Vetronics Architect: Modelling and Simulation

Dr George Valsamakis, Dr Ian Colwill
Vetronics Research Centre
University of Brighton
UNITED KINGDOM

g.valsamakis@vetronics.org i.colwill@vetronics.org

ABSTRACT

This paper introduces the Vetronics Architect (VA), a methodology and concept toolset for the behavioural simulation of formal systems engineering models of modern military vehicular systems in a synthetic 3D environment. The main driver of this work is the shortcomings of existing design systems in the face of ever increasing complexity and capability of military vehicle platforms and systems, and the subsequent need to identify opportunity and risk early in the design cycle of a system. Alongside new technological capabilities and their need to interconnect and interoperate, emerging defence standards and specifications create the need for efficient means of systems design for supporting their development so that they can be validated and verified. The principle aim of this study is to take advantage of the information available about systems at their design stages, and analyse its validity in scenarios operating in a simulated environment, along with other subsystems. A new vetronics systems design, modelling and simulation methodology that focuses on concept exploration, early de-risking and system acceptance through behavioural simulation is presented. Its overarching aim is to enable the identification of risks and opportunities early in the design cycle through improved understanding of the design within a modern vetronics context. A principle manifestation of this aim is the refining and driving of user requirements for vetronics systems. To achieve this, an existing Systems Engineering (SE) modelling process is enhanced by providing the capability of executing system models in a 3D behavioural simulation environment.

1.0 INTRODUCTION

New levels of capability, an increasing pace of technological progress and increasing levels of functionality achieved through sub-system interoperation represent a highly complex challenge for vehicle platform architects. Vetronics platform behaviour is in nature a function of the interaction between numerous complex sub-systems and components. This naturally obscures the ultimate real-world performance of the design from the perspective of the system designer and stakeholders.

Additional challenge to the designer is presented by the demands of an increasing number of intersecting open architecture standards that new platforms must comply with in the area of vetronics. Many of the standards are growing in scope and degree of prescription. Of particular interest to this work are finding the implication of new standards and specifications introduced for cross-domain systems integration and interoperability under the UK Land Open Systems Architecture (LOSA), the Generic Vehicle, Soldier, and Base Architectures (GVA, GSA, and GBA), as well as the upcoming NATO Generic Vehicle Architecture (NGVA) STANAG 4754.

Further, current design systems may not adequately represent some vetronics aspects and areas of interest which affect a system’s design. Important principles described by emerging open architectures standards in defence and the data models that accompany them can be represented within the design lifecycle more efficiently. Other concerns arising from the complex nature of modern platforms include technology and data ownership rights and subsequent significant hidden costs of modification for a long lived vehicle platform projects.

1.1 Background and Motivation

Processes of Model Based Systems Engineering (MBSE) prescribe system design stages to produce results that satisfy given requirements and provide full traceability of them across the design stages. A typical process is and tools are chosen for the scope of this work. The IBM® *Rational® Integrated Systems/Embedded Software Development Process Harmony™* [1] is one of these processes and follows the classical "V" development lifecycle. The tools used to help carry out this methodology are the IBM® Rational® suite of tools that include IBM® Rational® DOORS® for the analysis and management of the user and system requirements, and IBM® Rational® Rhapsody® for the systems design. The Harmony™ Systems Engineering (HarmonySE™) toolkit provides helper automations for aiding the designer to go through the process stages. This process provides guidelines to perform the system design by analysing requirements, to its functional analysis and to its implementation. In the past decade, entertainment technology has seen many advances in rapid prototyping gaming environments. At a stage before the system implementation on real hardware, it is perceived that the combination of extracted elements of interest from the system design and their virtual prototyping in such an environment can provide early de-risking capabilities and save costs.

A rapid development environment that can provide sufficient flexibility to conceptualise subsystem communications and overall vehicle behaviour in a battlefield scenario, over focused areas of vetronics interest has been chosen. Unity Technologies® Unity3D™ software development environment provides such capabilities. The study combines the system modelling artifacts extracted throughout the HarmonySE™ design process to configure a Unity3D™ battlefield simulation environment and experiment with concepts that can be quickly re-iterated and assessed.

Areas of special interest in modern vetronics targeted by this process are:

- GVA Def Stan 23-09 and emerging standards such as the NGVA.
- Data Models and Data Distribution Service (DDS).
- Horizontal Integration.
- Data / Architecture Ownership and Management.
- System acceptance through simulation.
- Component / Technology Substitution.

The methodology defined is referred to as the Vetronics Architect Process (VA Process) and it takes place alongside the different stages of the HarmonySE™ guidelines. Figure 1 shows this workflow in a state chart diagram.

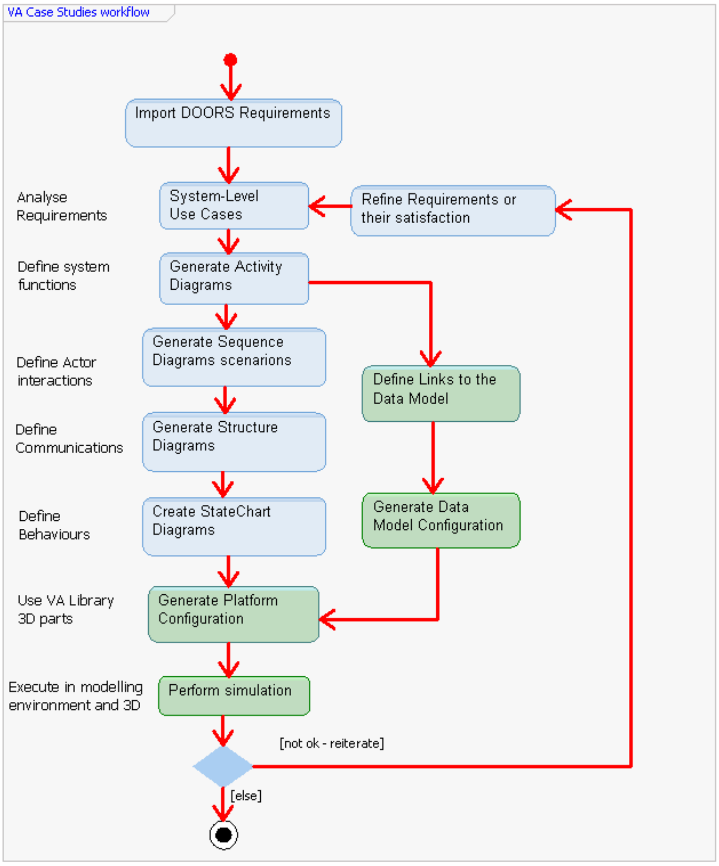

VA Case Studies workflow
Import DOORS Requirements
Analyse Requirements
System-Level Use Cases
Refine Requirements or their satisfaction
Define system functions
Generate Activity Diagrams
Define Actor interactions
Generate Sequence Diagrams scenarions
Define Links to the Data Model
Define Communications
Generate Structure Diagrams
Define Behaviours
Create StateChart Diagrams
Generate Data Model Configuration
Use VA Library 3D parts
Generate Platform Configuration
Execute in modelling environment and 3D
Perform simulation
[not ok - reiterate]
[else]

Figure 1: VA Process along HarmonySE™ stages workflow

The states in blue are the design stages of HarmonySE™ and the ones in green are specialised diagram types created to support extracting this information. These custom diagrams are created so that the HarmonySE™ or other MBSE based process is not affected by the VA simulation. The purpose of the modelling part of the methodology is to extract the sufficient information from the system model to configure the simulation. These typically include the data being communicated between components that the user expects to be of certain value limits in order to meet requirements. The data model interactions between are mandated by the defence standards and for the purpose of this work the UK Defstan 23-009 GVA Land Data Model [3] was used.

The following sections shall describe the system design modelling and simulation respectively. Some case studies were explored to validate different parts of the methodology. These make use of a virtual version of an e-Buggy, which represents a real-world platform that was also used during these tests. These case studies were:

- VA3D eBuggy Local Situational Awareness (LSA)
- VA3D eBuggy Navigation
- VA3D eBuggy Automotive

Each of the experiments focuses on specific aspects in the system design and simulation. The LSA experiment implements camera views around the vehicle and tests their interaction with the driver input. The Navigation experiment focuses on the navigation data model interactions between a GPS receiver and the navigation display in the e-Buggy crew station. The Automotive experiment focuses on performing trade-offs of drive train parameters for the e-Buggy by assessing performance and costs when substituting different gearboxes, electric engines, and batteries in the modelling environment. References to these case studies shall be used in the following sections to describe the use of the methodology.

2.0 MODELLING

A methodology is defined, referred to as the Vetronics Architecture Process (VA Process), whose purpose is to help pinpoint and extract certain vetronics related characteristics that emerge during the prescribed MBSE system design phases. The information extracted can then be used perform simulations in the synthetic 3D environment that is focused on specific aspects. The overall purpose of this is to drive user requirements by assessing their execution.

For interoperability and integration purposes, new open architectures standards mandate the satisfaction specific requirements. In a typical system requirements analysis, these are associated with the stakeholder requirements to produce the functional and system requirements.

HarmonySE™ makes use of the OMG SysML language [2] to formally describe the system design. The language uses Requirement Diagrams that capture their various parameters and links them with the Use Cases that satisfy them. HarmonySE™ bases its analysis in these Use Cases and during its Functional Analysis stage it derives the functionality that the system needs to perform in order to meet them.

As SysML is based on a subset of the Unified Modelling Language (UML) it inherits the support to design a Domain-Specific Language subset. This allows for the creation of the VA UML Profile which contains the terminology and entities required for the specification of the VA diagrams. This profile can be used inside the same tools that perform the MBSE process and be described alongside in the system design. This respects the workflow of the HarmonySE™ process and toolkit and does not alter it in order to perform the VA simulation.

Figure 2 shows the overall VA Process overview with its artifacts coloured in green, and how it stands along the HarmonySE™ Process.

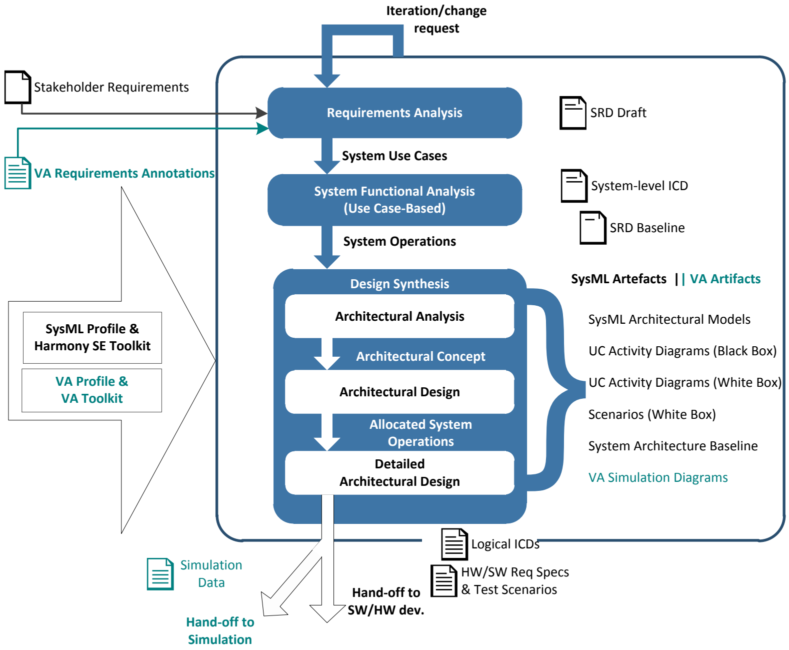


Figure 2: Vetronics Architect & HarmonySE™ SysML Processes Overview

VA Requirements Annotation: These are requirements that are normally imported from IBM® Rational® DOORS® modules. They contain VA specific attributes with information such as their categorisation and trace to standard specifications such as the UK GVA Def Stan 23-009 [3] and the upcoming NGVA STANAG 4754 [4]

VA Profile and Artefacts: These contain the vetronics architecture vocabulary (UML and SysML artefacts) to be used in Rhapsody® modelling environment. They provide the basic building blocks for carrying out the VA models configuration.

VA Simulation Diagrams: This is a new type of diagram, defined in Rhapsody® and made available by the VA Profile. The diagram is used to configure the simulation after the system has been designed during the functional analysis and design synthesis phases of the HarmonySE™ modelling process. The architecture's structural elements, SysML Blocks, are linked in this diagram using custom VA notation.

VA Simulation Data: This is a document in XML Metadata Interchange (XMI) generic format that results from the VA Simulation Diagrams.

The above artifacts are used to generate the following information:

- Platform Configuration Diagram
- Data Model Configuration Diagram
- Simulation Environment Configuration

The Platform Configuration refers to the structural blocks that comprise the system and the user has selected for them to have a representation in the 3D environment. The Data Model Configuration refers to the Data Model Data Distribution Service (DDS) communication elements that the system must implement for GVA or NGVA compliance.

The hierarchy of the basic building blocks of the VA Diagrams are shown in Figure 3. *VA Platform* refers to the virtual platform of the vehicle that has a 3D representation in the simulation environment. The *VALibraryClass* is the element for describing any subsystems contained in the platform. The *VABlock* is a placeholder that gets associated with the system model created by HarmonySE™. The *VADDSTopic* block is an abstraction for describing DDS topics that a component either subscribes or publishes to.

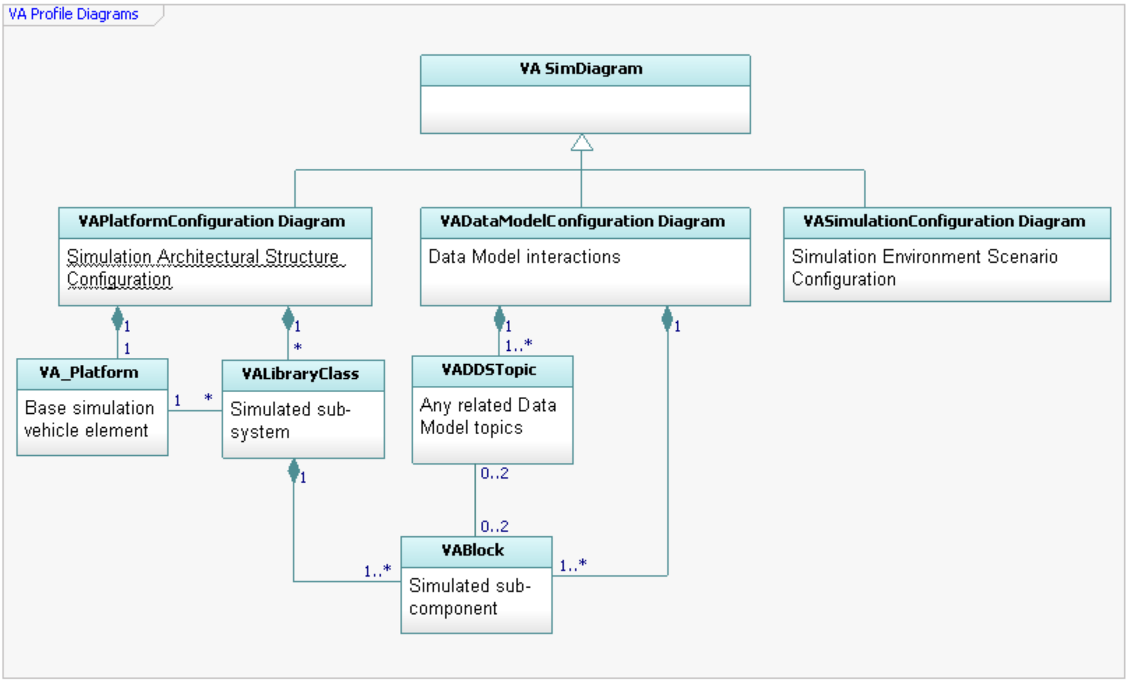

VA Profile Diagrams
VA SimDiagram
VAPlatformConfiguration Diagram
Simulation Architectural Structure Configuration
VADataModelConfiguration Diagram
Data Model interactions
VASimulationConfiguration Diagram
Simulation Environment Scenario Configuration
VA_Platform
Base simulation vehicle element
VALibraryClass
Simulated sub-system
VADDSTopic
Any related Data Model topics
VABlock
Simulated sub-component
1
1
1
*
1
*
1
1..*
1
1
0..2
0..2
1..*
1..*

Figure 3: VA Simulation Diagrams and UML Artifacts hierarchy.

An example system model of the Navigation case study and Data Model configuration for the communication interactions are shown in Figure 4 and Figure 5 respectively. It can be seen that the *VABlock* type elements (in green) are instances of the SysML Block (in blue) elements that were created in the Design Synthesis stage of the HarmoneSE™ analysis. Using the modelling tools, these can be traced back to Sequence, Activity Diagrams and consequently to the Use Case that they implement, which in turn leads to the Requirements that they satisfy.

The green *VABlock* type blocks are elements that are have virtual representions in the simulation environment. These same blocks also appear in the *VA Data Model Configuration Diagram* in Figure 5 that captures their publish/subscribe relationships with the DDS topics (in yellow) imported from the Rhapsody® project of the UK GVA Data Model.

This way, the VA elements in the model act as an abstraction between the HarmonySE™ model and the GVA Data Model. Even though the HarmonySE™ process considers a Service Oriented Architectures (SOA) for the communication “contracts” between subcomponents of the system, using this abstraction it is possible to describe the data-centric communication approach that is mandated in a DDS network.

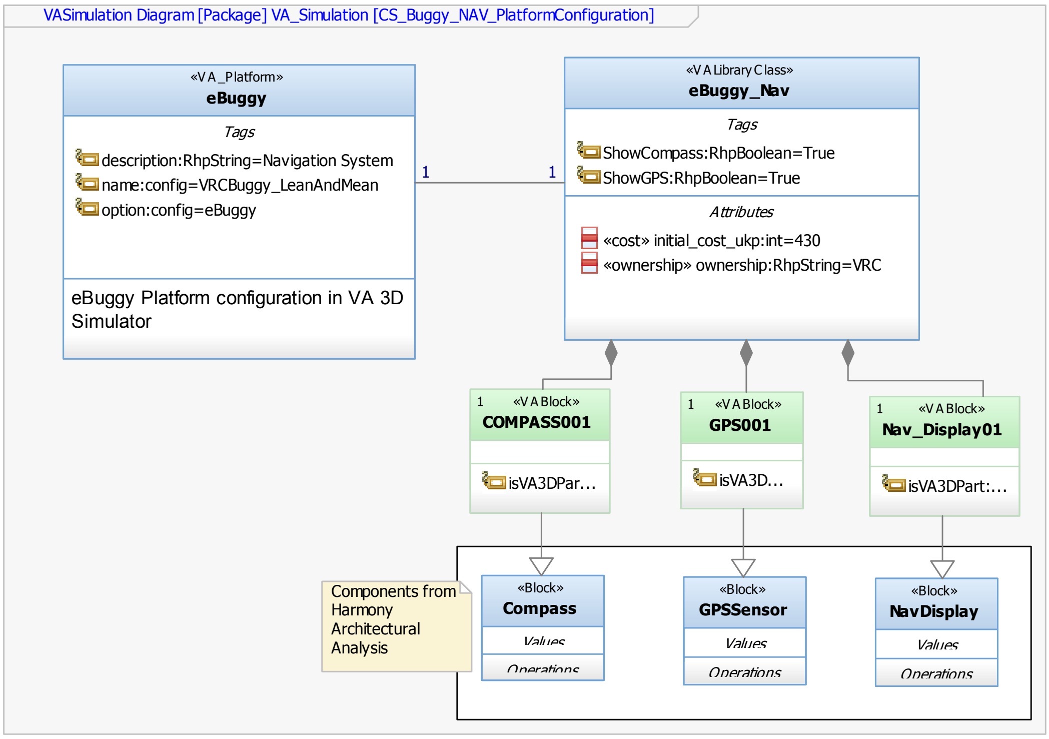

VASimulation Diagram [Package] VA_Simulation [CS_Buggy_NAV_PlatformConfiguration]
«V A_Platform»
eBuggy
Tags
description:RhpString=Navigation System
name:config=VRCBuggy_LeanAndMean
option:config=eBuggy
eBuggy Platform configuration in VA 3D Simulator
1
1
«V A Library C lass»
eBuggy_Nav
Tags
ShowCompass:RhpBoolean=True
ShowGPS:RhpBoolean=True
Attributes
«cost» initial_cost_ukp:int=430
«ownership» ownership:RhpString=VRC
1 «V A Block»
COMPASS001
isVA3DPar...
1 «V A Block»
GPS001
isVA3D...
1 «V A Block»
Nav_Display01
isVA3DPart:...
Components from Harmony Architectural Analysis
«Block»
Compass
Values
Operations
«Block»
GPSSensor
Values
Operations
«Block»
NavDisplay
Values
Operations

Figure 4: e-Buggy Navigation VA Platform Configuration

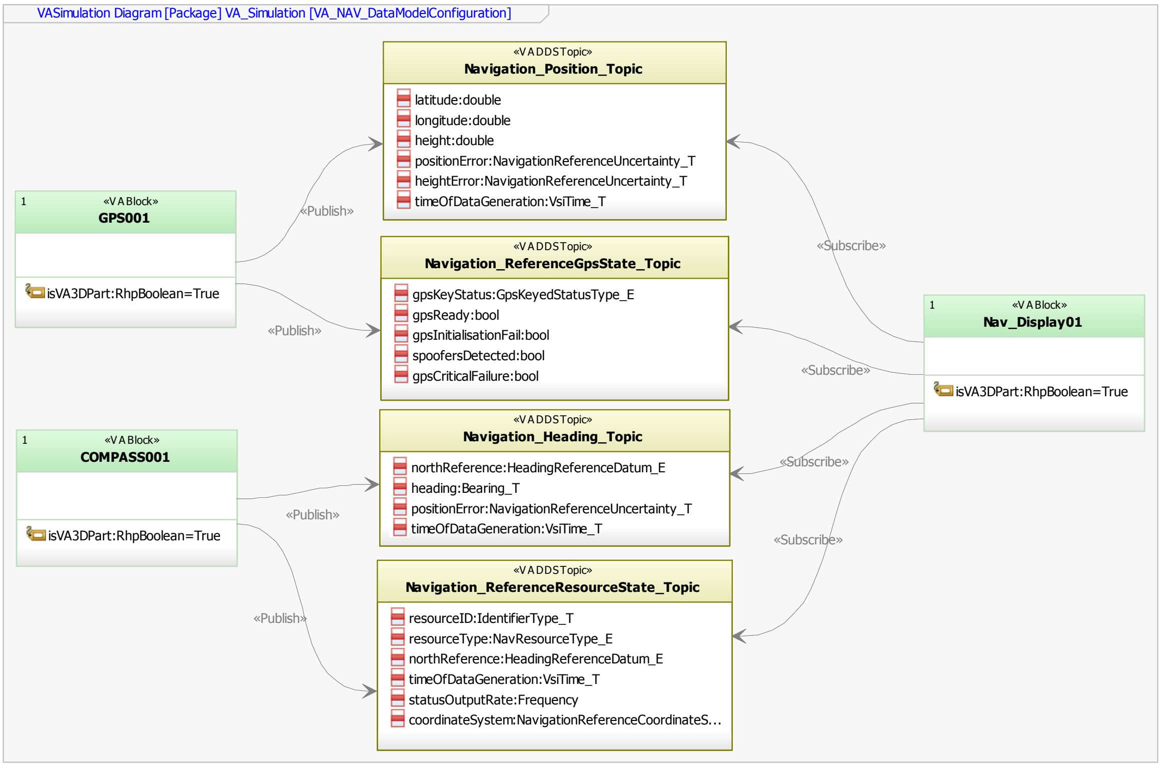
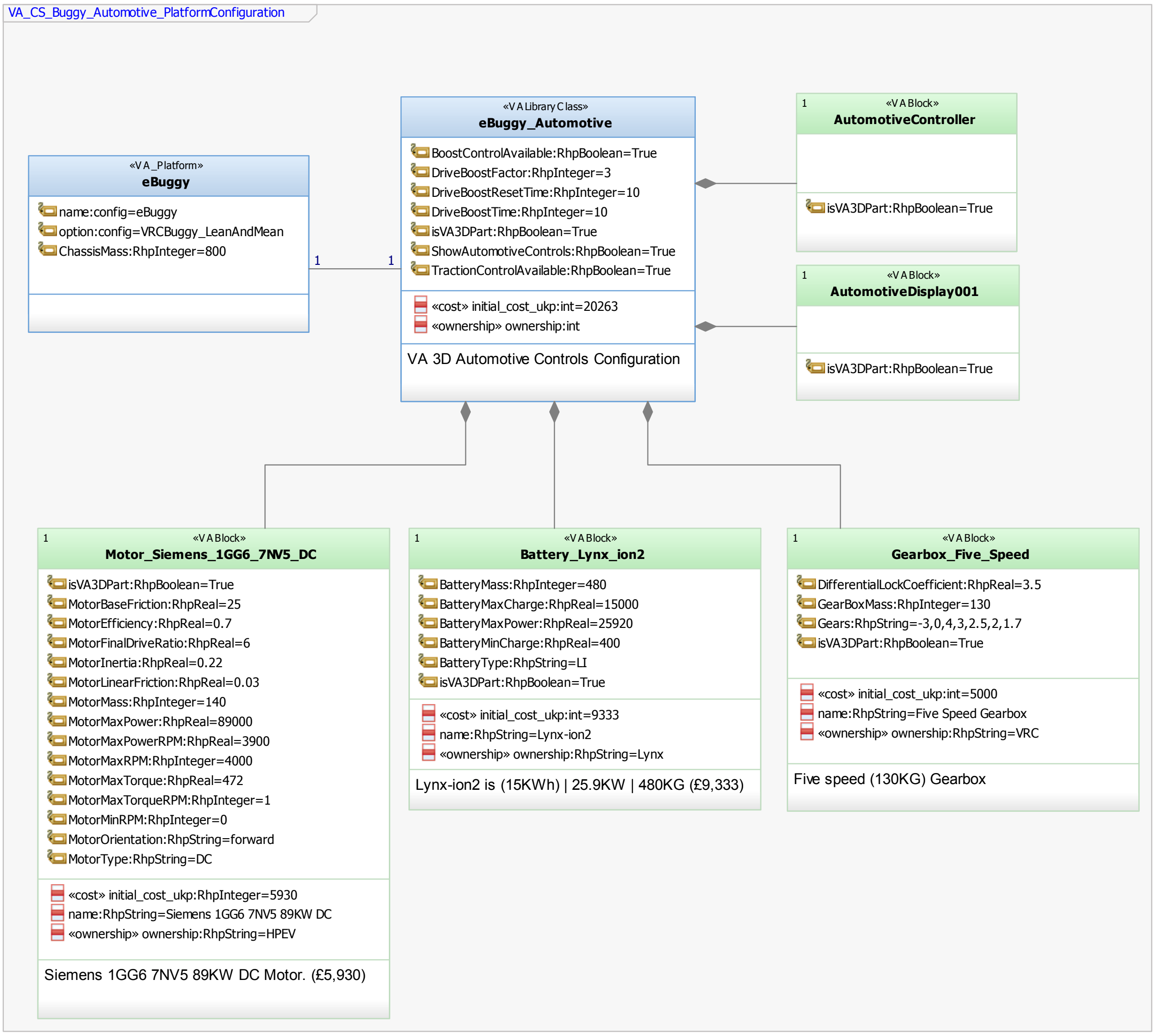

VASimulation Diagram [Package] VA_Simulation [VA_NAV_DataModelConfiguration]
«V A DDSTopic»
Navigation_Position_Topic
latitude:double
longitude:double
height:double
positionError:NavigationReferenceUncertainty_T
heightError:NavigationReferenceUncertainty_T
timeOfDataGeneration:VsiTime_T
«V A DDSTopic»
Navigation_ReferenceGpsState_Topic
gpsKeyStatus:GpsKeyedStatusType_E
gpsReady:bool
gpsInitialisationFail:bool
spoofersDetected:bool
gpsCriticalFailure:bool
«V A DDSTopic»
Navigation_Heading_Topic
northReference:HeadingReferenceDatum_E
heading:Bearing_T
positionError:NavigationReferenceUncertainty_T
timeOfDataGeneration:VsiTime_T
«V A DDSTopic»
Navigation_ReferenceResourceState_Topic
resourceID:IdentifierType_T
resourceType:NavResourceType_E
northReference:HeadingReferenceDatum_E
timeOfDataGeneration:VsiTime_T
statusOutputRate:Frequency
coordinateSystem:NavigationReferenceCoordinateS...
1 «V A Block»
GPS001
isVA3DPart:RhpBoolean=True
1 «V A Block»
COMPASS001
isVA3DPart:RhpBoolean=True
1 «V A Block»
Nav_Display01
isVA3DPart:RhpBoolean=True
«Publish»
«Publish»
«Publish»
«Publish»
«Subscribe»
«Subscribe»
«Subscribe»
«Subscribe»

Figure 5: e-Buggy Navigation VA Data Model Configuration

An example VA Platform Configuration diagram for the Automotive case study is shown in Figure 6. This VA Diagram shows artifacts related to costs and data ownerships as attributes in the *VABlocks* which are provided as re-usable libraries. The different parameters in each *VABlock* reflect the way the virtual representation of the platform behaves and this is a basis of performing quick trade-offs complementary to the trade-off analysis with metrics and performance analysis that HarmonySE™ prescribes.



«V A_Platform»
eBuggy
name:config=eBuggy
option:config=VRCBuggy_LeanAndMean
ChassisMass:RhpInteger=800

«V A Library C lass»
eBuggy_Automotive
BoostControlAvailable:RhpBoolean=True
DriveBoostFactor:RhpInteger=3
DriveBoostResetTime:RhpInteger=10
DriveBoostTime:RhpInteger=10
isVA3DPart:RhpBoolean=True
ShowAutomotiveControls:RhpBoolean=True
TractionControlAvailable:RhpBoolean=True
«cost» initial_cost_ukp:int=20263
«ownership» ownership:int
VA 3D Automotive Controls Configuration

1 «V A Block»
AutomotiveController
isVA3DPart:RhpBoolean=True

1 «V A Block»
AutomotiveDisplay001
isVA3DPart:RhpBoolean=True

1 «V A Block»
Motor_Siemens_1GG6_7NW5_DC
isVA3DPart:RhpBoolean=True
MotorBaseFriction:RhpReal=25
MotorEfficiency:RhpReal=0.7
MotorFinalDriveRatio:RhpReal=6
MotorInertia:RhpReal=0.22
MotorLinearFriction:RhpReal=0.03
MotorMass:RhpInteger=140
MotorMaxPower:RhpReal=89000
MotorMaxPowerRPM:RhpReal=3900
MotorMaxRPM:RhpInteger=4000
MotorMaxTorque:RhpReal=472
MotorMaxTorqueRPM:RhpInteger=1
MotorMinRPM:RhpInteger=0
MotorOrientation:RhpString=forward
MotorType:RhpString=DC
«cost» initial_cost_ukp:RhpInteger=5930
name:RhpString=Siemens 1GG6 7NV5 89KW DC
«ownership» ownership:RhpString=HPEV
Siemens 1GG6 7NV5 89KW DC Motor. (£5,930)

1 «V A Block»
Battery_Lynx_ion2
BatteryMass:RhpInteger=480
BatteryMaxCharge:RhpReal=15000
BatteryMaxPower:RhpReal=25920
BatteryMinCharge:RhpReal=400
BatteryType:RhpString=LI
isVA3DPart:RhpBoolean=True
«cost» initial_cost_ukp:int=9333
name:RhpString=Lynx-ion2
«ownership» ownership:RhpString=Lynx
Lynx-ion2 is (15KWh) | 25.9KW | 480KG (£9,333)

1 «V A Block»
Gearbox_Five_Speed
DifferentialLockCoefficient:RhpReal=3.5
GearBoxMass:RhpInteger=130
Gears:RhpString=-3,0,4,3,2.5,2,1.7
isVA3DPart:RhpBoolean=True
«cost» initial_cost_ukp:int=5000
name:RhpString=Five Speed Gearbox
«ownership» ownership:RhpString=VRC
Five speed (130KG) Gearbox

Figure 6: e-Buggy Automotive VA Platform Configuration

The VA diagrams are exported as XML configuration files that the 3D simulation reads and configures its execution. This allows for quick iterations through costless reconfiguration of the simulation from the modelling environment. The attributes inside the *VADDSTopics* that contain values are associated to requirements that trace back to the requirements imported from IBM® Rational® DOORS® during the Requirements Analysis. Using this technique, the Data Model interactions as well as the overall behaviour of the system can then be audited and assessed against user defined constraints that trace back to the early stages of the system analysis.

3.0 SIMULATION

The VA3D toolset is implemented as a set of libraries and tools that exist within the Unity3D™ environment. The complete toolkit comprises several tools for creation, operation and investigating of data model backed simulations (Figure 7).

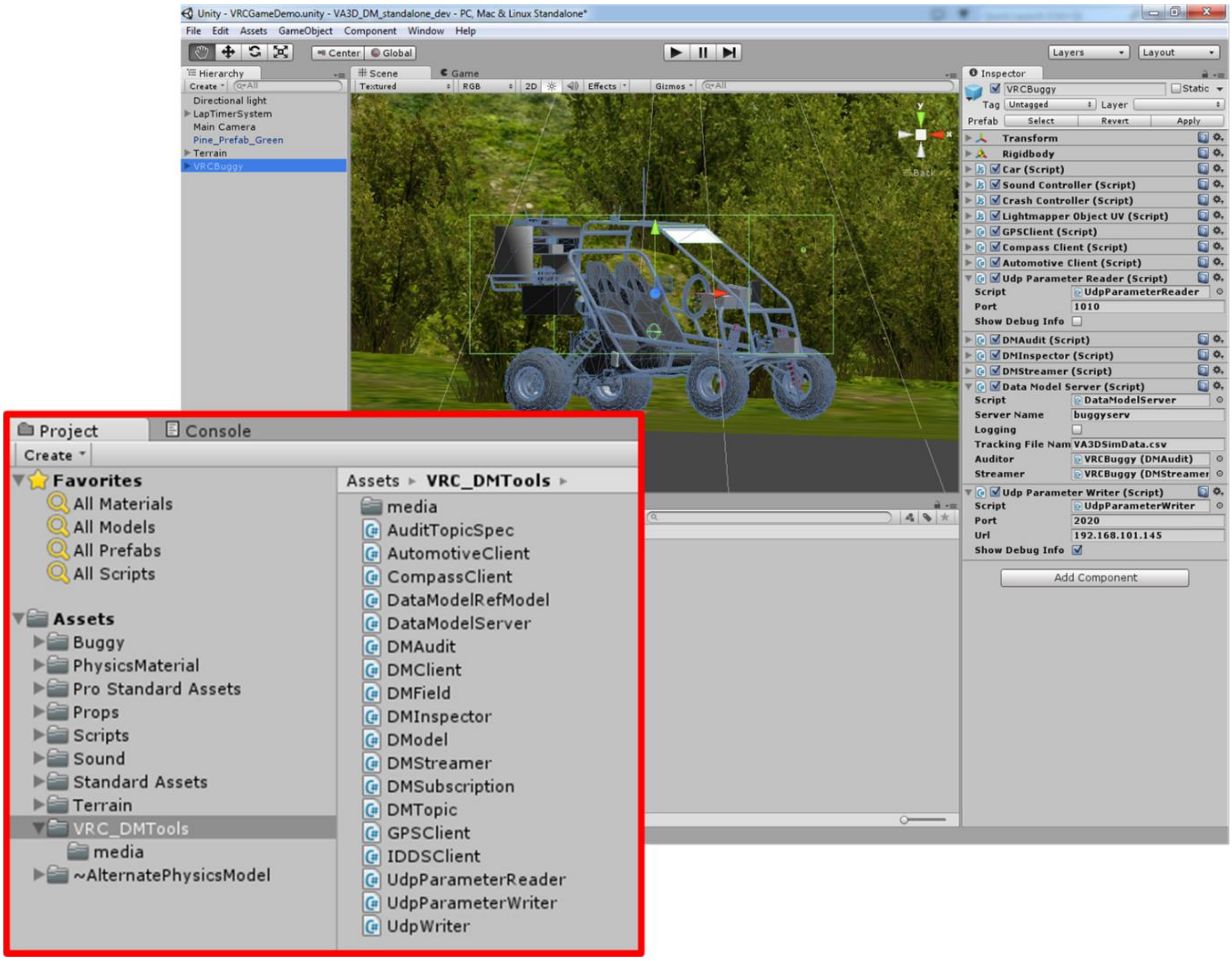
Data Model Service Infrastructure Data Model Server Data Model Data Model Topic Data Model Field Data Model Subscription	Provides the entities that enable a simulated data distribution service to be built. The entities work together to provide a publish and subscribe data sharing service.
Data Model Interface Data Model Client Data Model Client Examples	Allows all platform entities to interface with the data model service in order to share data abstract from their individual implementation complexities.
Network Tools Data Model Streaming Server UDP Parameter Reader UDP Parameter Writer	Adds the capability for external network connected entities to participate in the simulation. This includes controls, displays, platform equipment and cooperating simulation tools.
Inspection and Debugging Tools Data Model Audit Tool Data Model Inspector Requirement Tracking tool	Allows users to examine and record live data model activity and to compare to a known standard.
Configuration Tools XML Configuration Tool	Allows system design information to be passed to the simulation in order to control simulation behaviours in accordance with system design changes.

Figure 7: VA 3D Tools Overview

3.1 Data Model Service and Interfaces

At the heart of the tool set are *Data Model Infrastructure* tools. These allow any 3D entity to host a data distribution service comprising a data model host and numerous clients. With the Data Model Interface tools, the clients can individually establish, publish and subscribe to relationships with the servers in order to create an overall data model powered communication model.

The *Data Model Inspection and Debugging* tool group provides the user live Heads Up Display (HUD) type views of the running data model, allowing the user to monitor the content of the live data model changing in real-time. The inspector tool also allows selected data model topic live data to be streamed to disk for later analysis. The *data model audit* tool compares the live data model activity with a reference model specification that has been previously described in XML or output by the systems design tools. Any deviations from the specification are recorded in a data model ‘audit’ report file, and optionally displayed onscreen as errors or warnings during simulation. Figure 8 shows the Unity3D™ development environment with the VA toolset library.



Unity - VRCGameDemo.unity - VA3D_DM_standalone_dev - PC, Mac & Linux Standalone*

Hierarchy

- Directional light
- LapTimerSystem
- Main Camera
- Pine_Prefab_Green
- Terrain
- VRCBuggy

Inspector

☑ VRCBuggy □ Static

Tag: Untagged Layer

Prefab: Select Revert Apply

- Transform
- Rigidbody
- ☑ Car (Script)
- ☑ Sound Controller (Script)
- ☑ Crash Controller (Script)
- ☑ Lightmapper Object UV (Script)
- ☑ GPSClient (Script)
- ☑ Compass Client (Script)
- ☑ Automotive Client (Script)
- ☑ Udp Parameter Reader (Script)
  - Script: UdpParameterReader
  - Port: 1010
  - Show Debug Info: □
- ☑ DMAudit (Script)
- ☑ DMInspector (Script)
- ☑ DMStreamer (Script)
- ☑ Data Model Server (Script)
  - Script: DataModelServer
  - Server Name: buggyserv
  - Logging: □
  - Tracking File Nam: VA3DSimData.csv
  - Auditor: VRCBuggy (DMAudit)
  - Streamer: VRCBuggy (DMStreamer
- ☑ Udp Parameter Writer (Script)
  - Script: UdpParameterWriter
  - Port: 2020
  - Url: 192.168.101.145
  - Show Debug Info: ☑

Project

Favorites

- All Materials
- All Models
- All Prefabs
- All Scripts

Assets

- Buggy
- PhysicsMaterial
- Pro Standard Assets
- Props
- Scripts
- Sound
- Standard Assets
- Terrain
- VRC_DMTools
  - media
- ~AlternatePhysicsModel

Assets ▸ VRC_DMTools ▸

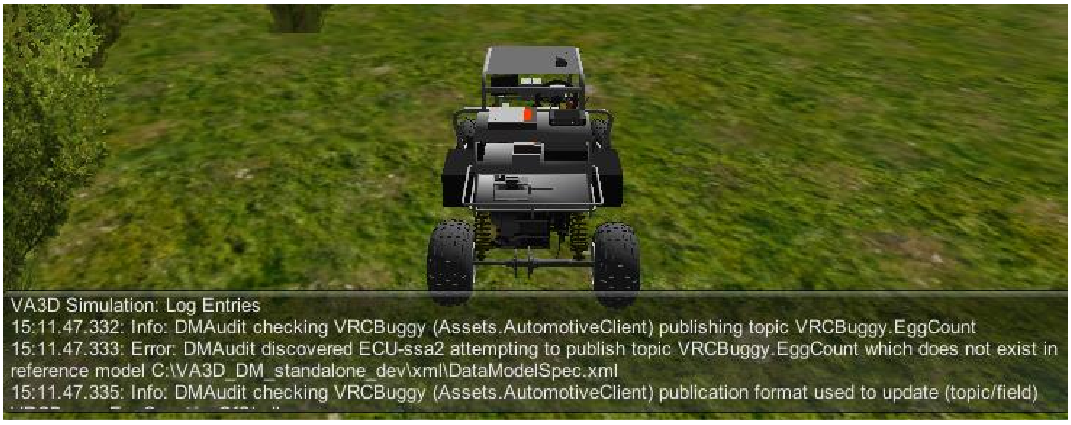
- media
- AuditTopicSpec
- AutomotiveClient
- CompassClient
- DataModelRefModel
- DataModelServer
- DMAudit
- DMClient
- DMField
- DMInspector
- DModel
- DMStreamer
- DMSubscription
- DMTopic
- GPSClient
- IDDSClient
- UdpParameterReader
- UdpParameterWriter
- UdpWriter

Figure 8: Modelled e-Buggy in the 3D simulation environment.

3.2 Data Model Audit Tool

The data model audit tool provides verification that the simulated system data model use complies with a corresponding system design. The tool operates in two stages.

- The system reads an XML description of the data model design. From this description, a reference data model is constructed. The model describes the legal use of the data model in terms of structure, format and legal access.
- In real-time (during simulation), all data model activity is compared to the reference model object. Any irregularities are stored in a log file and optionally are displayed on the Log Viewer (Figure 9).



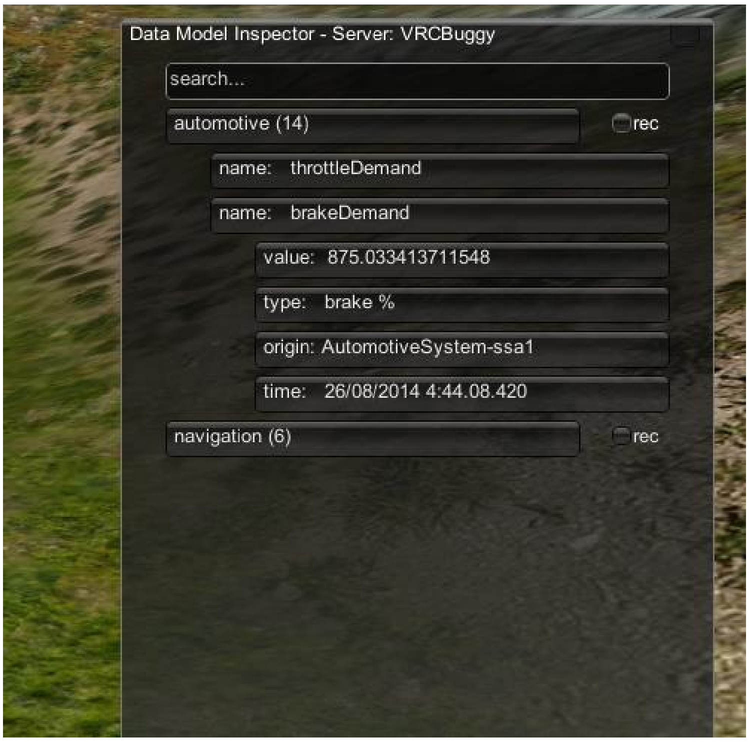
```
VA3D Simulation: Log Entries
15:11.47.332: Info: DMAudit checking VRCBuggy (Assets.AutomotiveClient) publishing topic VRCBuggy.EggCount
15:11.47.333: Error: DMAudit discovered ECU-ssa2 attempting to publish topic VRCBuggy.EggCount which does not exist in
reference model C:\VA3D_DM_standalone_dev\xml\DataModelSpec.xml
15:11.47.335: Info: DMAudit checking VRCBuggy (Assets.AutomotiveClient) publication format used to update (topic/field)
```

Figure 9: Screenshot Showing Dmaudit Tool Reporting Publication to a Topic in Contravention of The Reference Model.

The audit tool user controls allow the selection of the audit information required (info / warnings/ errors). Info messages describe each individual test carried out. Warnings describe minor issues such as incorrect character cases. Errors describe all deviations from the supplied XML reference model. The system allows live data model activity to be tested against a known standard with possible applications in standards compliance or integration troubleshooting.

3.3 Data Model Inspector

The *data model inspector* tool creates a live simulation HUD view of the current data model activity for the user. The inspector includes a search facility allowing the user to quickly locate topics in larger data model systems. Once located, the user can expand the data model to view topic fields or live individual field values (Figure 10).



Data Model Inspector - Server: VRCBuggy

search...

automotive (14) ○rec

name: throttleDemand

name: brakeDemand

value: 875.033413711548

type: brake %

origin: AutomotiveSystem-ssa1

time: 26/08/2014 4:44.08.420

navigation (6) ○rec

Figure 10: Data Model Inspector HUD

The record function is selectable for any topic. This records a time series data file of selected data model topic activity in the background, during simulation.

3.4 XML Configuration Tool

There are two critical processes required to perform configuration of a 3D simulation from the systems design environment (Figure.11):

- The systems design tool must be told what configuration options are available. The available options must be presented to the user to enable design time selections to be made.
- Once the user has selected the configuration options, these options must be presented to, and interpreted by, the 3D environment.

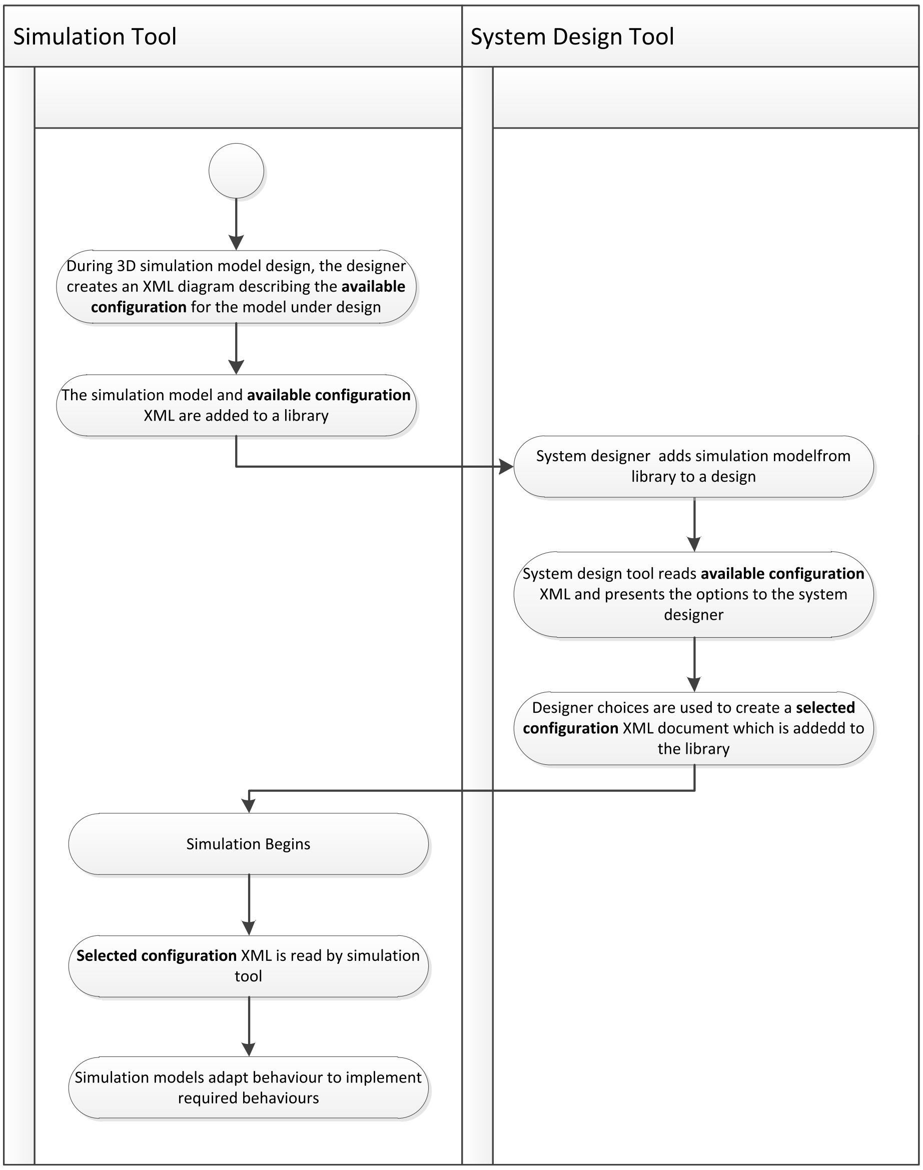

Simulation Tool
System Design Tool
During 3D simulation model design, the designer creates an XML diagram describing the available configuration for the model under design
The simulation model and available configuration XML are added to a library
System designer adds simulation modelfrom library to a design
System design tool reads available configuration XML and presents the options to the system designer
Designer choices are used to create a selected configuration XML document which is addedd to the library
Simulation Begins
Selected configuration XML is read by simulation tool
Simulation models adapt behaviour to implement required behaviours

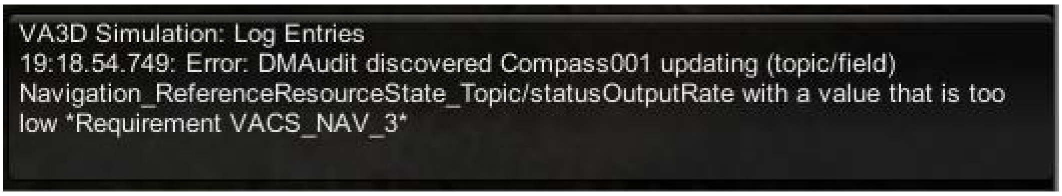
Figure.11: 3D Simulation Configuration Process

During 3D model design, the designer creates an XML document that lists all the available simulation model options. A library is used as a container for sets of simulation models and associated available option documents. The library is read by the systems design tool, which subsequently provides the appropriate design choices to the user when a model is selected.

Once the user choices have been made, the XML Configuration Tool is used to implement the interface between the simulation and the system design tools in terms of reading the selected configuration information and communicating the selections to the 3D models. The information is disseminated through the 3D object hierarchy in accordance with a message passing scheme. The appropriate simulation models consume the messages and adapt to implement the desired 3D form and behaviour.

3.5 Requirements Tracking Tool

The initial version of the requirement tracking tool allows the system designer to set limits for the legal values of a given data model topic field that relate to some systems design requirement. The simulation tool monitors the selected value and, if it exceeds the specified limits, the contravention is highlighted and the requirement that has been contravened is identified. This allows the user to provide limits for quantitative requirements that are automatically examined throughout simulation runtime. Figure 12 shows an example output of this capability, that shows a requirement ID (*VACS_NAV_3*) which is the responsible requirement identifier in the IBM® Rational® DOORS® document where it originated from before the system modelling analysis.



```
VA3D Simulation: Log Entries
19:18.54.749: Error: DMAudit discovered Compass001 updating (topic/field)
Navigation_ReferenceResourceState_Topic/statusOutputRate with a value that is too
low *Requirement VACS_NAV_3*
```

Figure 12: Requirement Tracking Tool - Catching and Reporting a Requirement Error.

4.0 CONCLUSION

The proposed methodology attempts to incorporate various different aspects in modelling for executing system designs before their handoff to real-world prototyping and development. These aspects are the methodologies currently considered as best-practices in MBSE, versus the interoperability requirements resulting from emerging defence standards. The study made use of elements collected during the synthesis and SysML formalisation of a system's design and aligned their communication needs with the data-centric interactions of a DDS network required for compliance with the UK GVA Defstan 23-009. The resulting architectural components are then modelled in a flexible 3D virtual prototyping platform which has the capability to iterate through changes in the system model design quickly and at very low cost. By taking advantage of the tools and the benefits of MBSE, it has been possible to annotate system requirements throughout the design stages from their origin all the way to their behavioural simulation.

7.0 REFERENCES

[1] IBM® Corporation, "Systems Engineering Best Practices Deskbook Release 4.1: Model-Based Systems Engineering with Rational Rhapsody and Rational Harmony for Systems Engineering", February 2014.

[2] Object Management Group® "OMG Systems Modelling Language (OMG SysML™) Version 1.4" September 2015.

[3] UK Ministry of Defence, Defence Standard 23-009 Generic Vehicle Architecture (GVA), Issue 4, October 2016

[4] NATO Standardisation Office, AEP-4754 NATO Generic Vehicle Architecture (NGVA), Under ratification, March 2016

[5] IBM® DeveloperWorks Technical Library, Case study of model-based systems engineering (MBSE): Part 2. Develop data-focused processes for distributed systems analysis and design, Mohit Choudhary, http://www.ibm.com/developerworks/rational/library/centralized-systems-model-rational-harmony-2/, (Obtained March 2016)

[6] Debbabi, M., Hassaïne, F., Jarraya, Y., Soeanu, A., Alawneh, L. “Verification and Validation in Systems Engineering - Assessing UML/SysML Design Models” Springer, 2010