</y> <z>2299.5</z> </NPt3D>
55                <NPt3D> <x>130000</x> <y>0</y> <z>0</z> </NPt3D>
56              </CPList3D>
57            </Contour3D>
58          </Plate3D>
59          <Plate3D id="TRP_47B5667D_0000002D">
60            <thickness>13</thickness>
61            <material>MAT_47B56663_00000001</material>
62            <Contour3D>
63              <order>2</order>
64              <CPList3D>
65                <NPt3D> <x>130000</x> <y>-19.5</y> <z>2299.5</z> </NPt3D>
66                <NPt3D> <x>130000</x> <y>-15930</y> <z>2299.5</z> </NPt3D>
67                <NPt3D> <x>130000</x> <y>-15930</y> <z>0</z> </NPt3D>
68                <NPt3D> <x>130000</x> <y>-19.5</y> <z>0</z> </NPt3D>
69                <NPt3D> <x>130000</x> <y>-19.5</y> <z>2299.5</z> </NPt3D>
70              </CPList3D>
71            </Contour3D>
72          </Plate3D>
73        </Plate3DCollection>
74
75      </Structure3D>
76
77    </Structure3DAssets>
78
79    <Transverse3D>
80
81      <Trans3DInstance id="TRI_47B5667D_0000000E">
82
83        <name>BASIC-FX130_1</name>
84
85        <refStruct>TP_47B5667D_0000000E</refStruct>
86
87        <Transform>
88          <Vector xs:type="Origin">
89            <x>0</x>
90            <y>0</y>
91            <z>0</z>
92          </Vector>
93        </Transform>
94
95        <panelFunction>Double Bottom Floor (watertight)</panelFunction>
96
97        <referenceX>0</referenceX>
98
99        <reflectState>As defined</reflectState>
100
101       <Plates>
102         <function/>
103         <function/>
104       </Plates>
105
106     </Trans3DInstance>
107
108   </Transverse3D>
```

Listing 2 – DIME example Data File (Excerpt).

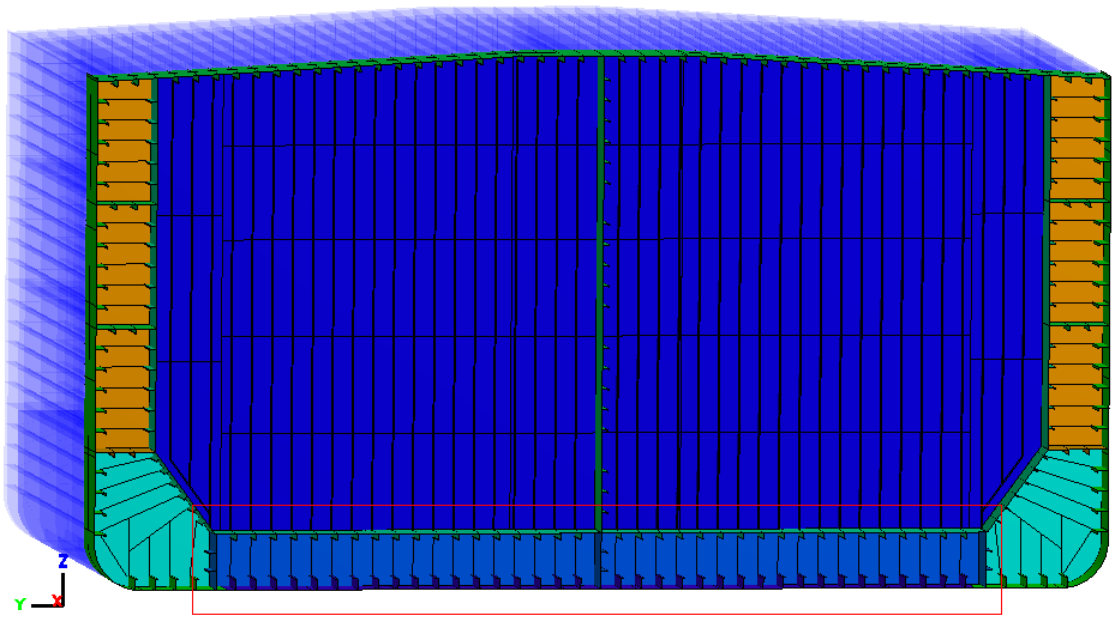

Figure 8: DIME example geometry in DIME application.

4 Translating from SPM Into Existing Databases

Both Trident Modeller and DIME currently support importing models from external sources (DIME more so than Trident Modeller at this point). This Section discusses the topic of data import into these applications from the standpoint of what is possible and makes sense. The discussion is flavoured, but not restricted, by the current approaches and their associated limitations. As in the previous Section, it is anticipated that study of the existing applications can yield insight that will contribute to our final proposal.

4.1 Trident Modeller

As mentioned in Section 3.1, RMGScript has two parts to its overall schema – the native schema and the extended schema. The former provides a rich set of entities for defining relational geometry in a parametric manner. For instance, the PlanarPlate entity allows for creating a trimmed planar plate by specifying only its plane, physical properties and trim boundaries. This can be done using the scripting language or using graphical interfaces in the Trident Modeller itself. The geometric entities supported by the native schema allow modeling the majority of the macro-structure found in ships, but by no means all structure. Indeed, mainly only common structure that is easily parameterized is included. As such, the native schema is easily populated, read and modified by end users.

The extended schema, on the other hand, contains entities that describe geometry in a canonical manner. These entities were conceived to provide support in Trident Modeller for non-simple geometry such as non-analytical curves and surfaces that are not available in the native schema entities. Instead of a few simple high-level parameters, these entities store "raw" data consisting of the mathematical definition of the geometry, which can easily consist of a few dozen to a few hundred high precision numbers. As such, these entities are virtually impossible to modify by manually editing data files, nor does Trident Modeller provide a graphical interface for modifying them. Once they reach Trident Modeller, the entities are effectively treated as "frozen" geometry.

With these two branches of the overall RMGScript schema, one can envisage three options for importing data from an SPM into the Trident Modeller. One approach might be to aim at converting only to the native schema, another to convert to the extended schema, and the third some combination of both. The details of each approach along with associated benefits and drawbacks follows.

4.1.1 Converting to the Native Schema

Converting strictly to the native RMGScript schema would amount to the exporting application resolving the inevitable mismatch between its own entity representation of vessel geometry and that of RMGScript. This would likely be achieved with heavy-weight, custom-coded conversion routines or data transformation scripts. Either approach would require intimate knowledge of the model being converted, both what data it is comprised of as well as potentially the approach used by the modeling application to generate and store it.

Having an external application export data exclusively to the RMGScript native schema would, however, be the ideal situation from Trident Modeller's point of view. Having a model consisting entirely of native entities fully realizes the benefits that RMGScript provide, namely a concise, parametric and relational model definition that is transformed into the Modeller's native geometric representation when the script is read into the application. Having the geometry in its native representation minimizes the likelihood that the Modeller will run into problems processing it downstream (e.g. equivalencing and meshing).

Unfortunately, there are a number of reasons it is difficult or impossible to convert all SPM data into only the RMGScript native schema. Firstly, it places a high burden on the exporting SPM to convert its own entities into RMGScript entities at the *parametric* level. The SPM's parameterization of a given structural component may be quite different from RMGScript's, and if the conversion is to be automatic, then sophisticated routines would need to be devised to resolve the mismatch. Indeed the SPM may not define its entities parametrically in the first place (i.e. some use CAD-like tools to create geometry), in which case the task is made all the more difficult. Secondly, the SPM may not be able to set up the RMGScript object dependencies that lead to a highly relational model. This could be because the SPM simply does not store such relationships (not all modelling packages are relational), or because its own representation is too difficult to convert to RMGScript's. Finally, it may well be that RMGScript lacks a native entity to represent a piece of structure present in the SPM, in which case it would be impossible to achieve the conversion without augmenting the RMGScript schema. Then, one has to draw a line on the complexity of such new entities lest they be better suited to the extended schema in the first place (i.e. perhaps by definition, the native schema is simply incapable of storing all structure).

4.1.2 Converting to the Extended Schema

Converting to the extended RMGScript schema would require the exporting application to export its geometry in a canonical format that could be recognized by Trident Modeller. Presently the Modeller uses the IGES format for such purposes, but other formats could be supported by purchasing off-the-shelf plug-ins to the Modeller's geometry kernel. Viable alternative formats include the Standard for the Exchange of Product Model Data (STEP), the ACIS geometry kernel's SAT format, Rhinoceros 3D's OpenNURBS format or the Parasolid geometry kernel's XT format.

Exporting the geometry to one of these formats does not provide an entire solution, however, in the context of using the data for engineering analysis. Physical properties such as plate thicknesses, material properties and stiffener scantling properties must also be passed on. These data could be accepted for each entity in the extended RMGScript schema, and could be populated by the routines in the exporting application. One would have to beware, using such an approach, that each entity's extent was defined in such a way that it could be augmented with a single value for each property type (i.e. a single material, single thickness, etc.). This might require extra processing on the exporting side.

Converting to the extended schema is the simplest solution from the point of view of the exporting application. It is a virtual certainty that the application would already support export to one of the abovementioned standard geometry formats, so little customization work (mainly just coming up with a way to associate physical properties with exported geometric entities) would be

needed. In addition, it is hard to conceive of a case where geometry present in the exporting application could not be exported using this approach (the industry-standard canonical file formats are mature and well-used). Furthermore, on the importing side, the task would be easier as well since all imported geometry could be processed in a generic way (i.e. no need for many different handlers for parametric definitions).

Unfortunately however, from the Trident Modeller viewpoint, this approach is the least desirable. As stated above, geometry that is supplied in canonical format is considered frozen to Trident Modeller – it cannot be modified in any way. Having a model consisting entirely of frozen geometry rules out realizing most of the benefits of RMGScript (i.e. relational, easily modified, etc.). In addition, although the geometry was in all likelihood optimally represented in the geometry kernel of the exporting application, when read into Trident Modeller, it may not be in an optimal form for its own kernel. This may lead to difficulties with downstream operations. Barring that, imported canonical geometry can, however, be augmented with native RMGScript geometry, and it can be passed through the Trident Modeller meshing process.

4.1.3 Converting to a Combination of the Native and Extended Schema

Converting entirely to the native or extended RMGScript schemas represents two extreme approaches. One is difficult to achieve on the export side and yields the most benefits on the import side and the other is easy to implement on the export side and limiting on the import side. A more practical solution is to strive to achieve something in between. Although there are many ways to strike such a compromise, one is particularly obvious.

Because the complexity of defining geometry typically increases with its *order* (or *degree*), a natural compromise would be to aim to convert all low-order (planar/linear) geometry to native RMGScript entities and all high-order (nonplanar/nonlinear) geometry to the extended schema. It is reasonable to suggest that low-order geometry can be succinctly represented parametrically, but high-order geometry is more suited to a canonical representation. For example, planar decks and bulkheads and linear stiffeners can easily be represented by RMGScript parametric entities, whereas complex hullforms and hull stiffeners are best stored using more complicated mathematical definitions such as NURBS trimmed surfaces and curves. Transferring low order entities in this way would still require some logic in the exporting routines to set up trim boundaries, but while tricky, it would perhaps not be too difficult as they are usually built up simply by referring to other objects.

It is worth noting a subtle point with this approach. Just because the RMGScript native schema is preferred for low-order geometry that can be described parametrically, it is not restricted to creating only low order geometry. High order geometry can and does still come about from entities in this schema. For example, there are various entities that parametrically define high-order geometry in the first place (so-called analytical shapes such as cylinders and domes). In addition, high order geometry can arise as a result of standard operations on low-order geometry such as trimming and projecting it with other high-order geometry. As such, it should not be thought that the appearance of any high-order geometry whatsoever would necessitate use of the extended schema.

Indeed, this strategy most closely matches the approach currently used by Trident Modeller. Non-planar ship hullforms, camber decks, superstructure and submarine casing geometries are

imported into the program using the IGESEntity member of the extended RMGScript schema. Hull longitudinal trace curves are also imported using the IGESTraceStiffener entity. Most other geometry, including decks, bulkheads, stiffeners, penetrations, etc. is then modeled using native schema entities (this being the main limitation of the current implementation – native schema entities are modeled instead of imported). Complex models of ships have been created using the Modeller in this way (see Figure 9 (CPF) and Figure 10 (mega-yacht)).

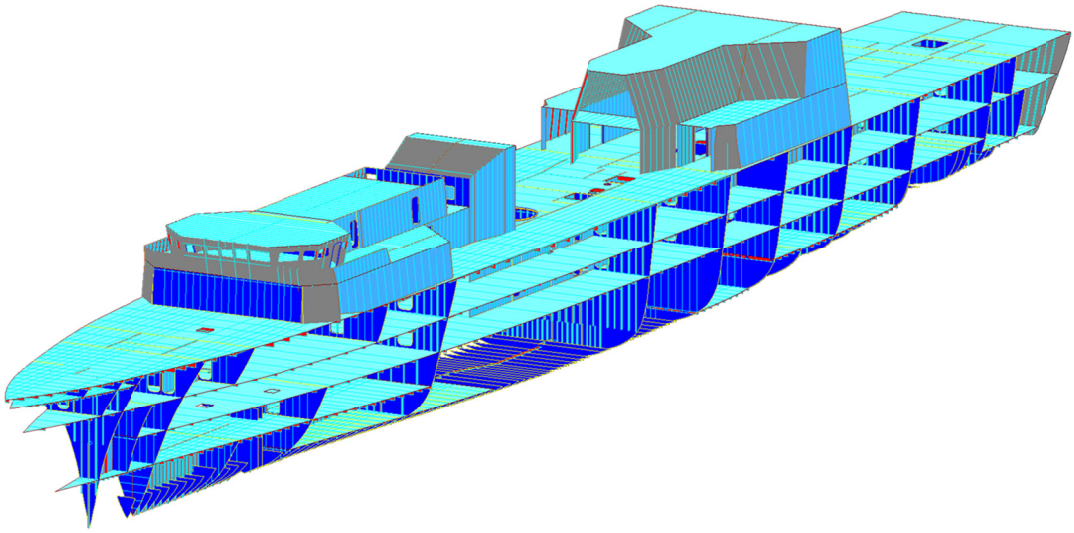

Figure 9: RMGScript model of the CPF.

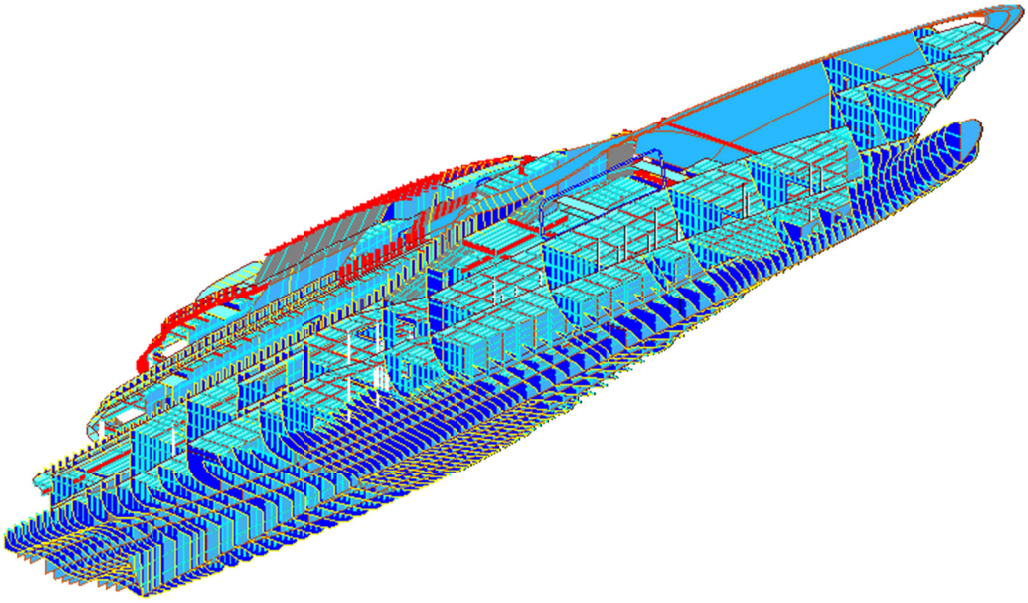

Figure 10: RMGScript model of a mega-yacht.

4.2 DIME

The intended mechanism to transfer data into DIME is via a published application programming interface (API) consisting of routines that populate a file according to its data schema. This API would allow commercial ship design package vendors to write export routines that pass data from their specific package into a well-formed data file to be opened by DIME. There are numerous advantages to this approach. From the point of view of the team on the exporting side, it is typically more convenient to code export routines against an API than to attempt to populate formatted files (although XML makes this much easier). The API also provides an extra layer that can shield adopters (on the export side) from any changes that arise to the specific file format (though, in theory, parts of the API could also change).

In practice, however, few, if indeed any, vendors have made use of the DIME API. Instead, vendors have supplied LR with schemas for how they export data from their tools, and LR has written DIME import routines for those schemas. These import routines take the form of autonomous modules (dynamic link libraries or DLLs) that can be plugged in to the main DIME interface. Most such modules consist of custom-coded algorithms to process and convert the vendor-specific files into the DIME data schema. Some of the modules, however, make use of standard XSLT to perform part of the conversion. XSLT allows pulling data from one XML file and writing it to another, effectively transforming all or part of a set of data from one (XML) format into another. XSLT is a good fit for this purpose, particularly in cases where vendors

export data to XML files instead of other propriety formats. See Figure 11 for an example of a SmartMarine model of the Maritime Coastal Defence Vessel (MCDV) imported into DIME using DIME's custom SmartMarine import DLL.

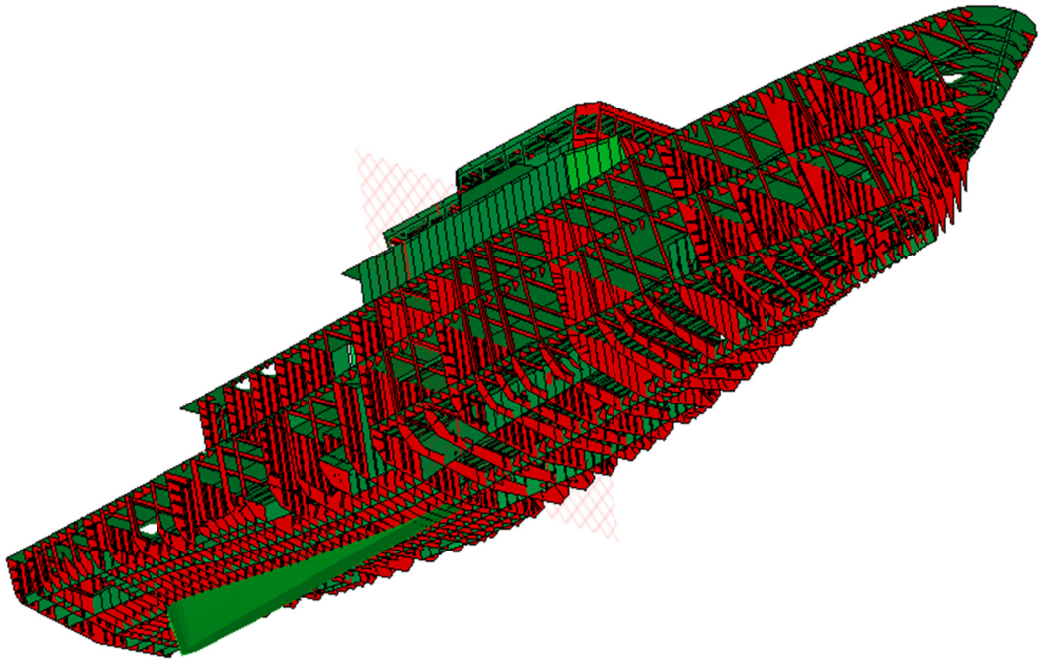

Figure 11: DIME model of the MCDV imported from SmartMarine.

The DIME schema shares some similarities with RMGScript, and as such, some of the translation issues would be the same. For instance, the DIME schema has parametric definitions for various entities (brackets, openings, etc.), necessitating the exporting application populate these entities at the parametric level. Also, the schema supports object references for trimming plates, so the exporting application has to have a way to set these references up from its own (if it stores any).

On the other hand, there are various fundamental differences between RMGScript and the DIME format which lead to translation challenges particular to the latter. Perhaps most importantly, the DIME format provides the assets and instances mechanism for re-using object shape definitions. To properly leverage this, the exporting application would have to be able to identify collections of like structure to become candidates for assets, and properly build the model from asset instances. This could be a trivial matter if the exporting application employed a similar internal dichotomy, but might be quite difficult if not.

As the DIME schema supports inline canonical definitions of some geometry, this too presents unique challenges. While it is perhaps quite easy for the exporting application to populate this data, potential difficulties arise on the importing side of things. The file format has to store

values with enough precision to ensure the geometry is accurately reproduced upon loading. In a text-based file format this can sometimes be a challenge when dealing with parametric values as they often require high precision (e.g. parameters in a normalized parametric space defining a NURBS spanning the full length of a ship). Failures in this regard can lead to inaccuracy in the model, and in the extreme, a breakdown in the coherence of the model leading to problems performing analysis (e.g. introduction of misalignments and gaps).

Finally, the DIME format supports a wealth of metadata that would also have to be populated. In all likelihood the exporting application, if it is indeed a vessel modeller, will contain most of this metadata, and it may potentially be a simple task to pass it on. It can, however, lead to difficulties similar to those encountered reconciling different parametric geometry definitions. If one system uses different conventions, then translation logic is required to manage the conversion. For instance, if conventions for assigning panel functions differ, a mapping has to be devised between the values in one system and the other.

5 Critical Analysis of the Existing Databases

This Section contains a critical analysis of the DIME and Trident Modeller databases with respect to the broader requirements of SPM storage for LCM. As each was deemed to be incapable of fulfilling all requirements in its current form, emphasis will be given to the shortcomings in an effort to recognize necessary characteristics to be included in the proposed data format.

It is worth noting that the specific storage mechanism for both Trident Modeller and DIME, i.e. XML, will not be discussed in this Section. This is not to say that the use of XML does not contribute advantages or disadvantages to either file format, but such discussion is simply deferred to the next Section.

5.1 Trident Modeller

5.1.1 Advantages

The Trident Modeller data format, RMGScript, provides more than simply a data serialization specification. Instead, it is an interpreted domain-specific scripting language for defining geometry. As such, it provides a concise, parametric definition for geometry that can be built up in a relational manner. The language supports inline mathematical operators and numerous built-in functions (a.k.a. anonymous objects) to ease building up complex object definitions from simpler ones. The RMGScript native schema can capture complex models with a small number of entities and parameter values, keeping model size down in comparison to storing resultant canonical geometry.

As well, because RMGScript defines geometry parametrically, the actual geometry must be built up by the interpreting application upon loading the script file. This fact, although leading to somewhat longer model load times, can be advantageous in that the resulting geometry is generated "clean" and stored in the optimal (native) manner for the kernel used by the application. This rules out a category of problems typical in scenarios when geometry is being exported from one system and read into another – tolerance mismatch (when different treatment of tolerances between two systems leads to difficulties recreating identical geometry and topology in each system).

5.1.2 Disadvantages

The fact that RMGScript is a full-fledged scripting language places a high burden on any application that has to work with its files. Unless the application is merely transforming or transferring the data without interpreting it and creating geometry, it must provide a full interpreter implementation backed by a suite of geometric functions that can be leveraged by the interpreter to create geometry. In most cases, the geometric functions would best be provided by integrating a formal geometry kernel (for instance, Trident Modeller uses SMLib for this). Although an alternative data format that stored geometry canonically would still require a reader (and be well-served by a geometry kernel), the task of reading a standard, canonical file format is less daunting than implementing an interpreter for a custom DSL with its own language grammar

and syntax. In addition, the fact that the script has to be interpreted and all geometry fully created "from scratch," leads to file load times that are significantly longer than situations in which resultant geometry is simply being read in and "re-initialized" in memory.

There are also a few shortcomings of the native RMGScript schema for storing SPM data that are owing to the fact that it was purpose-built to capture models for global FEA. It was not designed with storage for the various metadata and functional data required by some types of analyses (e.g. rules checking). For instance, while generic plates can be grouped together in "deck" or "bulkhead" collections in an RMGScript model, no explicit data identifying specific plate function is stored. No vessel principal particulars are stored, nor is anything pertaining to shipyard, classification status, regulatory particulars, etc. In addition, there are no entities to describe stiffener connection details, to identify spaces, nor to model equipment. Recall, too, that the native schema does not include entities for general non-planar geometry, and while the extended schema does, its entities suffer the limitations discussed at length in previous Sections.

5.2 DIME

5.2.1 Advantages

DIME was purpose-built to provide data translation from commercial ship design packages to LR design assessment tools. As such, one can recognize its schema is naturally well suited to data translation. This is reflected in the fact that its geometry definition has a strong canonical flavour (i.e. Contour3D is ubiquitous). Canonical definitions are well suited to passing data around, particularly where it is manipulated (i.e. written and read) behind-the-scenes. In addition, if a standard format is supported by all participating applications, oftentimes off-the-shelf components can be used to read/write geometry without custom processing. This results (in theory) in near identical reproductions of the model between systems with very little work.

In further support of its use for data translation, the DIME schema promotes the use of assets and instances. This is intended to further the cause by decreasing file sizes – a noble goal if the files are to be passed around frequently (particularly over the wire). As vessel models, particularly commercial vessel models, often contain many repeated structures, this scheme could well shave significant size off the model file without loss of detail in the model itself.

Finally, as DIME's provenance is in support of design assessment work, it has a much richer schema for storing all sorts of vessel data above and beyond simply geometry. This includes vessel metadata (ship particulars, regulatory information, etc.), plating function (e.g. identifying plates based on placement or function), compartment/space definition, connection details, etc. As such, one could argue that it has more coverage of the data required by an SPM than RMGScript presently does (it is, however, lacking entities to model equipment).

5.2.2 Disadvantages

If one considers the scripting language features of RMGScript to be an advantage, then by necessity their absence in the DIME schema would be considered a disadvantage. Often times, such functionality is useful and can lead to a concise definition of an entity. This is, however,

most useful if the file is to be read or manipulated by a user (e.g. making changes to the model), so it may be of limited concern for a translation-oriented data format (except perhaps that it may help keep file sizes down).

As stated above, the DIME data format defines geometry in a canonical manner in numerous locations. This has been discussed at length in previous Sections, but is reiterated here as a possible disadvantage, in particular if normalized parameter space values have to be stored. Recall from Section 4.2 the necessity to maintain high precision values in such cases can lead to potential problems.

Although the DIME schema does make use of object relationships in some situations, defining entity values relationally is not as widespread as within RMGScript. For example, while it is possible to trim a plate relationally, it is not possible to position the plate relationally (e.g. by stating the plate should be offset a given distance and direction from another plate). Again, while this is most useful when modifying a model, it can also be a means of retaining model coherence as it typically replaces a canonical value specification with a parametric one. Furthermore, object relational data can be leveraged for advanced operations such as model defeaturing.

While the DIME data format allows for non-planar surfaces to be built up from a definition of the bounding contour, this is limited to ruled surfaces, or at most bilinearly interpolated surfaces (not counting the custom HullForm entity). As such, the data format is lacking support for general non-planar surfaces. This has been partially circumvented using an approach analogous to the RMGScript extended schema, but the extension has yet to be formally included in the schema.

Finally, with regards to the use of assets and instances, there are two potential drawbacks. The first is that the currently supported location transformation is limited to a vector translation. It would be very useful to also support at least a standard rotation matrix to allow objects to be rotated into place. In addition, a means to scale the geometry should be employed. A naïve approach would be to allow a simple scale factor to be applied in each of the principal axis directions, but a better approach would be to allow the asset to be parameterized such that its actual dimensions could be set for each instance (keeping the topological “shape” constant). The second drawback to using assets and instances occurs only if they are not used properly. If a model is created in such a way that there is a single instance for each asset, then the goal of decreasing model size is not only not met, but indeed the file will be larger than it would have to be otherwise (to contain extra space for the boilerplate instance data).

6 Plan For The Way Ahead

Based on the critical analysis of the existing DIME and Trident Modeller model file formats, and based on the requirement to ultimately support the wide range of analysis types outlined earlier in the report, the following proposal is being made toward development of an initial version of a common data format that merges the best features of both existing formats. While the new format will initially be drafted to include storage for the majority of the required data, it is recognized that it will not exhaustively cover all data for all analyses. As such, the extensibility mechanism discussed in the final Section will ensure that it can be augmented as necessary with minimal impact on existing analysis applications that make use of it.

The remaining sub-Sections herein will present characteristics of the proposed database format. It is the intention that these characteristics are sufficiently detailed and complete that they can be implemented in a straightforward manner with reference back to both the DIME and RMGScript schemas.

6.1 Storage Mechanism

The new data format is proposed to be based, like both DIME and Trident Modeller, on XML technologies. As this proposal is setting forth a recommendation for merely version 1.0 of the data format, it is anticipated many additions and potential changes will be made in short order (e.g. to support more specialized analysis data). It is felt that a highly transparent, malleable file format best supports the currently recognized requirements while at the same time being agile enough to meet the needs of future requirements. We believe XML provides this, and leveraging the wealth of supporting technologies (e.g. XPath, XSLT), libraries (e.g. parsing routines, schema validation) and applications (GUI schema builders, schema-to-code converters, etc.) should significantly decrease time to implement solutions based on the proposal.

There are, however, potential drawbacks to basing the file format on XML. Firstly, XML is arguably verbose and inefficient at storing large amounts of repetitive, structured data. This can potentially lead to large file sizes and slow load times. Secondly, XML files are merely text files and not themselves capable of handling “enterprise” issues such as security, history tracking, corruption prevention nor concurrent use. However, there are readily available strategies to mitigate these drawbacks, which if needed, can be deployed in the short term. With respect to file sizes, this can be dealt with by collapsing whitespace (usually an option in the XML writer), or in the extreme, compressing the files when being saved or loaded by an application or converting to binary XML (likely only necessary if going to be transmitted over a network). With respect to the enterprise issues, these can be solved by the systems storing and accessing the files. Standard network security, routine backup practices and run-of-the-mill file version control systems (e.g. Mercurial) can be employed to provide counters to most of the issues.

Taking into consideration the short-term mitigating strategies, the potential drawbacks to choosing XML are not felt to outweigh the inherent benefits. This is particularly true for a file format that is in its infancy and will be undergoing much more design and modification. In the long term, as the format stabilizes and matures, it becomes more cost-effective to consider

optimizing strategies to alleviate any pressures that may truly be felt (if indeed there are any). At that point, technologies such as XML databases may be considered.

The initial implementation of the proposed database should be made in terms of a formal XML schema definition (XSD). Although XML documents can be created "on the fly," and in so doing allow for experimentation and fast evolution of a design, a schema file provides a number of benefits. Firstly, the schema forces all attributes and entities to be formalized in terms of their value constraints, whether or not they are required, parent-child relationships and repetition. Without a schema, these characteristics of a design must be carefully captured in design documentation, or risk being confused or forgotten. Secondly, the schema allows documents to be validated, which can save development time by automatically catching errors that go beyond trivial XML syntax – indeed the semantics of the document is checked. Finally, schema documents can be used by a variety of applications to help with design and implementation. The wealth of GUI applications that allow visualizing and modifying schema documents make them convenient for conveying design options within a team. Tools that allow code to be generated from a schema can also give developers a jump on implementing applications to manipulate documents based on the schema. For these reasons as well as others, it is strongly recommended the implementation of this proposal center on a formal XML schema.

6.2 Ship Metadata

Ship metadata is required for several of the analysis types under consideration, not the least of which are the LR design assessment applications ShipRight and RulesCalc. As the DIME database presently fully supplies these applications with required metadata, it is recommended that the new database clone the DIME schema in this regard. It is felt that all analysis packages should be well-served by these data, with the possible exception of hydrodynamic analysis. Hydrodynamic analysis requires many more particulars, but it is expected that most of these can be calculated from the principal particulars and/or the geometric model. What cannot should either be queried in the hydrodynamic application, or if considered a static characteristic of the model, included in the metadata entities via extension.

6.3 Ship Geometry

Discussion of what ship geometry to include in the model as well as how it is to be represented is discussed in the below sub-Sections.

6.3.1 Plating

Plating should be defined in a manner such that the geometric shape should be independent of the structural component type and/or function. The structural component type and/or function should be captured alongside the plate shape definition. This approach allows a relatively small number of generic plating types to cover a wealth of structural components (i.e. rather than a single entity for each structural component, even though some entities share the same shape). This strategy is effectively employed in both RMGScript and DIME, though RMGScript is perhaps more rigorous as it does not explicitly divide plating into transverse and longitudinal groups. For design

assessment analysis, however, it is important to distinguish transverse vs. longitudinal structure, so this distinction must be kept along with type and function metadata.

Plating must support various geometric shapes. Planar plates will be the most common, covering structural components such as decks, transverse and longitudinal bulkheads, floors, stringers, deep girders, inclined hopper plates, web frames, etc. Plating represented by analytical surfaces is also common. The analytical surface types should include cylinders, cones, hemispherical/torispherical/ellipsoidal domes, ruled surfaces, surfaces of revolution and bilinear surfaces. Structure typically represented by analytical surfaces includes the turn of bilge, shafts, domes, etc. Finally, to capture all other shapes, a general non-planar surface type should be supported. A NURBS surface is most suitable for this purpose. Structure such as hull forms, propellers, superstructure, camber decks, etc. would be captured with this surface type.

All plating types should support being trimmed in arbitrary fashion. This perpetuates the approach taken in both DIME and RMGScript whereby object trims could be built up by either reference to other objects or to geometric shapes (anonymous objects in RMGScript or Contour3D in DIME). This approach affords the most flexibility to apply generic plating shapes in situations which may contain complicated trim arrangements (i.e. it does not require a specific plating type for each trim arrangement).

All plating types should support subdivision for purposes of defining regions of uniform thickness or material property (or for other purposes such as defining paint). This corresponds to the notion of T/MZones in RMGScript and Plate3Ds in the DIME schema. Overall plates should support sub-division using the same flexible trimming mechanism that was employed to create them in the first place (i.e. no loss of flexibility and easier to reuse the trim entities already defined).

6.3.2 Stiffeners

All manner of stiffeners need to be supported, including those with planar and nonplanar webs, and planar and nonplanar flanges. Deck and bulkhead stiffeners typically exhibit both planar webs and flanges, but in some cases, flanges can be nonplanar (e.g. stiffeners with radiused end flare). Hull frames (transverse) typically also have a planar web with a nonplanar flange that follows the contour of the hullform. Hull longitudinals, the most complex, typically consist of fully nonplanar webs and flanges. The webs can be orthonormal to the hullform along their length, but they may also be constrained to particular angles at various locations (e.g. to align with other structure). The flanges are then orthonormal to the webs.

Most stiffeners can be built up from a trace line (where the web intersects the geometry to which the stiffener is attached), a profile, and optional orientation constraints. Both RMGScript and DIME take such an approach, but RMGScript mostly defines the trace line in terms of object intersections whereas DIME does so in terms of a Contour3D (RMGScript uses a NURBS for the definition of IGESTraceStiffener). Both mechanisms should be supported.

Many stiffeners are best represented as a collection of segments. This can be useful for varying properties such as scantlings, orientation, heading, material, etc. in the definition of an overall “logical” stiffener. Neither RMGScript nor DIME fully realizes this capability. While both can manually aggregate disparate segments into a logical overall definition, no shape continuity is

maintained between the segments (i.e. they are each standalone). DIME, however, does provide the SegmentCollection child of Stiffener3D which allows varying the orientation, but none of the other parameters.

While most stiffeners (or stiffener segments) simply extrude a single profile for their entire length, some do not. Stiffeners may be flared or tapered over part of their length. The capability, therefore, to support linearly interpolating two profiles (of the same type but different dimensions) defined at either end of a segment is required (this is supported in RMGScript with the *s1* and *s2* attributes of PlanarWebStiffener). Also, the ability to introduce a knuckle into the trace line defining the stiffener and a corresponding radius to smooth out the knuckle on the flange side is required (again, supported by RMGScript with the *r* attribute of PlanarWebStiffener).

Pillars represent a special type of stiffener, sufficiently different to warrant their own discussion. Unlike stiffeners, pillars are attached only at their ends. They can, therefore, be represented as linear extrusions of a profile between their two end attachment points. What cannot be determined automatically, though, is the orientation to use for the profile, if the pillar section is non-axisymmetric. As such, the orientation for the web (and possibly flange) must be specified as separate parameters (*d1* and *d2* in the RMGScript Pillar entity).

6.3.3 Libraries

Based on the Libraries entity within the DIME schema as well as the Material and Section entities in RMGScript, a library approach should be used to define parametric assets that can be referred to when creating geometry. This should include stiffener profiles, materials, end connections, brackets and penetration shapes. The specific parameters to include should correspond to the union of the properties currently defined in both DIME and RMGScript. In the case of brackets, end connections and penetration shapes, a collection of pre-defined, common parametric shapes should be provided (as is currently the case in both schemas with regards to penetrations). The collection can initially contain the union of the shapes defined in both schemas.

6.3.4 Penetrations

Penetrations should be included in roughly the same manner that they exist in both RMGScript and DIME presently. Namely, supporting the ability to parametrically define the penetration shape (taken from the library discussed above) in a flat space and then apply it to a given plate. Face plates (coamings) should also be supported by specifying one or two stiffener profiles to apply around the penetration edge (RMGScript's handling of coamings is slightly more flexible than DIME's in that DIME can only presently support a single profile). RMGScript's approach to orienting the penetration shape before cutting out the hole should also be preferred as it allows for a custom rotation to be applied in addition to defining a center by vector translation.

6.3.5 End Connections

The DIME schema is the only one that currently contains a definition for end connections. Therefore, its definition should be used as a starting point for the new schema. The end connection types presently supported by the DIME schema, however, are quite limited and will

undoubtedly need to be expanded. As such, this area of the schema will need to easily accommodate extension. It is recommended that each end connection parametric template be wholly contained in its own single (or parent) entity to help support extension.

6.3.6 Brackets

As for end connections, bracket definitions are currently only supported by DIME, so DIME's approach should be copied in this regard. A ready means to extend the library with new bracket types should be included.

6.3.7 Equipment and Other Non-structural Objects

The inclusion of equipment and other non-structural objects (e.g. furnishings, provisions, stores, etc.) is required by various analyses. In all cases, however, an approximate representation of the objects is sufficient. This can be in the form of an idealized volumetric representation. For instance, an engine can be represented by a rectangular volume, a torpedo by a cylindrical volume, etc. It is important, however, to capture as accurately as possible the actual mass and center of gravity of the object. Storage for accurate profiles of other properties should also be provided (e.g. to augment the equipment with an accurate profile of ferrous metal for magnetic analysis). Furthermore, a means to store a description of the connection of the equipment to the rest of the model will be required (e.g. are there specialized damping mounts, etc.).

6.3.8 Spaces

Spaces will need to be defined in a manner similar to that presently supported by DIME. A locating point should be chosen which will be considered to be within the given space. A set of bounding entities will then have to be chosen such that they form a closed volume containing the locating point. The bounding entities should, in most cases, be selected from existing plating, but in some cases it may be necessary (or convenient) to include standalone geometric surface shapes (i.e. anonymous objects). Furthermore, in the case that existing entities are chosen as bounding surfaces, they may have to be limited to sub-regions of their entire domain. The same flexible mechanism that is used to trim the plates in the first place should be supported for this purpose.

Spaces, once defined, should support associating arbitrary metadata as per the requirements of various analyses. This should include name, function, type (e.g. tank, cargo, air), etc.

6.4 Extensibility

The specific mechanism to achieve extensibility of the database schema will be discussed at length in the next Section. The functional goals of the extensibility mechanism, however, are given here.

Firstly, the extensibility mechanism must allow augmenting the schema with new types and/or information corresponding to new structures, properties, metadata, etc. This will include adding new root entities, new child entities, new entity attributes, and new attribute values. As it will be impossible to foresee all ways in which the schema may need to be extended over its lifetime, it

should be preferred to allow extension at virtually any location. Cases where it may seem necessary to limit extension should be considered very carefully lest they introduce a legacy of inconvenience.

Secondly, the extensibility mechanism must carefully consider compatibility. Ref. [8] states that there are two fundamental types of compatibility:

(i) backwards compatibility – newer versions of the schema should be readable by older applications

(ii) forwards compatibility – older versions of the schema should be readable by newer applications

The extensibility mechanism should strive to maintain both types of compatibility. Doing so will afford the most flexibility for the schema and its application ecosystem to evolve at different rates.

6.5 Miscellaneous

DIME's assets and instances are discussed at length in previous Sections (e.g. 3.2.1). It is suggested that an enhanced version of the concept be introduced in the new schema. Whereas DIME's current implementation only allows transforming an asset during instancing, it is suggested that the model be enhanced to support parameterization of key asset dimensions. The exact parameterization can be left up to the asset designer, but should allow dimensional variation of a given asset topology. This should make assets much more tolerant of small perturbations to their geometry, and hence more able to be re-used. In addition, their locating transformation should support a rotation in addition to a vector translation.

In discussing the various geometric shapes to support plate definition (Section 6.3.1), specific strategies for storing the geometry were not covered. It is suggested that a strategy similar to RMGScript be employed for this. Namely, planar and analytical surface shapes should, in most cases, be defined parametrically within the new schema, whereas complex general non-planar surfaces should be stored canonically (planar or analytical surfaces could be selectively stored canonically if it is convenient in a particular situation). Entities stored canonically, however, should use an industry-standard format such as IGES or STEP. No attempt should be made to devise a set of custom entities to try and capture a canonical definition, as the task will be wrought with pitfalls (it often takes many years for the industry-standard formats to mature). The canonical definitions can be stored either inline (encoded if they are binary) or external to the model file (the latter option is most common, though requires keeping many accompanying files with the main model file).

7 Extensibility Mechanism

The data requirements for each of the engineering analyses specified in Section 2 vary greatly. While each requires some representation of the structure of a vessel (to varying levels of detail), individual analyses also typically require specific extra information. For instance, infrared signature analysis requires data describing ship ventilation and exhaust, radar signature requires special material properties (permittivity, conductivity, permeability), magnetic signature requires machinery data, etc.

Although the initial version of the data format laid out in Section 6 focuses on the most common data requirements, a means to cover the additional requirements is being proposed as a generic extensibility mechanism. The main enabler for the proposed extensibility mechanism is the fact that the data format is serialized in XML format. The World Wide Web Consortium (W3C) XML schema definition language, W3C XML Schema 1.0 [10], provides mechanisms for extensibility, but they must be employed with appropriate foresight to ensure they achieve their purpose with minimal friction and, above all, whilst maintaining forwards and backwards compatibility of the data format.

The extensibility mechanism proposed in this section was derived from [8] and [9]. These references suggest practical strategies for designing XML schemas that are both versionable and extensible within the constraints of the W3C XML schema language. While other XML schema languages such as Schematron (11), and RelaxNG [12] can offer superior extensibility mechanisms, they are generally not as widely known and adopted. As such they were not considered viable for this work as an emphasis was placed on a language that was commonly known and had extensive tool support. In addition, while the next version of the W3C XML Schema language (1.1) promises to provide features that make extensibility much easier [13], it has not been finalized and it may be some time before compliant tools are on the market.

Ref. [9] builds on [8] and proposes that the following guidelines be followed in order to achieve extensibility in XML schema definitions:

1) XML formats should be designed to be extensible.

2) Extensions must not use the namespace of the XML format.

3) All XML elements in the format should allow any extension attributes, and elements with complex content should allow for extension elements as children.

4) Formats that support extensibility must specify a processing model for dealing with extensions.

W3C XML Schema provides the following features that promote extensibility:

1) Wildcards `xs:any` and `xs:anyAttribute` that allow the occurrence of elements and attributes from specified namespaces. In the wildcard definitions, the attribute `namespace` is used to specify the namespace from which elements or attributes the wildcard matches

can come from. The attribute `processContents` is used to specify if and how the XML content matched by the wildcards should be validated.

2) Substitution groups and abstract elements. A substitution group contains elements that can appear interchangeably in an XML document. A restriction exists in that the members of a substitution group must be of the same type or they must belong to the same type hierarchy as the element being substituted. Listing 3 shows and example of use of a substitution group taken from [9].
3) Polymorphism via `xsi:type` and abstract types. The `xsi:type` attribute can be applied to an element in an XML document to change its type with the restriction that the new type is in the same hierarchy as the original type of the element and is generally used to name the type of a primitive value specified as `xs:anySimpleType` in a schema. Abstract types on the other hand are complex type definitions that have their `abstract` attribute set to true and which cannot be used in an instance document, but must be replaced by a derived type. Listing 4 shows an example of use of a type redefinition taken from [9].
4) `xs:redefine` used for redefinition, where a type effectively derives from itself. `xs:redefine` brings in declarations and definitions from another schema and makes them available in the target namespace. The included declarations and types must be from a schema with the same namespace or have no namespace.

```
---------------------------------------------------------------------------------------------------------------
example.xsd:

<xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.com"
        xmlns:ex="http://www.example.com"
        elementFormDefault="qualified">

        <xs:element name="book" type="xs:string" />

        <xs:element name="magazine" type="xs:string"
                substitutionGroup="ex:book" />

        <xs:element name="library">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="ex:book"
                                        maxOccurs="unbounded"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
</xs:schema>

example.xml:

<library xmlns="http://www.example.com">
        <magazine>MSDN Magazine</magazine>
        <book>Professional XML Databases</book>
</library>
---------------------------------------------------------------------------------------------------------------
```

Listing 3 – Application of substitution groups

```
example.xsd:

<xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.com"
        xmlns:ex="http://www.example.com"
        elementFormDefault="qualified">

        <xs:element name="book" type="xs:string" />

        <xs:complexType name="bookWithIsbnType">
                <xs:simpleContent>
                        <xs:extension base="xs:string">
                                <xs:attribute name="isbn" />
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>

        <xs:element name="library">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="ex:book"
                                        maxOccurs="unbounded"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>

</xs:schema>

example.xml:

<ex:library xmlns:ex=http://www.example.com
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <ex:book>Mort</ex:book>
        <ex:book xsi:type="ex:bookWithIsbnType"
                isbn="0-06-105764-9">Feet of Clay</ex:book>
</ex:library>
```

Listing 4 – Application of type polymorphism

Ref. [9] proposes that the following guidelines be followed to achieve versionability in XML schema definitions:

1) If a format is backward compatible with previous versions, the old namespace name must be used in conjunction with XML's extensibility model. Namespace names should not be the primary mechanism for versioning. Instead, other means such as a `version` attribute on the root element should be used.
2) A new namespace name must be used when backward compatibility is not permitted.
3) Formats should specify a `mustUnderstand` model for dealing with backward incompatible changes to the format that do not change the namespace name. Consumers of the format must be able to identify a change in version, usually done using a version number on the root element, and they must understand all elements from the target namespace of the format if they support the specified version number.

Listings 5 and 6, taken from [9], show the evolution of two versions of a schema illustrating the application of guidelines 1 and 2.

```
----------------------------------------------------------------------------------------------------------------
BOOKS-CORE.XSD

<xs:schema blockDefault="#all"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.com/books-core">
        <xs:attribute name="mustUnderstand" type="xs:boolean" />
</xs:schema>

BOOKS-V1.XSD

<xs:schema blockDefault="#all" elementFormDefault="qualified"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.com/books/v1"
        xmlns:b1="http://www.example.com/books/v1">

        <xs:element name="books">
                <xs:complexType>
                        <xs:sequence>
                        <xs:element name="book" type="b1:bookType"
                                maxOccurs="unbounded" />
                        </xs:sequence>
                        <xs:attribute name="version" type="xs:string" />
                </xs:complexType>
        </xs:element>

        <xs:complexType name="bookType">
                <xs:sequence>
                        <xs:element name="title" type="xs:string" />
                        <xs:element name="author" type="xs:string" />
                        <xs:any namespace="##other" minOccurs="0"
                                maxOccurs="unbounded"
                                processContents="lax" />
                </xs:sequence>
                <xs:attribute name="publisher" type="xs:string" />
        </xs:complexType>
</xs:schema>

BOOKS.XML

<books version="1.0" xmlns="http://www.example.com/books/v1">
        <book publisher="IDG books">
                <title>XML Bible</title>
                <author>Elliotte Rusty Harold</author>
        </book>
        <book publisher="Addison-Wesley">
                <title>The Mythical Man Month</title>
                <author>Frederick Brooks</author>
        </book>
        <book publisher="WROX">
                <title>Professional XSLT 2nd Edition</title>
                <author>Michael Kay</author>
                <price xmlns="http://www.example.com/book/extensions">
                        24.99
                </price>
        </book>
</books>
----------------------------------------------------------------------------------------------------------------
```

Listing 5 – First version of schema

```
-----------------------------------------------------------------------------------------------------------------------
BOOKS-V1.XSD

<xs:schema blockDefault="#all" elementFormDefault="qualified"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.com/books/v1"
        xmlns:b1="http://www.example.com/books/v1"
        xmlns:b2="http://www.example.com/books/v2">

        <xs:import namespace="http://www.example.com/books/v2"
                schemaLocation="books-v2.xsd" />

        <xs:element name="books">
                <xs:complexType>
                        <xs:sequence>
                        <xs:element name="book" type="b1:bookType"
                                maxOccurs="unbounded" />
                        </xs:sequence>
                        <xs:attribute name="version" type="xs:string" />
                </xs:complexType>
        </xs:element>

        <xs:complexType name="bookType">
                <xs:sequence>
                        <xs:element name="title" type="xs:string" />
                        <xs:element name="author" type="xs:string" />
                        <xs:element ref="b2:isbn" />
                        <xs:any namespace="##other" minOccurs="0"
                                maxOccurs="unbounded" processContent
                        s="lax" />
                </xs:sequence>
                <xs:attribute name="publisher" type="xs:string" />
        </xs:complexType>
</xs:schema>

BOOKS-V2.XSD

<xs:schema blockDefault="#all" elementFormDefault="qualified"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.com/books/v2"
        xmlns:core="
        xmlns:core="http://www.example.com/books-core">

        <xs:import namespace="http://www.example.com/books-core"
                schemaLocation="books-core.xsd" />

        <xs:element name="isbn">
                <xs:complexType>
                        <xs:simpleContent>
                                <xs:extension base="xs:string">
                                <xs:attribute ref="core:mustUnderstand"
                                        fixed="false"/>
                                </xs:extension>
                        </xs:simpleContent>
                </xs:complexType>
        </xs:element>
</xs:schema>

BOOKS.XML

<books version="2.0" xmlns="http://www.example.com/books/v1"
        xmlns:p="http://www.example.com/book/extensions"
```

```
        xmlns:v2="http://www.example.com/books/v2"
        xmlns:bc="http://www.example.com/books-core">
        <book publisher="HCI">
                <title>A Child Called It</title>
                <author>Dave Pelzer</author>
                <v2:isbn bc:mustUnderstand="false">1-55874-766-9</v2:isbn>
                <p:price>9.95</p:price>
        </book>
</books>
```

Listing 6 – Second version of schema

Listings 7 and 8, taken from [9], illustrate the use of extensibility points with sentries, which overcome the problem of non-deterministic content models caused by naïve use of wildcards. Listing 7 also overcomes two problems of the approach demonstrated in listings 6 and 7, in which new constructs are represented by new namespaces, namely, the difficulty for human readers in distinguishing core aspects of the format from extensions and the lack of forward compatibility.

```
BOOKS-CORE.XSD
<xs:schema blockDefault="#all"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.com/books-core">

        <xs:element name="delimiter">
                <xs:complexType />
                </xs:element>

        <xs:element name="end">
                <xs:complexType />
        </xs:element>

        <xs:attribute name="mustUnderstand" type="xs:boolean" />

</xs:schema>

BOOKS.XSD

<xs:schema blockDefault="#all" elementFormDefault="qualified"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.com/books"
        xmlns:b="http://www.example.com/books"
        xmlns:bc="http://www.example.com/books-core">

        <xs:import namespace="http://www.example.com/books-core"
                schemaLocation="books-core.xsd" />

        <xs:element name="books">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="book" type="b:bookType"
                                        maxOccurs="unbounded" />
                        </xs:sequence>
                        <xs:attribute name="version" type="xs:string" />
                </xs:complexType>
        </xs:element>
```

```
        <xs:complexType name="bookType">
                <xs:sequence>
                        <xs:element name="title" type="xs:string" />
                        <xs:element name="author" type="xs:string" />
                        <xs:element name="isbn" type="xs:string" minOccurs="0" />
                        <xs:sequence minOccurs="0" maxOccurs="1">
                                <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                        <xs:element ref="bc:delimiter" />
                                        <xs:any namespace="##targetNamespace ##local"
                                                minOccurs="0" maxOccurs="unbounded"/>
                                </xs:sequence>
                                <xs:element ref="bc:end" />
                        </xs:sequence>
                        <xs:group ref="b:extensionGroup" minOccurs="0" />
                </xs:sequence>
                <xs:attribute name="publisher" type="xs:string" />
        </xs:complexType>

        <xs:group name="extensionGroup">
                <xs:sequence>
                        <xs:element name="extensions">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:any namespace="##other" minOccurs="0"
                                                        maxOccurs="unbounded"
                                                        processContents="lax" />
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:group>
</xs:schema>

BOOKS.XML

<books version="1.0" xmlns="http://www.example.com/books">

        <book publisher="IDG books">
                <title>XML Bible</title>
                <author>Elliotte Rusty Harold</author>
        </book>

        <book publisher="Addison-Wesley">
                <title>The Mythical Man Month</title>
                <author>Frederick Brooks</author>
                <isbn>0-373-70708-8</isbn>
        </book>

        <book publisher="WROX">
                <title>Professional XSLT 2nd Edition</title>
                <author>Michael Kay</author>
                <extensions>
                        <price xmlns="http://www.example.com/book/extensions">24.99</price>
                </extensions>
        </book>

</books>
-----------------------------------------------------------------------------------------------------------------------------
```

Listing 7 – First version of schema using extensibility points with sentries

```
----------------------------------------------------------------------------------------------------------------
BOOKS.XSD

<xs:schema blockDefault="#all" elementFormDefault="qualified"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.com/books"
        xmlns:b="http://www.example.com/books"
        xmlns:bc="http://www.example.com/books-core">

        <xs:import namespace="http://www.example.com/books-core" schemaLocation="books-core.xsd"/>

        <xs:element name="books">
                <xs:complexType>
                        <xs:sequence>
                           <xs:element name="book" type="b:bookType" maxOccurs="unbounded" />
                        </xs:sequence>
                        <xs:attribute name="version" type="xs:string" />
                </xs:complexType>
        </xs:element>

        <xs:complexType name="bookType">
                <xs:sequence>
                        <xs:element name="title" type="xs:string" />
                        <xs:element name="author" type="xs:string" />
                        <xs:element name="isbn" type="xs:string" minOccurs="0" />
                        <xs:sequence minOccurs="0" maxOccurs="1">
                                <xs:element ref="bc:delimiter" />
                                <xs:element name="edition-number" type="xs:positiveInteger"
                                minOccurs="0" />
                                <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                        <xs:element ref="bc:delimiter" />
                                                <xs:any namespace="##targetNamespace ##local"
                                                        minOccurs="0"maxOccurs="unbounded"/>
                                </xs:sequence>
                                <xs:element ref="bc:end" />
                        </xs:sequence>
                        <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"
                        processContents="lax" />
                </xs:sequence>
                <xs:attribute name="publisher" type="xs:string" />
        </xs:complexType>

</xs:schema>

BOOKS.XML

<books version="2.0" xmlns="http://www.example.com/books"
        xmlns:p="http://www.example.com/book/extensions"
        xmlns:bc="http://www.example.com/book-core">

        <book publisher="HCI">
                <title>A Child Called It</title>
                <author>Dave Pelzer</author>
                <isbn>1-55874-766-9</isbn>
                <bc:delimiter />
                        <edition-number>1<edition-number>
                <bc:end />
                <extensions> <p:price>9.95</p:price> </extensions>
        </book>
</books>
----------------------------------------------------------------------------------------------------------------
```

Listing 8 – Second version of schema using extensibility points with sentries

It is proposed that the SPM XML schema be developed in accordance to the guidelines outlined above.

In addition, it is proposed that it use version numbers to differentiate between different versions of the schema by means of a version attribute on the root element of the format. The major and minor versioning scheme from traditional software development practices should be used to help users identify compatibility between versions in a straightforward manner.

It is proposed that the XML format be namespace qualified and the namespace name be the same between minor versions and that it be changed with major versions.

It is proposed that the schema be explicitly referenced in all instance documents via the attribute xsi:schemaLocation.

It is proposed that and that all of the versions of the schema be readily available to clients for the purpose of validation.

It is proposed that clients adhere to the schema and that they conform to the Must Ignore in combination with a Must Understand processing model, as per the schema definition.

It is proposed that breaking changes to the format be minimized, i.e. changes involving addition of new concepts and deprecation of existing concepts shall be favoured over changes to existing concepts and their removal.

In cases where breaking changes become unavoidable it is proposed that means for conversion be provided via standard technologies such as XSLT, translation components, or cloud services.

Following these guidelines and proposals will make it more likely that a versionable and extensible SMP XML schema be realized.

References

[1] Brennan, D. et. al. *Single High Fidelity Geometric Data Sets for LCM – Model Requirements*, Martec Technical Report No.: TR-06-18. Martec Limited, Halifax, Nova Scotia. August 2006.

[2] Lloyd's Register, *Interface Toolkit DIME 2010.1 User's Guide*. Lloyd's Register, London, U.K. July 2010

[3] Lloyd's Register, *RulesCalc 2010.0 User's Guide*. Lloyd's Register, London, U.K. June 7, 2010

[4] Lloyd's Register, *ShipRight 2010.1 User's Guide*. Lloyd's Register, London, U.K. March 9, 2010

[5] MacAdam, Tom. *RMGScript User's Guide* – Martec internal memo. Martec Limited, Halifax, Nova Scotia. December 2007

[6] Roberts, Paul. *Geometry Model – "Assets & Instance" Aims and Future Planning/Discussion*. LR MPD Engineering Software internal memo. Lloyd's Register, London, U.K. Dec 18, 2009

[7] Yu, Lei. *Application Constraints for Data Access Interfaces with ShipRight SDA*. LR Ship Design Systems internal report. Lloyd's Register, London, U.K. June 6, 2003.

[8] Obasanjo, D. *Designing Extensible, Versionalble XML Formats*. July 15, 2004. Online: http://msdn.microsoft.com/en-us/library/ms950793.aspx

[9] Orchard, D. *Extensibility, XML Vocabularies and XML Schema*. October 27, 2004. Online: http://www.xml.com/pub/a/2004/10/27/extend.html

[10] Fallside, D., Walmsley, P. *XML Schema Part 0: Primer Second Edition*. October 28, 2004. Online: http://www.w3.org/TR/xmlschema-0/

[11] Information Technology – Document Schema Definition Languages (DSDL) – Part 3: Rule-based Validation – Schematron. ISO/IEC 19757-3. June 1, 2006. Online: http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip

[12] Clark, J., Makoto, M. RELAX NG Specification. December 3, 2001. Online: http://www.relaxng.org/spec-20011203.html

[13] Orchard, D. *Guide to Versioning XML Languages Using New XML Schema 1.1. Features*. 20 July 2007. Online: http://www.w3.org/TR/xmlschema-guide2versioning/

This page intentionally left blank.

Annex A DND LCM Analysis Data Requirements Summary

A.1 Global Structural FEA

The data required for a global finite element analysis are:

- geometry (equivalenced to ensure proper connectivity) including section properties,
- material properties,
- mass data, and
- loads

Of these, it is the geometric, mass and material data that will be imported from a LCM database.

A.1.1 Geometric Data

For longitudinal strength analysis of a global finite element model in the preliminary design stage it is necessary to supply geometric and material data for the following structural components:

- All longitudinal elements that are continuous along the ship including decks, longitudinals, girders, bulkheads, hull plating
- Longitudinal elements that are not continuous should be modelled, paying special attention to the way the discontinuity is modelled.
- Girders (should be modelled with beam elements)
- Stiffeners (could be smeared into the plate or modelled with beam elements)
- Major transverse bulkheads have to be modelled
- Frames (should be modelled with beam elements)
- Pillars
- Floors

Since girders, stiffeners and frames are either incorporated into adjoining plate elements or are modelled with beam elements it follows that centreline locations and section properties are sufficient. Geometric data for all other components would consist of mid-thickness locations and plate thicknesses.

Using the geometric data from the concept and/or preliminary design software will be essential for a global model analysis tool. However, the program should also be able to generate a model of the hull shape from the lines of form.

A.1.2 Section Properties and Material Data

Plate thickness, stiffener scantlings and structural material properties are all required for the global finite element analysis.

The most basic structural material data includes Young's modulus and Poisson's Ratio. Depending on the type of analysis, additional material data may be required. This could include: density, yield stress, non-linear structural properties, fatigue properties.

A.1.3 Mass Data

An accurate representation of the weight of the ship is required for global structural analysis. The most efficient approach to obtain the correct weight and distribution is to use a weight curve. The mass of items such as engines and equipment that weigh over 10 tonnes should be represented separately and located at the correct position in the global model.

A.2 Detailed Structural FEA

Data requirements for a detailed FE analysis are similar to those for a global FE analysis.

A.2.1 Detailed Geometric Data

- equivalenced to ensure proper connectivity

The level of detail required of a structural FE model will depend on the specified analysis. For example, a fatigue and fracture analysis will require a more detailed mesh than a structural stress analysis that determines the structural integrity of a deck. Therefore the fidelity of a model must be sufficiently flexible to accommodate the diverse requirements of various types of structural analyses.

A.2.2 Section Properties and Material Data

- as per global analysis

A.2.3 Mass Data

- as per global analysis

A.2.4 Boundary Conditions

- not from LCM data,
- from a top-down (global FEA) analysis as well as any internal BCs.

A.3 Hydrodynamics

The following information is required for the hydrodynamic analysis of a displacement type ship. For other types of ships, such as hydrofoil vessels or air-supported crafts, different data will be required.

A.3.1 Ship Geometry and Weight Distribution

Basic ship data: First, and foremost, the geometry of ship hull surface, without appendages, rudder and so on is modelled. This surface description could take the form of a NURBS expression, or a set of meshes in standard boundary element format, or an offset table. If the hull geometry is defined by an offset table then three types of projection lines (body-plan lines, water-plan lines and buttock-plan lines), must be provided together with necessary additional lines such as the chine line and deck edge line. The points listed in the table should have the characteristic property like FAIR or KNUCLE. Skin roughness is needed for the fraction resistance estimation. In addition, the following particulars are needed

- Length overall
- Length between perpendiculars (L)
- Beam (B)
- Midships draft (T)
- Trim
- Volume displacement (∇)
- Displacement
- Longitudinal location of the center of gravity in station
- Center of gravity above keel
- Longitudinal metacentre height
- Transverse metacentre height

- Wetted hull surface area
- Water plane area (Awp)
- Radius of gyration in roll
- Radius of gyration in pitch
- Radius of gyration in yaw
- Cross mass inertia moment between x- and y-axis
- Cross mass inertia moment between y- and z-axis
- Cross mass inertia moment between z- and x-axis
- Block coefficient (Cb=∇/(L*B*T))
- Midship coefficient (Cm=Immersed area of midship section)/(B*T))
- Water plane coefficient (Cwp=Awp/(L*B))
- Prismatic coefficient (Cb/Cm)
- Vertical prismatic coefficient (Cb/Cwp)

A.3.2 Hull Section Parameters

Sectional data, based on station sections or frame sections, are also needed. At each section the required information includes a body-plan line as well as the following section parameters.

- Section beam (b)
- Section draft (t)
- Section area (S)
- Section mass
- Center of section gravity above keel
- Section radius of gyration in roll
- Section radius of gyration in pitch
- Section radius of gyration in yaw
- Section coefficient (Cs= S/(b*t))

- Section length

- Area of wetted hull surface

A.3.3 Appendages

- Bilge Keel: Geometric bilge information includes: root line coordinates defining the intersection between the bilge keel and the ship hull; tip line coordinates; the section shape; bilge keel height; bilge keel length; and the bilge keel depression angle.

- Fins: Fin section shape; fin span; fin chord; fin thickness; fin root submergence; bilge radius at fin location; fin depression angle; distance of fin centre of pressure from fin root; longitudinal location; lateral offset from the ship centerline; vertical position from keel plane; mechanical limitation of attack angle.

- Skeg information: Skegs can be defined by a set of the panels if it is not included in the hull surface geometry part.

- Rudder system: Rudder geometric information includes section shape; rudder span; rudder thickness; rudder chord information including rudder flap chord and rudder mean chord; ratio between flap angle and mechanical angle; numerical factor related to flaps if the rudder has flaps; bilge radius at rudder location; rudder depression angle; distance of rudder centre of pressure from rudder root; longitudinal location; lateral offset from the ship centerline; vertical position from keel plane.

- Propeller system: The geometry of the propeller, including the blades and hub should be defined by a NURBS expression of a set of panels. The geometry of the nozzles part should also be defined in a similar way if the system is a ducted type. Thrust, RPM, wake factor, location of the centre of the propeller disk and turning direction should be provided. Following parameters are needed as well: number of blades, diameter, pitch, blade thickness ratio, pitch angle, disk area, developed area of blades outside hub, developed area ratio, projected area of blades outside hub, projected area ratio, blade width ratio, mean width ratio.

The geometry (shape) of the ship hull and appendages should be available from the database and should be sufficiently detailed to provide shapes to a scale of roughly 1 to 2 meters.

A.3.4 Environmental Parameters

Required environmental parameters for a hydrodynamic analysis include:

- Ship speed, and engine working data,

- Water depth or bathymetry data,

- Wind speed and wind direction,

- Current speed and direction,

- Wave statistic parameters such as the type of the random sea, the significant wave height and peak or averaged wave period, and the principal wave direction of the operation area for a irregular sea analysis,

- Measured wave spectrum of the operation area for an irregular sea analysis, or the wave condition: wave frequency, wave direction and wave height for a regular wave case analysis.

Wave statistics data should be available from AES hindcast database, while the measured wave spectrum could be obtained from sea trials. The bathymetry data is available from some database such as ETOPO2.

A.4 Radar Signature

A.4.1 Geometric Properties

- 3-D configurations of the hull, superstructure, mast, and appendages such as equipment, as sonar dome, propeller shafts, brackets, rudders, etc

A.4.2 Material Properties

- Material type, thickness, permittivity, conductivity, permeability

A.4.3 Others

- Source power, frequency, location, threshold field strength

A.5 Infrared Signature

A.5.1 Geometric Properties

- 3-D configuration of ship geometry, including location and size of weapons, sensors, and other equipment;

- Propulsion and auxiliary engine exhaust properties for range of power settings;

- Ship insulation plan drawings;

- Internal machinery layout and exhaust routing arrangement drawings;

- Ship ventilation plan drawings (machinery room ventilation is most important);

A.5.2 Non-Geometric Properties

- Technical data (specifically thermal properties and geometry) on weapons, sensors, and other equipment;
- Ship surface properties, namely paint selections (spectral emissive data);
- Data on other miscellaneous sources of thermal IR, for example: galley stove exhausts, effluent discharges, heated widows;

A.5.3 Supplementary Data

- range of environmental conditions under which the ship is to operate.

A.6 Electrical Potential Signature and Cathodic Protection

Model input parameters for both corrosion analysis and underwater electric potential include:

A.6.1 Geometric Data

A detailed description of the geometry of the wetted hull and any submerged appendages such as shafts, propellers and rudders. As the modeling technique deals with the surfaces of the structure, the equivalent surface area of any materials exposed directly to the seawater needs to be accurately represented by the model (typically this includes the propeller)

A.6.2 Material Data

Paint quality and paint damage of the wetted surfaces. Typically the degree of damage is represented as a percentage of exposed metal surface at a given location.

Potentiostatic polarization curves for any wetted surface material exposed to seawater (either by design or through paint damage), including sacrificial anodes. The polarization curve represents the relationship between current density and electric potential (relative to a standard electrode).

A.6.3 Cathodic Protection System

A description of the cathodic protection system, including: the location and number of impressed current anodes; location and number of reference electrodes; location, number and size of any sacrificial anodes; as well a as general description of the operation of the impressed current control (control algorithm, maximum anode currents, number of power sources, typical set points for reference electrodes, etc.).

A.6.4 Supplementary Data

Other factors considered in the models relate to the operating environment and include such things as:

- Conductivity of the surrounding seawater
- Littoral geometry
- Ship speed and propeller rpm
- Nearby marine structures (i.e. ships, piers, pipelines, etc.)

A.7 Magnetic Signature

A.7.1 Geometric Data

The shape, size and (magnetic) material description of all major ship components made of ferrous materials, including structural and non-structural components are required.

Geometric descriptions of all structural steel should be available from the central database. This should provide a description of the geometry down to a resolution of 1 to 2 meters. Partial descriptions of some non-structural steel components, such as engines, generators, shafts, etc. should be available from the database. However, it is unlikely that such a database would be able to indicate the amount and distribution of ferrous material within such components. It is expected that this type of information would have to be extracted from an independent source. Likewise, magnetic material properties and descriptions of degaussing systems would probably have to come from independent sources.

A.7.2 Material Data

- induced and permanent magnetic properties of all ferrous materials

A.7.3 Magnetic Field Sources

- descriptions of major fixed magnetic fields (the earth's magnetic field),
- other major electrical circuitry that will produce large magnetic fields outside the ship (primarily the degaussing circuits).

A.8 Low Frequency Acoustic Signature

A.8.1 Geometric Data

The basic data requirement for boundary element based tools, such as AVAST, is a geometric description of the wet surface of the ship structure (including all appendages). In general, this geometric definition will be in the form of three or four-node facets or panels, however some acoustic modeling tools (including AVAST) support higher order isoparametric panel formulations. The fidelity of the panel mesh is a function of the frequency at which the acoustic signature is to be computed (i.e.: the higher the frequency the finer the associated boundary element size). As a result, it is important that any tool(s) used to extract the structural geometry and generate low frequency acoustic models provide a capability for defining the appropriate mesh size based on frequency. It is anticipated that the number of boundary elements required for the low frequency acoustic analysis (both radiated noise and target strength) of Canadian Forces vessels will exceed 15,000.

For cases where the elastic response of the ship structure is of interest, a global finite element model of the ship structure will also be required. This global finite element model is used to capture the global natural frequencies of the structure. In practice, researchers at both DRDC-Atlantic and Martec have used finite element models having a level of refinement similar to that found in Maestro [9] models, providing an upper frequency bound for ship structures (similar in size to the CPF) of approximately 30 Hz.

A.8.2 Supplementary Data

Additional information, in terms of hull coating impedances and the location of air-backed / water backed panels, will also be required for target strength prediction. It is anticipated that most, if not all, of this information will be stored as part of a SPM database. The only outstanding information, related to the material properties of the fluid domain, must be supplied by the user at run time.

A.9 High Frequency Acoustic Data

A.9.1 Geometric Data

One of the most demanding aspects of EFEA modeling is related to the development of a suitable mesh describing the geometry of the structure. Fortunately the EFEA approach has a significant advantage over other high frequency analysis methods in that it is compatible with finite element modeling, i.e., a finite element mesh of a ship structure could, in theory, be used as input to an EFEA analysis. In practice, the level of refinement used to discretize a structural model (i.e., number of nodes and elements) depends on what is of interest to the analyst, and as a result, the level of refinement could vary from location to location within the EFEA model. For example, it may be very important to model the spatial variation of energy within the engine room with a

high degree of accuracy, but less important in the galley. As a result, the model of the engine room would be much more detailed

Although defining the overall geometry of the structure is an important issue, our experience has shown that what limits the size (i.e.: degree of refinement) of EFEA models is the level of effort needed to define the junctions (or connections) between structural components. SNAP uses a junction to define how energy is transferred between structural components at a structural discontinuity. At present, L- and T-connections are supported (unfortunately crosses are not). In the current version of the SNAP code, the definition of junction data must be prepared manually, and as a result, is extremely time consuming and error-prone. What is needed to make EFEA analysis practical for ship structures is a modeling tool that automatically computes the junction properties. Without such a tool, the complexity of models that may be analyzed using EFEA software is quite limited, perhaps to models containing fewer than 100 junctions.

A.9.2 Input Power Data

Another important issue related to EFEA modeling is the manner in which input power is defined. In the current version of the SNAP code, users define the input power in terms of forces and moments. The code then converts these loads into input power using the structural input impedance. The formula used in SNAP for computing the input power associated with a harmonic force of amplitude Fo is provided below in Equation (A.1):

$$P_{in} = Re\{F_o e^{jwt}\} Re\{v_o e^{jwt}\} \tag{A.2}$$

Where vo represents the velocity generated by the application of the force Fo. Working with time-averaged values, Equation (A.3) can be shown to be equivalent to Equations (A.4) and (A.5):

$$P_{in} = ½|F_o|^2 Re\{1/Z\} \tag{A.6}$$

$$P_{in} = ½|v_o|^2 Re\{Z\} \tag{A.7}$$

Where Z represents the input impedance. Expressions for the impedance are available in the literature, some of which have been coded in the current version of the SNAP code.

Given the fact that input power could be defined using either forces or velocities, it may be possible to convert source vibration data into applied power. Further investigation will be required in order to access the viability of doing so. Using vibration data measured by a manufacturer may be difficult to apply directly because the input impedance used in the tests may not be known.

In summary, the SNAP software will allow for point load inputs applied to a select number of structural foundations (such as simple plates or beams). Using the structural information for that foundation, the input forces are converted to input power, which is the required input for the EFEA software. While the number of allowable input structures is presently limited, they are

likely sufficient for a large number of naval applications and more complex types would be developed as required.

This page intentionally left blank.

Annex B RMGScript Language Entities

B.1 RMGScript XML Entities

B.1.1 Collection Entities

Submarine { name, visible, fidelity, units, tolerance }

Collection { name, visible, fidelity }

B.1.2 Reference Value Entities

RefConic { name, c }

RefPlane { name, pl }

RefPoint { name, p }

RefString { name, s }

RefValue { name, v }

RefVector { name, v }

B.1.3 Geometric Entities

ConicPlate { name, visible, fidelity, color, pl, r1, r2, h, amin, amax, t, m }

EllipsoidalPlate { name, visible, fidelity, color, pl, rn, rj, amin, amax, t, m }

HemisphericalPlate { name, visible, fidelity, color, pl, r, amin, amax, t, m }

TorisphericalPlate { name, visible, fidelity, color, pl, ra, rb, rj, a, h, amin, amax, t, m }

BilinearPlate { name, visible, fidelity, color, p00, p10, p11, p01, t, m }

Pillar { name, visible, fidelity, color, p, d1, d2, e1, e2, s, m }

Penetration { name, visible, fidelity, color, pl, sproj, snegproj, m }

PlanarPlate { name, visible, fidelity, color, pl, t, m }

PlanarWebStiffener { name, visible, fidelity, color, pl, p, r, e1, e2, trim1, trim2, s1, s2, m }

RingStiffener { name, visible, fidelity, color, pl, amin, amax, m, mf, s }

IGESEntity { name, visible, fidelity, color, igesfile, t, m, ashierarchy }

IGESTraceStiffener { name, visible, fidelity, color, igesfile, dw, df, p, e1, e2, s1, s2, m }

HullFrame { name, visible, fidelity, color, pl, p, e1, e2, s, m }

B.1.4 Property Zone Entities

MZone { name, visible, fidelity, color, objref, m }

TZone { name, visible, fidelity, color, objref, t, adjustment, offsetdir }

B.1.5 Auxiliary Data Entities

AppliesTo { objref }

AttachBase { objref, normside }

Penetrate { objref }

Circle { r }

Corner { x, y }

RoundedRect { lx, ly, r }

OblongCircle { d, r }

Trim { objref, s, m, off, p }

PlateRefPt { p, reverse }

StressStrainCurvePoint { str, stn }

Value { v }

B.1.6 Planar Intersection Entities

PlanarIntersection { pl, path }

IntersectWith { objref, omit }

B.1.7 Material Type Entities

Material_Iso { name, YoungsModulus, PoissonsRatio, Density, YieldStress,

B.1.8 Section Type Entities

Section_Angle { name, d, tw, wf, tf }

Section_Channel { name, w, d, tw, td }

Section_CHS { name, r, t }

Section_Flatbar { name, d, tw }

Section_I { name, d, tw, wf1, tf1, wf2, tf2 }

Section_RHS { name, w, d, t }

Section_SHS { name, w, t }

Section_T { name, d, tw, wf, tf }

B.2 RMGScript Attribute Value Syntax

B.2.1 Units Specifiers

Valid unit specifiers: m, mm, ft, in

B.2.2 Object Reference Syntax

Object reference syntax: [name]

Object reference attribute syntax: [name].attr.subattr

B.2.3 Anonymous Object Syntax

Anonymous object syntax: {TYPE(param1,param2)}

Anonymous object attribute syntax: {TYPE(param1,param2)}.attr.subattr

B.2.4 Inline Arithmetic Operators

Legend: val=scalar value, vec=vector, pnt=point

'–' operator: –val=>val or –vec=>vec

'*' operator: val*val=>val or vec*val=>vec

'/' operator: val/val=>val or vec/val=>vec

'+' operator: val+val=>val or pnt+vec=>pnt or vec+vec=>vec

'-' operator: val-val=>val or pnt-vec=>pnt or pnt-pnt=>vec or vec-vec=>vec

'(' and ')' allowed for precedence

B.2.5 Values

VALUE (v)

LENGTH (x, y, z) || (p1, p2) || (v)

B.2.6 Points

POINT (x, y, z)

INTPOINT (obj, o, v)

ROTPOINTCW (p, o, v, a)

ROTPOINTCCW (p, o, v, a)

XLOC (x)

YLOC (y)

ZLOC (z)

B.2.7 Vectors

VECTOR (x, y, z)

INTNORMAL (obj, o, v)

NORMVECT (x, y, z) || (p1, p2) || (v)

AFTDIR ()

STARDIR ()

UPDIR ()

FWDDIR ()

PORTDIR ()

DOWNDIR ()

B.2.8 Planes

PLANE (o, n, x) || (o, n)

XYPLANE (o, x, y)

THREEPTPLANE (p1, p2, p3)

ROTPLANECW (o, n, v, a)

ROTPLANECCW (o, n, v, a)

UPPLANE (z) || (p)

STARPLANE (y) || (p)

AFTPLANE (x) || (p)

DOWNPLANE (z) || (p)

PORTPLANE (y) || (p)

FWDPLANE (x) || (p)

B.2.9 Conics

CONIC (pl, r1, r2, h, amin, amax) || (pl, r1, r2, h)

List of symbols/abbreviations/acronyms/initialisms

API	Application programming interface
CAD	Computer aided design
COTS	Commercial off the shelf
CPF	Canadian Patrol Frigate
DIME	Data Interface Management Engine
DIR	Defence industrial research
DLL	Dynamic link library
DND	Department of National Defence
DRDC	Defence Research & Development Canada (Atlantic)
DSL	Domain specific language
FEA	Finite element analysis
GUI	Graphical user interface
IGES	Initial Graphics Exchange Specification
ISSMM	Improved ship structures maintenance management
IST	ISSMM Software Tool
LCM	Lifecycle management
LR	Lloyd's Register of Shipping
MCDV	Maritime Coastal Defence Vessel
NURBS	Non-uniform rational B-spline
RMGScript	Relational Meshable Geometry Script
SDA	Structural design assessment
SPM	(ship) single product model
SSX	Ship structure exchange
SubSAS	Submarine Structural Analysis Suite
XML	Extensible markup language
XSD	XML schema definition
XSLT	XML transformation stylesheets

Distribution list

Document No.: DRDC Atlantic CR 2012-003

LIST PART 1: Internal Distribution by Centre

2 GL/NPSS (1 hard copy, 1 CD)
1 SH/WP
1 J.R. MacKay
3 DRDC Atlantic Library (1 hard copy, 2 CDs)

7 TOTAL LIST PART 1

LIST PART 2: External Distribution by DRDKIM

1 Library & Archives Canada, Attn: Military Archivist, Government Records Branch

1 DRDKIM

1 National Defence Headquarters
DGMEPM/DMSS 2
555 Blvd de la Carriere
Hull, PQ K1A 0K2

3 TOTAL LIST PART 2

10 TOTAL COPIES REQUIRED

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.) Martec Limited 1888 Brunswick Street, Suite 400 Halifax, Nova Scotia B3J 3J8	2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.) UNCLASSIFIED (NON-CONTROLLED GOODS) DMC A REVIEW: GCEC June 2010

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Determination of the Most Efficient Database Format for Ship Analysis Models

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

MacAdam, T.; Wallace, J.; Lichodzijewski, M.

5. DATE OF PUBLICATION (Month and year of publication of document.) April 2012	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.) 82	6b. NO. OF REFS (Total cited in document.) 13

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence R&D Canada – Atlantic
9 Grove Street
P.O. Box 1012
Dartmouth, Nova Scotia B2Y 3Z7

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.) 11ge04	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.) W7707-088100
10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.) DRDC Atlantic CR 2012-003

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

This report sets out a high-level proposal for a data schema capable of storing ship model data suitable for various types of lifecycle management engineering analyses. A critical analysis of two existing ship model data formats, RMGScript serving global finite element analysis and Ship Structure Exchange serving ship design assessment, was undertaken as the basis for the schema proposed. The new schema combines the strengths of both existing formats and sets forth guidelines for implementing extensions with minimal disruption on applications that make use of it.

Le présent rapport établit une proposition de haut niveau pour un schéma de données capable d'emmagasiner des données concernant un modèle convenant à divers types d'analyses techniques de gestion de cycle de vie. Une analyse critique de deux formats de données de modèle existants (RMGScript : sert une analyse générale par éléments finis, et Ship Structure Exchange : sert une évaluation de conception de navire) a été entreprise comme base du schéma proposé. Le nouveau schéma combine les forces des deux formats existants et explique des lignes directrices pour mettre en œuvre des extensions en perturbant le moins possible les applications qui utilisent cela.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

ship modelling; single product model; SPM; RMGScript; XML schema; DIME;

This page intentionally left blank.

Defence R&D Canada

Canada's leader in defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

DEFENCE DÉFENSE

www.drdc-rddc.gc.ca