MASTERS THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF **MASTER OF SCIENCE**

TITLE: Application of Model Based System Engineering (MBSE) Principles to an Automotive Driveline Sub-System Architecture

PRESENTED BY: Robert Kraus, George Papaioannou and Arun Sivan

ACCEPTED BY:

Advisor, *Michael Vinarcik*	**Date**
Advisor, *Dr. Jonathan Weaver*	**Date**
Department Chairperson, *Dr. Darrell Kleinke*	**Date**

APPROVAL:

Dean, *Dr. Gary Kuleck* **College of Engineering and Science**	**Date**

Disclaimer

This thesis is submitted as partial fulfillment of the graduation requirements of University of Detroit Mercy to obtain a Master of Science in Product Development degree.

The conclusions and opinions expressed in this thesis are those of the authors and do not necessarily represent the positions of the University of Detroit Mercy or the Ford Motor Company.

Preface

This thesis represents the culmination of our academic work at the University of Detroit Mercy in combination with our many years of work experience at the Ford Motor Company. Although this thesis is the result of our personal effort, we would like to acknowledge and extend our sincere gratitude to the following people for their valuable time and assistance, without whom the completion of this thesis would have been impossible:

Mr. Michael Vinarcik, Adjunct Professor, University of Detroit Mercy for his guidance, instruction and patience teaching MBSE, SysML and the MagicDraw modeling tool.

Dr. Jonathan Weaver, Professor of Mechanical Engineering, University of Detroit Mercy for his guidance, encouragement of innovation and teachings on how to find and solve important problems others may overlook.

Mr. Charles Krysztof, Senior Engineer, Ford Motor Company for his expertise on driveline sizing and torque transfer physics.

Mr. Michael Carter, Axle Systems Technical Specialist, Ford Motor Company for his expertise on driveline system boundary diagrams and design rules.

No Magic, Incorporated for providing student evaluation licenses for MagicDraw 18.2 with SysML plug-in for model based systems engineering. For additional information on MagicDraw, e-mail sales@nomagic.com.

Table of Contents

List of Figures

List of Tables

Abstract

This paper documents the real-world application of model-based systems engineering (MBSE) methodology and principles to an automotive driveline system. A system model developed in the course of this thesis supports the definition and sizing of the driveline system to best communicate and deliver functional specifications tied to customer requirements. Furthermore, this system model documents, organizes, manages and provides traceability for all customer, system, sub-system and component level requirements and is a useful tool supporting the system architecting and specification cascade process.

To support the project our team investigated and analyzed existing driveline component sizing tools and developed a comprehensive list of key assumptions and system design requirements which have been linked through a comprehensive SysML model. The key project deliverables are the SysML system model and a list of lessons learned / best practices collected during the development process.

The project was selected based on identified needs in the Ford Transmission and Driveline Engineering product development process. This is believed to be the first application of model based systems engineering at Ford Motor Company. The system model was conceived and executed using No Magic, Inc.'s MagicDraw software modeling package adapted to SysML systems modeling.

1. System Engineering Concepts

Currently, automotive driveline systems engineering is accomplished using a *document based* approach, where the system requirements and specifications are controlled and communicated through paper or electronic documents. They may also be resident in online databases or other storage media without rigorous traceability. Typically, complex system specifications result in reams of paper and electronic documents that are often incomplete or conflicting. The objective of this paper was to develop a working SysML model to replace the document based approach for a general driveline system as a proof of concept of the application of Model Based System Engineering (MBSE) and SysML to a mechanical system. The foundation for modeling any complex system is an understanding of basic system engineering fundamentals.

The System Engineering V-Model

The V-model is a graphical representation of a typical product engineering lifecycle. It was first developed in the 1980's and has been refined and applied in many different industries (Ryen 2008). *Figure 1-1, Driveline System Engineering V-Model* illustrates a typical V-model diagram for the automotive driveline system. The V-model divides the product development process into two halves.

The left side of the V deals with decomposition and definition. In this portion the top level system requirements are decomposed down to the sub-system level. The sub-system requirements are then decomposed into the component level

requirements, which are further cascaded to the sub-component level. At each level the requirements are defined and refined to the point of becoming design specifications.

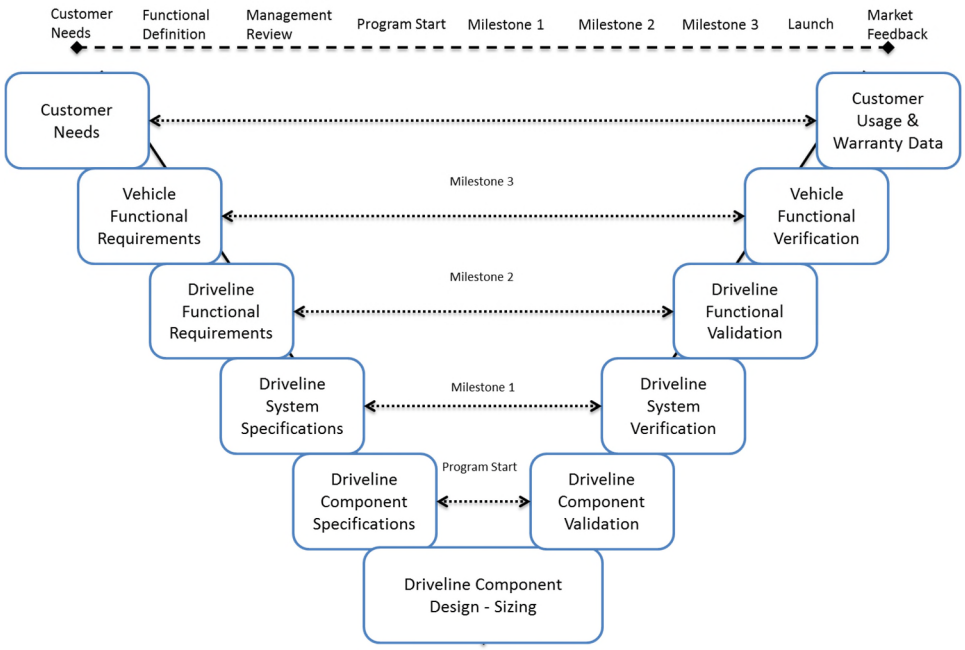


Figure 1-1, Driveline System Engineering V-Model

The right side of the V deals with integration and reconstitution. The designs developed at the component and sub-component levels are assembled, integrated and tested to ensure conformance to the original requirements. In this step the engineer's intent is to validate complete system performance. Each level of the V has a specific validation plan. Performance to requirements is verified at the system, sub-system, component, and, on occasion, even sub-component levels.

At the top system level the vehicle program and customer needs are received, analyzed, and transformed into functional requirements that define what the overall driveline system will do. At the top level the requirements define the

what, but not the *how*. This is the ***functional definition*** of the system and it describes the ideal behavior of the system without any bias toward a specific implementation.

At the sub-system level, a logical architecture is created based on the functional definition. In SysML, the logical architecture represents an intermediate abstraction between the functional and physical abstraction. The components within the logical architecture represent abstractions of physical solutions that deliver the functional operations (Pearce and Friedenthal 2013). Consider the rear axle from a heavy duty pickup truck and a sports car. They are physically very different, but they share the same basic functions. The ***logical structure*** is a high-level design that defines the overall framework for the driveline system and how the various sub-systems interact. The major sub-systems are then identified and defined. These sub-systems are further decomposed into components and sub-components. The system requirements are allocated appropriately to each sub-group at each level. Once the requirements are cascaded, the interfaces between the various sub-systems are defined.

In the traditional document based approach to system engineering, specifications are created at each level of the V. These specifications are managed independently by the various component engineers. Communication and coordination can be difficult across the various sub-system boundaries. In many cases these boundaries represent different suppliers, and in some cases direct competitors. In the driveline world, a driveshaft supplier like Neapco may not want to directly share their DFMEA (Design Failure Mode and Effects

Analysis) or DVP&R (Design Verification Plan and Report) with Dana, the axle supplier, if it contains proprietary information. This results in communication errors when specifications change or when requirements are generated at the sub-system or sub-component level. As an example, assume that the top level system engineer specifies that a system will *"sense driveshaft speed."* Who does this requirement apply to? Textually, it might be incorrectly applied to the driveshaft. The requirement is correctly applied to the rear axle, but the signal is used by the driveline controller. Even if the rear axle supplier includes a speed sensor, it may not have the resolution required by the controller. In this manner the system specification grows and grows and all communication between the axle and controller supplier must be handled by the system engineer. These communications disconnects result in system design errors.

System Boundary Definition

Incorrectly defined boundaries often drive mistaken assumptions and miscalculations, thereby resulting in system failures. To prevent such failures, it is critical to precisely identify the system's boundaries and the system's interactions with the external environment. This can be a deceptively simple task for the driveline, which is one of the more misunderstood systems in a vehicle.

A context diagram is a graphical tool for representing a system's interactions with the external environment. *Figure 1-2, Context Diagram for Driveline System* illustrates the context diagram for a typical driveline system. Also known as a black box diagram, it is characterized by a simple system representation in

the middle with minimal detail surrounding it. A black box definition should not assume or expose the internal structure, behavior or properties of a system. This approach delineates the definition of the *specification* (what the system is expected to do) from any particular *solution* (how the system could be implemented) (Pearce and Friedenthal 2013).

Figure 1-2, Context Diagram for Driveline System

Even this simplified driveline context diagram illustrates how many external factors come into play when analyzing a complex system. Considerations must be made for effects such as heat transfer, external contamination, interfacing components, sub-system requirements and customer requirements. Even the service technician must be considered when defining the system context. The driveline system boundary is subjectively interpreted when one starts defining inputs and environmental contributors. It is also this segregation of entities and

interactions which help to form and define the main substance for the system's requirements.

System P-Diagram

The system context can be further expanded and refined to include noise factors, inputs, outputs and the resulting illustration is known as a P-diagram. *Figure 1-3, P-Diagram for Driveline System* shows the input signals, noise factors and control factors going into the primary function, which is the transfer of torque from the output shaft of the transmission to the wheels (Carter 2015). The input signal here is the torque transmitted through the transmission from the engine as a result of driver input. The input signal also includes taking the loads from the suspension mounts and wheels.

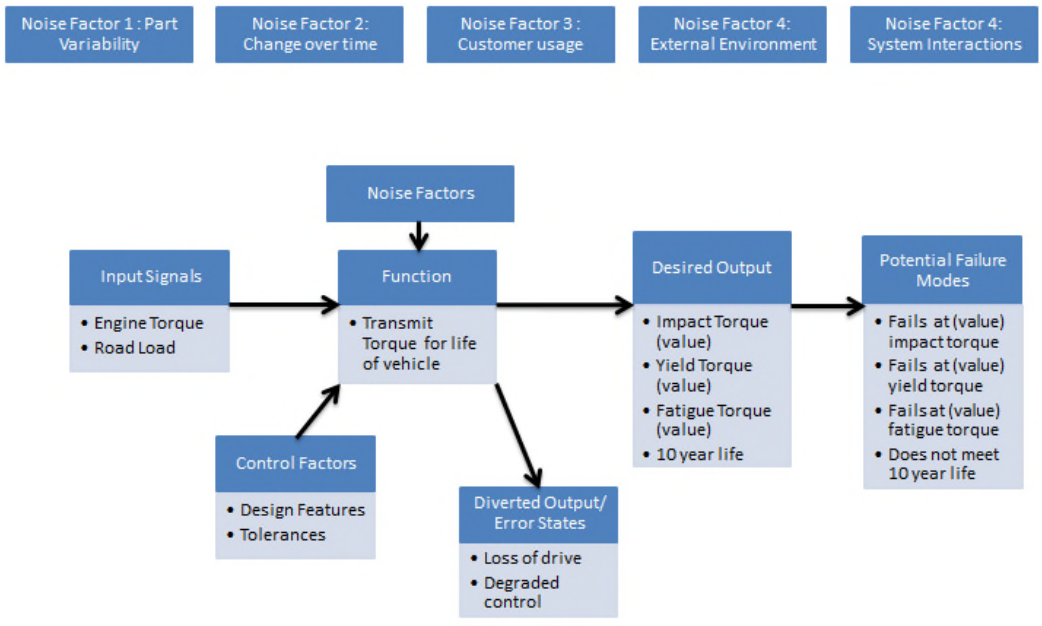


Figure 1-3, P-Diagram for Driveline System

The control factors are any features that aid in the intended function of torque transfer over the life of the vehicle and include items such as heat dissipation, gear ratio, spline size and bearing preload method. There are several noise factors that need to be considered as well, including piece to piece variation, degradation over time, customer usage, external environment and system interaction. All of these factors that feed into the function *block* of the driveline system appear in the context diagram, but are studied in detail to predict their contributions to desired output, derived outputs / error states and potential failure modes in the P-diagram.

The desired outputs shown in the driveline system engineering P-diagram are impact torque, yield torque and fatigue torque capability. Diverted outputs / error states can be locked axles, sudden or gradual loss of drive and degraded vehicle control. An example of potential failure-modes would be *failure of system at lower than target impact load* or *failure to meet fatigue life*.

2. Model Based Systems Engineering Concepts

The function of systems engineering is "*to guide the engineering of complex systems*" (Seymour and Biemer 2011). As discussed in Chapter 1, complex system specifications typically result in reams of paper and electronic documents that must be continuously managed and updated by the system engineer. Requirements can be generated or satisfied at the system, sub-system, component or sub-component levels with limited (or no) communication between work teams. Changes are not rigorously cascaded up or down most document based organizations, which often results in conflicting or missed requirements. Outside of functional testing, there is rarely any method to ensure all requirements are satisfied or verified at the overall system level.

To resolve this situation, this team proposes adopting a ***model based*** rather than document based approach for managing requirements. First, the requirements and specifications are decomposed using the logical structure of the systems engineering V-model. These requirements and specifications are then imported into a Model-Based Systems Engineering (MBSE) software package where they are tracked and verified throughout the project lifecycle.

A document based system specification can be thought of as a collection of textual requirements for the system (Pearce and Friedenthal 2013). Beyond a certain complexity level, a large number of specification requirements cannot be effectively managed. In the case of an automotive system, if the vehicle maximum speed is raised from 120 mph to 140 mph, it affects the specifications

for driveline shafts, bearings, seals, lubrication systems, heat rejection, NVH, driveline joints and system durability. The requirement change must be cascaded and verified at the system, sub-system, component and sub-component levels, which can easily represent more than 50 individual affected sub-components. Even if the requirement is captured and cascaded, there is no effective system to verify that the updated requirement is satisfied. Application of MBSE has the potential to solve this problem.

Language, Method and Tools

According to Lenny Delligatti, the three pillars of MBSE are the modeling language, the modeling method and the modeling tool (Delligatti 2013). The modeling language is any standardized medium for communication. The rules defined within the modeling language give ***unambiguous meaning to the model's elements and relationships***.

For our model, we have chosen to use Systems Modeling Language (SysML) to define the driveline system's structure, behavior, requirements and constraints. SysML allows engineers and designers to express and communicate the essential aspects of a complex system's functions, structure, behaviors and requirements in a concise logical model. Like any language, SysML has its own conventions for grammar and vocabulary. In a sense, SysML is spoken when system engineers use MBSE to communicate ideas about their systems to their coworkers and suppliers. While SysML is common, it is not the only modeling

language available to systems engineers and other engineering domains will use other languages more appropriate for their applications.

A modeling method is *"a documented set of design tasks that a modeling team performs to create a system model"* (Delligatti 2013). The method ensures that each team member is working to build the model consistently. In the absence of a documented and agreed modeling method there can be significant variance in the breadth, depth and fidelity of each engineer's contributions to the model. To ensure consistency, the project manager needs to clearly define the purpose of the modeling effort. What does the team hope to accomplish? What are the required inputs and outputs? The purpose defined will determine the level of detail required for defining the external environment and functional decomposition of the system. Effectively, the modeling method is closely tied to project scope. Our team was able to work close together to maintain a consistent modeling method through the course of the project, but on a larger team this could quickly become a concern.

Finally, the team must select the modeling tool. The modeling tool is the software package used to define and manage the system model. While ***the modeling language is "vendor neutral," the choice of modeling tool is not***. There are many commercial grade modeling tools available, in the same manner that there are many CAD or CAE tools. A selection of available tools is listed in *Table 2-1, Software Modeling Tools*.

There are many considerations in selecting a modeling tool; such as ease of use, compatibility with existing systems, security or product support. But these concerns are outside the scope of this paper. For this project we chose to use MagicDraw by No Magic with the SysML Plug-In. MagicDraw is commercially available modeling software that is geared towards enterprise implementations. It is most often used for software development.

Table 2-1, Software Modeling Tools

Software Package	Creator / Publisher	License
Agilan	Visual Paradigm	Commercial
Artisan Studio	Atego	Commercial
Enterprise Architect	Sparx Systems	Commercial
Cameo Systems Modeler	No Magic	Commercial
Rhapsody	IBM Rational	Commercial
UModel	Altova	Commercial
Modelio	Modeliosoft	Open Source
Papyrus	Atos Origin	Open Source
SysML Solution	Concept Draw	Commercial
Lattix Architect	Lattix	Commercial
Software Ideas Modeler	Dusan Rodina	Open Source
SysML Designer	ObeoNetwork	Open Source
SCADE System	Esterel Technologies	Commercial

MagicDraw supports UML, SysML, BPMN, UPDM, and other modeling languages. MagicDraw's native file format uses XML Metadata Interchange (XMI) so models can be exported to other applications (No Magic Inc. 2015a). We selected this software due to existing familiarity, but by no means was it the only appropriate solution.

MBSE and SysML were built directly upon the discipline of software engineering and architecture. SysML was based on the actual standard for software engineering, the Unified Modeling Language (UML). Software architecture represents software elements realizing the functional aspects of a product. SysML performs this same function for electrical, hydraulic and mechanical systems using the same intellectual tools. UML was developed within the Object Management Group (OMG) consortium (Balmelli 2007). XMI is an Object Management Group (OMG) standard for exchanging metadata information via Extensible Markup Language (XML). It can be used for any metadata whose metamodel can be expressed in Meta-Object Facility (MOF) (No Magic Inc. 2015a).

Functional Architecture

Functions define what a system does. They define what actions or activities must be accomplished or completed to achieve a desired outcome. They are ***behaviors***. Any system can be modeled based on its functions. In the SysML language, we elected to represent functions as ***operations***. An *operation* is a *property* of a *block*, where a block is an abstract representation of any part of a system. A block can be a physical entity, in the case of hardware, or abstract, in the case of a control signal or idea. Functions are linked through logical relationships to the various sub-systems and components. Each level of the model provides an increasing or decreasing level of specificity or abstraction.

The high level functional decomposition of an all-wheel-drive driveline is illustrated in *Figure 2-1, SysML Driveline Functional Decomposition*. The primary functions identified in this diagram are *Direct Torque Front / Rear*, *Transmit Torque, Disconnect Secondary Driveline, Multiply Torque,* and *Direct Torque Left / Right.* These functions are defined as ***operations*** in SysML and are properties of a function-stereotyped block.

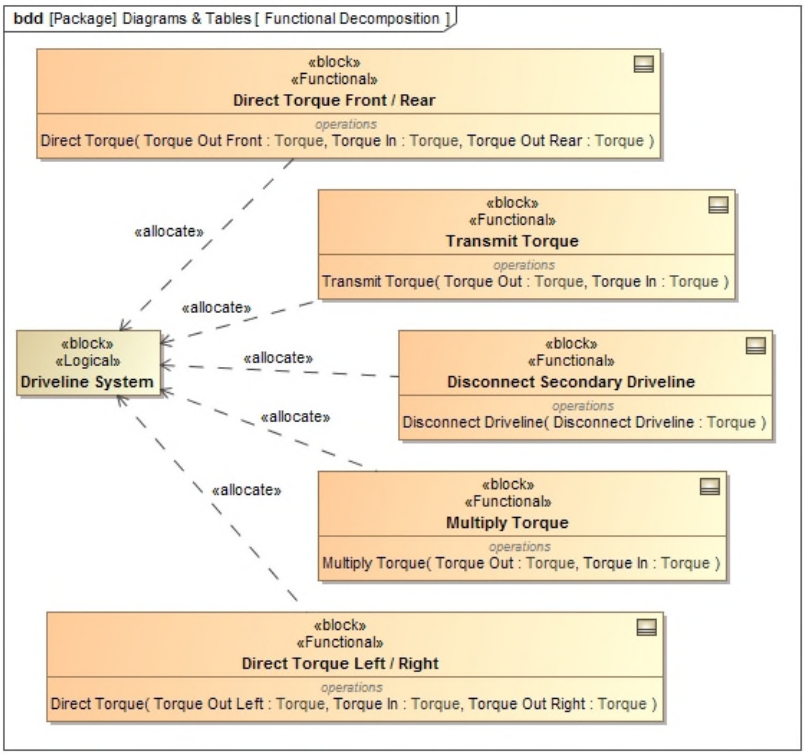


Figure 2-1, SysML Driveline Functional Decomposition

In this model the functions are ***related*** to the driveline sub-system through the ***allocate*** relationship (dashed arrow head). A ***behavioral allocation*** relationship refers to the activity of associating a function with a structural element. In SysML, this relationship tells the reader that all instances of the

receiving block can perform the behavior (Delligatti 2013). Each affected sub-system or component can then inherit the operation from the function-stereotype block according to the relationships defined by the system engineer.

In the case of the *Axle* sub-system, it has inherited the operations *Direct Torque Front / Rear*, *Multiply Torque* and *Transmit Torque*. Any functional requirements associated with these three primary functions will be cascaded down to the axle sub-system. If the system engineer later associates *Maximum Impact Torque* with the *Transmit Torque* operation, the *Max Impact Torque* value will automatically be cascaded to the *Axle* sub-system as a functional requirement. In MBSE, *Max Impact Torque* need only be updated in one location and ***any change to the requirement is automatically updated throughout the entire model***.

System Requirements

Requirements define customer and stakeholder needs in technical terms. They describe what characteristics and activities the system shall satisfy. In SysML, system requirement statements are also defined as objects. The requirement element (the object) contains the requirement text and a unique identifier. The requirement is then linked to a corresponding feature in the logical definition of the system. The *type* of requirement defines the type of feature the requirement can be associated with. *Table 2-2, Available SysML Requirement Types* provides a list of standard requirement types in SysML.

Table 2-2, Available SysML Requirement Types

Requirement Type	General Description	Example
Functional Requirement	Specifies a behavior of the system	Must transmit torque from transmission to wheels.
Performance Requirement	Specification, a quantifiable measurement of performance	Operational at vehicle top speed of 120 mph.
Interface Requirement	Specification for how system components connect	Must mount to transmission output flange PN FRZ102345.
Design Constraint	Design rule, or constraint on implementation	Threaded fastners must use common metric threads and standard hex sizes.
Physical Requirement	Physical constraints on the system	Must fit within underbody package envelope.
Usability Requirement	Constraint on usage by physical actors	Must allow clearance for 95th% hand to access control lever.
Business Requirement	Constraint related to business processes	System must be back compatible with existing service axle lubricants.

Functional requirements must be satisfied by *operations*, performance requirements must be linked with *value properties*, interface requirements must be linked to *proxy ports*, and so forth (No Magic Inc. 2015b). Refer to the SysML online documentation for a full explanation of acceptable relationships between requirement type and model objects. These restrictions provide rigor and clarity when defining the system.

In the case of vehicle maximum top speed, it would be associated with the top level abstraction *Driveline System*. The requirement is then *inherited* by all generalizations or constructions of the concept *Driveline System*. Once the link is created, SysML automatically associates the vehicle maximum top speed with the appropriate sub-systems, such as the driveshaft and rear axle, and by extension their associated components and sub-components. Thanks to this feature, duplication of requirements across sub-systems is unnecessary and easily

avoided during model construction. ***Each requirement is created and maintained in only one location.***

In the example of the Driveline System V, top level requirements would be impact torque or fatigue torque. As the system is decomposed, additional requirements will be created or documented. In the case of the *measure driveshaft speed* requirement, the user of the signal, the control module, would create a requirement specifying the signal. That requirement would have a relationship with the affected sub-system, axle, which would then inherit the *measure driveshaft speed* requirement created by the control module team. In SysML, it is easy to confirm that all requirements are mapped, or related, to an object in the model such as a block, interface or operation. A single driveline implementation can have as many as 800 separate requirements that the system must manage and satisfy.

In SysML, the generalization relationship described earlier is used to *inherit* functions and properties. Logical architectural elements specialize functional architectural elements; they then inherit operations and other properties such as requirements. This can greatly reduce the number of relationships that must be managed. For example, if 100 driveline system-level design requirements are related to the rear axle, and the rear axle has 100 components, 10,000 individual requirement relationships are required to cascade the driveline requirements to the component level. But using the generalization relationship, each requirement must only be connected once to the *Rear Axle* logical block, and the requirement will be inherited by each individual component.

Functionally, design constraint requirements are often developed by subject matter experts as *design rules*. These requirements are normally created and managed at the sub-system or component level. If a requirement or specification is developed by the Driveshaft Technical Specialist, it is stored in the *requirements for driveshafts* document. But what if the specification is actually applicable to all mechanical components, such as *grease sealing*? Will it be duplicated for transfer cases and axles? Do we end up with conflicting design requirements? The document-based system is heavily dependent on good communication across design teams to ensure that the requirements derived for the driveshaft are consistent with other mechanical systems. Furthermore, it is only effective if the requirement is read, understood and applied correctly when the next driveline system design is developed.

Most requirements are managed at the component or sub-system level without being tied back to the system as a whole. High level system requirements, such as vehicle maximum speed, often will not be cascaded through to the lowest level component directly, and as a result there is the potential to miss vehicle level or system level requirements during the system engineering exercise. Inheritance of requirements through generalization relationships and the ***satisfy*** relationship in SysML prevent these errors.

Finally, to ensure system compliance, every requirement must be verified by one or more ***test cases***. Test cases are physical check points, such as design reviews or validation tests, to ensure the design proposal meets the system requirement or specification. If the driveline system must have disconnect

capability, the engineer must check that the system bill of material contains a disconnect device. If the system is advertised to be capable of operation at -40 degrees Celsius, there should be a cold weather test procedure to validate operation at -40 degrees Celsius. Each of these instances is a *test case* that is associated with one or more requirements via the <<*verify*>> relationship.

Logical Architecture

A logical architecture defines how a system will be implemented. It abstractly defines a technical solution based on the system's required sub-systems, components and their interrelations according to the functions, technical requirements and customer needs defined earlier. A logical architecture should only be created after the system's functions and requirements are clearly defined. It does not define any particular system implementation, but rather the general guidelines of the eventual solution.

The object of creating the logical architecture is to parse out the system functions in a logical manner that can be implemented in the final engineering solution. Defining the sub-systems defines the interfaces. Linking the top level functions with the applicable sub-system later establishes the inheritance of the functional and performance requirements.

Figure 2-2, SysML Driveline Logical Decomposition illustrates the logical architecture of the IRS driveline system. The diagram chunks the driveline system into its component blocks, such as the axle, driveshaft and halfshafts. All blocks are of the stereotype <<*logical*>>, indicating that these are abstract,

rather than physical, entities. The component blocks list the parts, values and operations that have been defined or inherited from higher level abstractions. A chevron (^) character preceding the object name indicate that it was inherited from a higher-level element in the architecture.

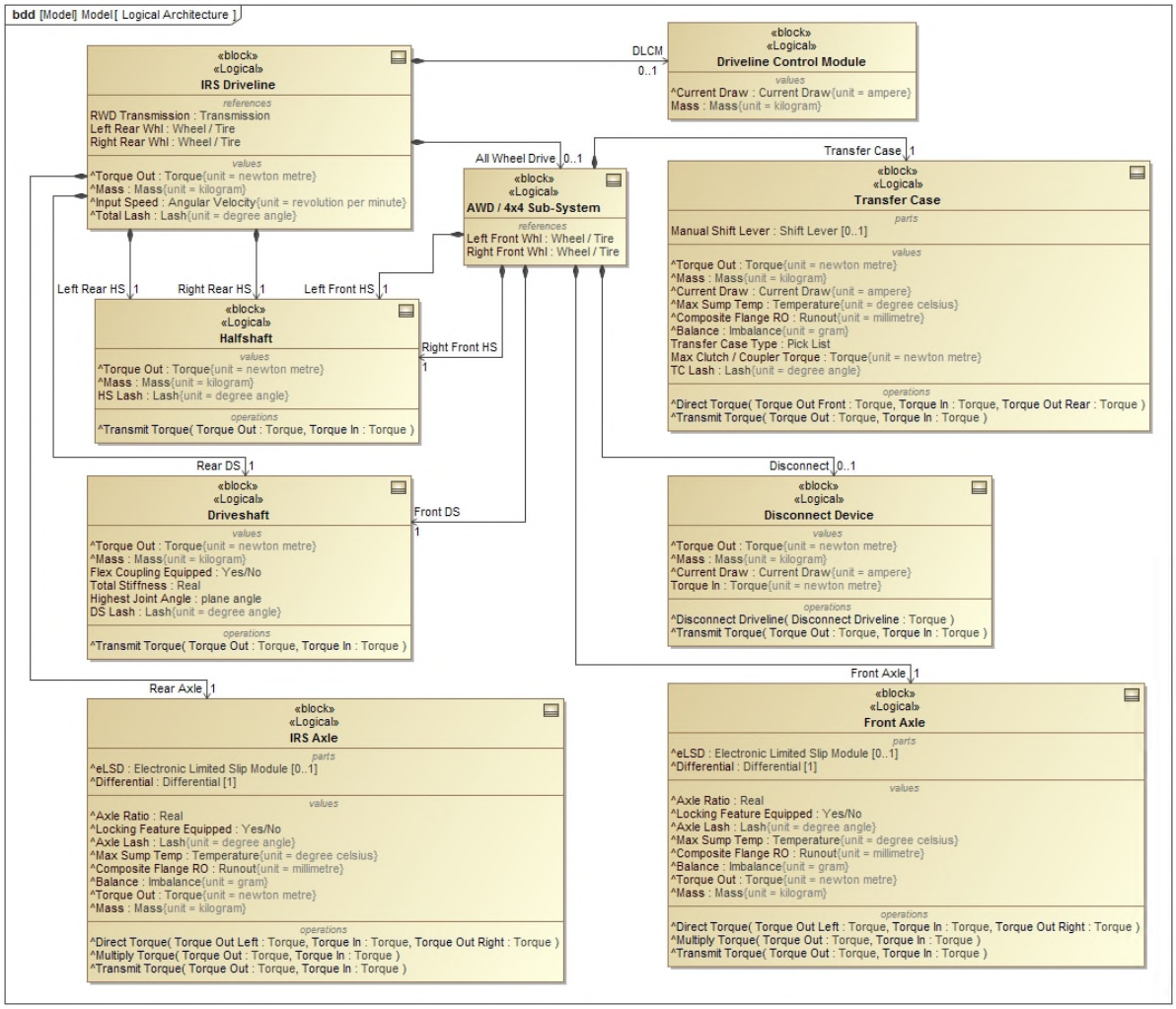


Figure 2-2, SysML Driveline Logical Decomposition

The diagram also defines optional constructions using the ***multiplicity*** relationship. ***A multiplicity defines how many entities are involved in each relationship***. For instance, the model defines the components included in the

AWD / 4x4 sub-system but also indicates that the entire subsystem is optional with the multiplicity call out *All Wheel Drive, 0..1*. This indicates that the driveline system can contain zero or one AWD sub-system. In plain terms, it indicates that our customer can purchase this particular vehicle with or without all-wheel-drive.

Physical Architecture

The physical architecture defines an actual instance of the real world product. As an example, a physical architecture would have a specific bill of material with component part numbers and CAD data. It is the lowest level of abstraction in a MBSE installation. At the physical level it is possible to visualize the actual system implementation. All of the component variable properties of the logical model become fixed physical properties in the physical model.

In the instance of our IRS driveline, the physical architecture could be the driveline system for the 2015 Mustang. It would include a two-piece aluminum driveshaft, 8.8” aluminum rear axle with mechanical limited slip and two halfshafts. Since the Mustang has no AWD option, the AWD subsystem multiplicity in our logical model would be ‘0.’ Each of these components has a physical part number and defines a specific hardware instance of our functional and logical models.

3. Driveline Definitions and Concepts

An automobile driveline is the system of components linking the transmission output to the drive wheels. By definition, the driveline explicitly ***does not include the engine or transmission***. The driveline's primary function is to transmit drive torque from the powertrain (engine and transmission) to the ground (wheels). Secondary functions include reacting torque from the brakes, carrying chassis loads, and transmitting torque from the wheels to the powertrain during engine braking events. An automobile's driveline system can be as simple as two shafts connecting a conventional geared transaxle to the front wheels, or as complex as an electro-hydraulically controlled torque vectoring system which distributes torque to all four wheels via computer control.

Driveline Architecture

There are many different layouts and architectures that accomplish the primary function of torque transfer. Most passenger cars are *front-wheel-drive* to reduce weight and cost through a shared casting that incorporates both the transmission and differential. Sports cars and luxury vehicles are often *rear-wheel-drive* for handling and performance, through better weight and power distribution. Trucks and SUV's often have 4x4 systems for off-road capability, or optional all-wheel-drive systems for improved traction and stability. *Figure 3-1, Typical IRS AWD Driveline* illustrates a typical independent rear suspension all-wheel-drive architecture, such as the one found in the BMW X5 or Dodge Durango.

Figure 3-1, Typical IRS AWD Driveline

Within the SysML modeling language, all of these variants are treated as ***generalizations***. Generalizations represent specific subtypes of driveline systems. The generalization relationship conveys inheritance between any two elements. The more general element is the *supertype* while the more specified instances are all *subtypes*. The generalization relationship is intended to be *one to many*. In the case of *Figure 3-2, BDD of Common Driveline Systems* the more generalized element is *Driveline System*, and the more specialized elements are *Longitudinal Driveline System* (rear-wheel-drive) and *Transverse Driveline System* (front-wheel-drive). ***In SysML, an open arrowhead indicates a generalization relationship.*** The direction of the arrow is from the more specific instance to the more general type. The *filled diamond* arrowhead in the diagram indicates a *part* relationship. In this diagram, the *Differential* is a part of the *Axle* sub-system. The arrow points from the part to the up-level assembly.

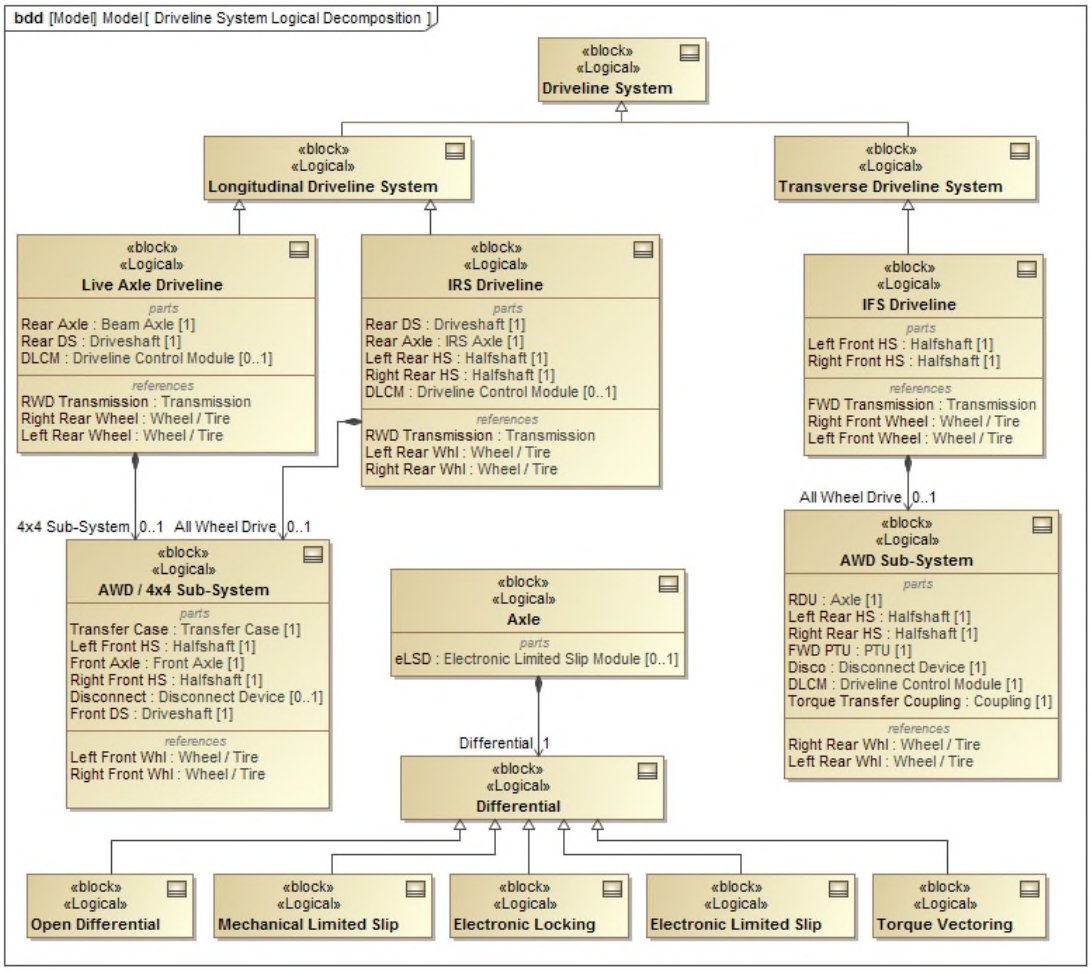


Figure 3-2, BDD of Common Driveline Systems

The selection of the basic driveline system architecture is driven by many factors, including drivability, off road performance, track performance, towing capacity, vehicle package, fuel economy, weight, investment, cost, complexity or customer demand.

Driveline Components

A typical 4x4 or all-wheel-drive (AWD) driveline consists of a transfer case, front and rear driveshafts, front and rear axles, a disconnect device and four

shafts connecting the axles to the wheels. *Figure 3-3, Isometric View of IRS AWD* Driveline illustrates the relationship and connections between the various components for an independent rear suspension (IRS) driveline system. In pick-up truck applications the independent rear suspension is replaced by a rear beam axle to carry the vehicle payload.

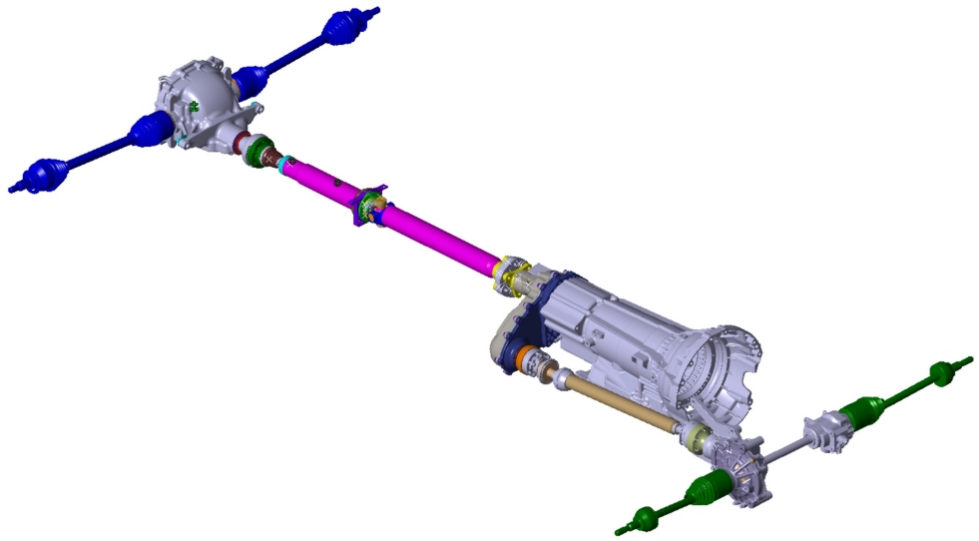


Figure 3-3, Isometric View of IRS AWD Driveline

The function of a transfer case is to split the torque from the transmission and direct it to the front and rear wheels. An AWD transfer case model is illustrated in *Figure 3-4, Transfer Case*. There are many types of transfer cases, from the rather simple manual control 4x4 cases found on heavy duty pick-up trucks; to complex computer controlled automatic cases that direct torque based on driver demand and/or vehicle dynamics. In all cases, the intent of the transfer case is to direct torque to the wheels with the most traction. In front-wheel-drive

applications the PTU replaces the function of the transfer case. In the SysML language the PTU is an instance of the generalization *Transfer Case*.

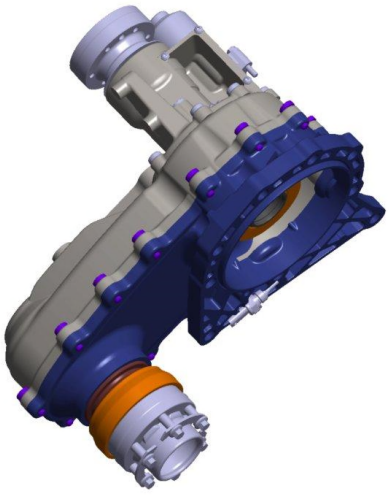


Figure 3-4, Transfer Case

The function of the front and rear axle is to multiply the torque from the driveshaft and direct it to the left and right wheels. The torque multiplication is fixed and is based on the *axle ratio*. An axle ratio represents the relationship between the number of driveshaft revolutions to the number of wheel rotations and is defined by the number of teeth on the rear axle ring and pinion gears. According to Chevrolet communications manager Tom Wilkinson, the axle ratio for Chevrolet pickup trucks is "*chosen to balance performance, capability and fuel economy,*"(Kurt Niebuhr 2014). Lower numerical ratios are biased toward fuel economy, while higher ratios provide additional towing capability. On a vehicle equipped with 4x4 or AWD, the front and rear axle ratios must be the same. *Figure 3-5, Rear Axle* highlights the rear axle geometry from the isometric IRS AWD driveline package.

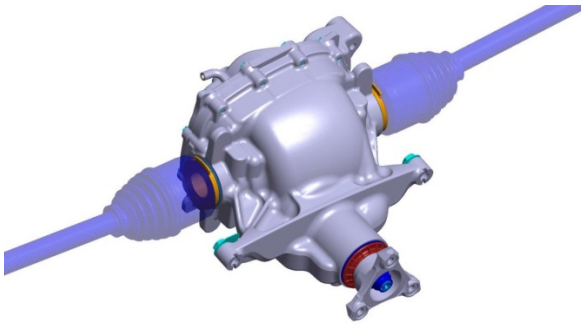


Figure 3-5, Rear Axle

Shaft components, such as driveshafts and halfshafts, are considered modular components of the driveline system. They are manufactured as tubes, solid shafts, welded tube shafts (WTS), swaged tubes, and monobloc tube shafts (MTS). Shaft design selection is based on strength, stiffness and fatigue durability requirements. Typically a driveshaft or halfshaft is designed as the weakest component for durability. *Figure 3-6, Rear Driveshaft* highlights the rear driveshaft in the AWD driveline model. This is a two-piece driveshaft with a center bearing that connects to the vehicle underbody. The interface between the shaft components and the mechanical components is typically a spline.

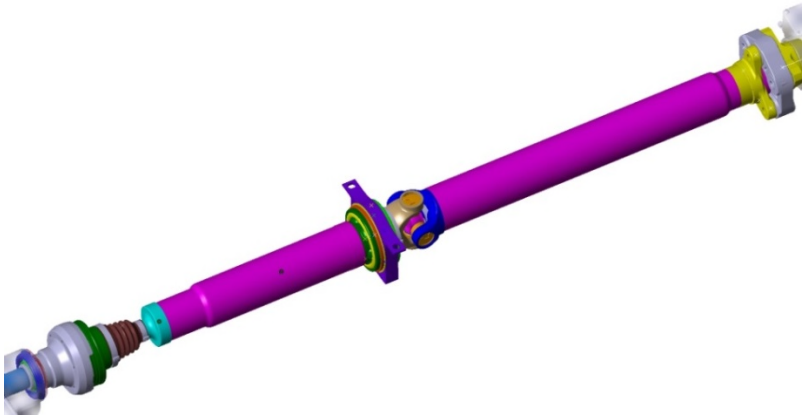


Figure 3-6, Rear Driveshaft

The primary functions of driveshafts and halfshafts are to transmit torque and to accommodate chassis and suspension deflection, such as changes due to payload, steering and vehicle cornering. Key design considerations are driveline angles, torsional impact and torsional fatigue. The front and rear halfshafts are highlighted in *Figure 3-7, Front and Rear Halfshafts*.

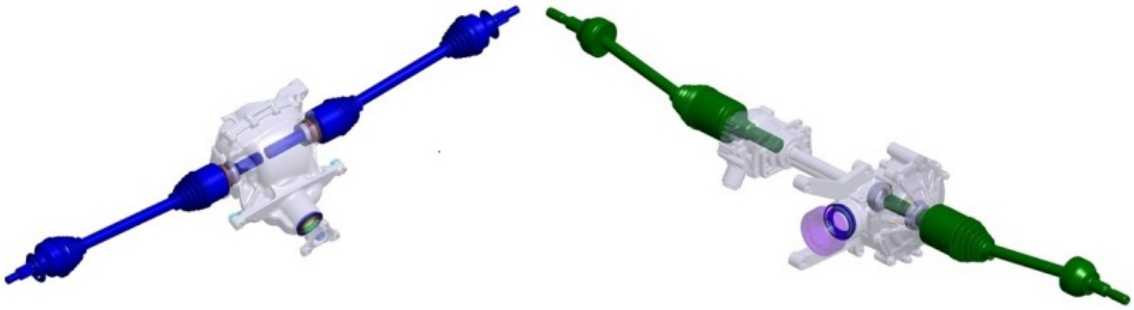


Figure 3-7, Front and Rear Halfshafts

The final primary component in the AWD driveline system is the disconnect device. The purpose of the disconnect device is to disconnect the front axle from the wheel ends. Along with the transfer case, the disconnect device stops the front axle and driveshaft from spinning when AWD or 4x4 function is not required. This reduces spin losses and improves fuel economy. A linkshaft based AWD disconnect device is shown in *Figure 3-8, Front Axle and Disconnect Device*, though there are alternative methods for decoupling, such as integrated wheel ends, locking hubs and front axle mounted / integrated devices.

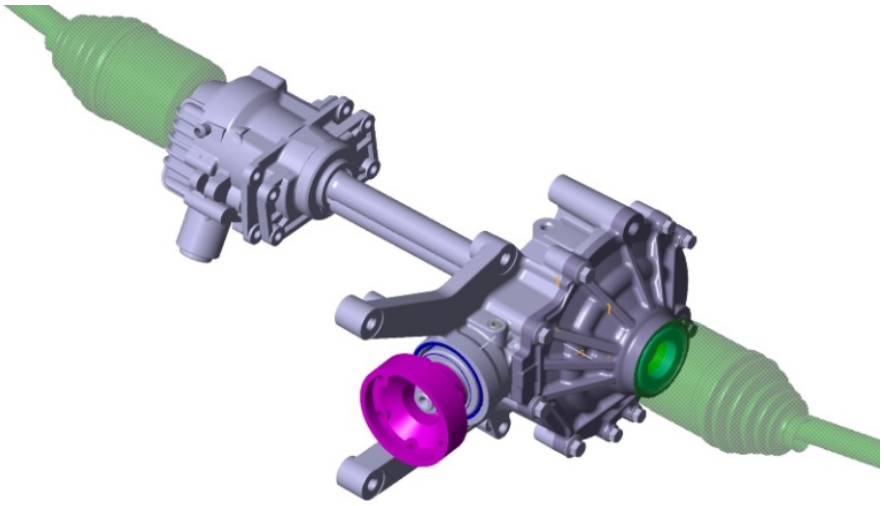


Figure 3-8, Front Axle and Disconnect Device

Joint Definitions and Overview

A key design consideration in driveline joint design is NVH, or *noise, vibration, harshness*. A customer's expectations for NVH refinement are often in direct contradiction to the vehicle's other design targets and attributes, like fuel efficiency, weight reduction and cost. A primary goal of the driveline system engineer is to develop a system that avoids known NVH phenomena. These phenomena can include but are not limited to: launch shudder, which occurs during a take-off launch and results from instability in the driveline; driveline boom, which is an objectionable low frequency boom noise caused by driveline resonances; and driveline whine, which is mainly caused by axle hypoid gear transmitted error (Wellmann, T., Govindswamy, K., Braun, E., and Wolff, K. 2007).

There are three common types of driveline joints in production today. The oldest and simplest is the universal joint or U-joint (UJ). It consists of two shaft

yokes positioned at right angles to each other and connected by a four point cross tying the yokes together. The cross rides inside four bearing cap assemblies which are pressed into the yoke eyes. This is illustrated in *Figure 3-9, Universal Joint Exploded View.* U-joints are commonly found in both front and rear drive shafts. The problem inherent in the basic design of the u-joint is that the angular velocities of the components vary over a single rotation. This can result in NVH concerns and limits their use to applications with low working angles.

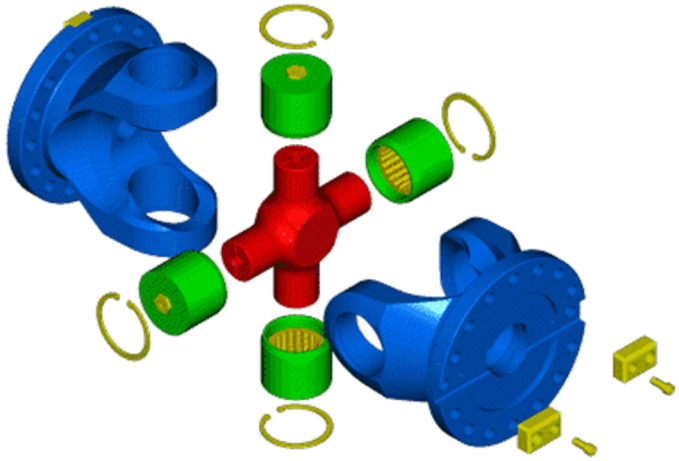


Figure 3-9, Universal Joint Exploded View

The second common joint is the constant velocity joint (CVJ). CVJ's allow a shaft to transmit torque through a variable angle with a constant rotational speed. This joint eliminates the variation in angular velocity inherent in U-joints, but with a minimal increase in rotational friction and end play. *Figure 3-10, Constant Velocity Joint* shows a typical fixed CVJ which is found on most halfshafts. CVJ's are typically more expensive than U-joints but allow much higher working angles, in addition to improved working properties.

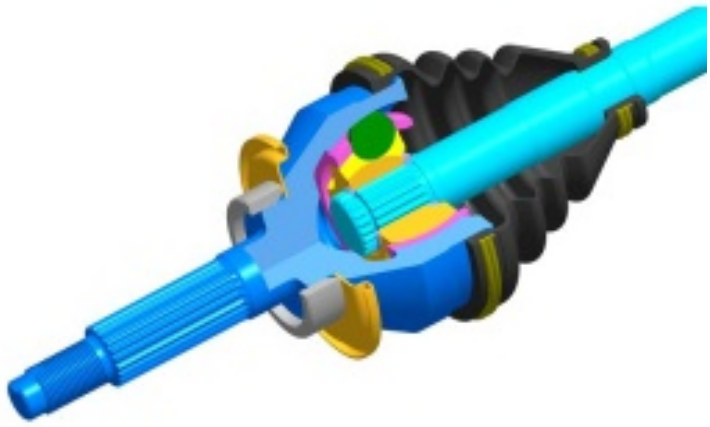


Figure 3-10, Constant Velocity Joint

The last commonly available driveshaft joint is the flexible coupling. Flexible couplings are typically used to decouple NVH concerns between the transmission and driveshaft / axle. *Figure 3-11, Flex Coupling Joint* illustrates a common type of driveshaft flexible coupling. The input and output shaft flanges are bolted to a disc made of rubber over nylon cord using alternating bolt positions. This ensures that each shaft section is not rigidly connected to one another, but instead through the rubber coupling. The elasticity of the rubber absorbs vibration and flexes for alignment. It follows that the rubber must withstand the vehicles maximum transmitted torque, for which reason the rubber is often reinforced internally using molded-in nylon or carbon fiber materials. It is a relatively expensive joint design and requires a minimal working angle.

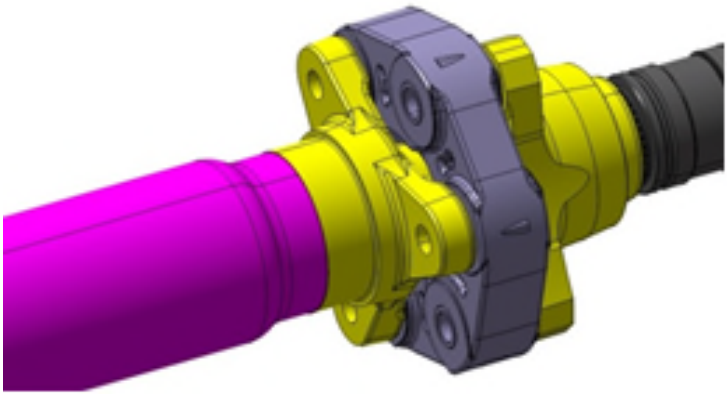


Figure 3-11, Flex Coupling Joint

Engineering joint selection considers the joint working angle, average torque transmitted, average speed and NVH requirements. Proper joint selection and sizing are typically key enablers for reducing overall driveline system cost and weight.

Sub-System Definition

For the purposes of this paper and the construction of the model, the all-wheel-drive (AWD) or 4x4 system is considered to be a sub-system of the driveline system. The AWD sub-system connects to the transmission output (rear-wheel-drive or front-wheel-drive) and splits the drive torque from the primary drive axle to the secondary drive axle. In the case of a front-wheel-drive architecture, torque is sent to the rear wheels. In the case of a rear-wheel-drive architecture, torque is sent to the front wheels.

Regardless of the vehicle architecture, additional components that are included in the AWD system are a transfer case / PTU (rear-wheel-drive versus front-wheel-drive), a secondary driveshaft, a secondary axle and typically some

type of disconnect device. In all cases other than a manual transfer case, torque transfer is accomplished through an electronic driveline controller.

4. Driveline Sizing

Commercial truck customers are very demanding. The more their trucks can haul or tow, the more work they can accomplish using fewer vehicles or repeat trips, and the more profitable they become. They demand durability, reliability, fuel efficiency and the lowest total cost of ownership. And they want it all delivered at a competitive price. Customer expectations, fierce competition and increasing governmental regulations are forcing automakers to rethink and improve their product development strategy. It has created an environment where technology, engineering tools, methodologies and manufacturing capabilities must be chosen appropriately. Automakers must target beyond their current limitations to achieve best in class product and customer satisfaction. Today's market demands creativity, innovation, precision, efficiency and most importantly, excellence of execution. Design optimization of each component, system and sub-system is the primary objective of any driveline sizing activity.

The Purpose of Driveline Sizing

In engineering, it's very tempting to over-specify or over-size components to achieve performance targets. Often, aggressive program milestones and budget constraints can force engineers to take liberties in their deliverables through minimal optimization exercises. This typically results in over-designing driveline components, adding cost and weight, reducing fuel economy and hurting the customer's bottom line. On the other hand, under-specifying results in early failures. In the short-term, usage related failures result in higher warranty repair

costs and reduced customer satisfaction, but the potential long term damage to the brand is more concerning.

Today, driveline components are generally designed and sized individually, often with insufficient input requirements, significant design carryover (the *"we've always done it this way"* approach) and little regard for system interactions. This can result in over-design and less efficiency for the overall system. With the increasing complexity of today's driveline systems, together with the competitive demands placed upon automakers toward improving on customer expectations, it has become evident that an advanced driveline sizing tool is not just an asset, but a necessity.

A sizing tool takes the inputs from the vehicle team and converts them into three basic outputs: ***impact torque, fatigue torque and yield torque***. These three basic outputs can vary along the driveline due to transfer case and axle ratios, but are required for each component. The vehicle driveline defined here is the total system connecting the transmission output to the vehicle's drive hubs and typically includes the driveshaft, axle and halfshafts. The driveline must be sized to include all potential powertrain configurations (multiple engine displacements, diesel, hybrid, manual / automatic transmission), drivetrain configurations (two wheel drive, four wheel drive), customer usage profiles (heavy towing, snow plow, passenger car, final gear ratio, wheel / tire sizing) and vehicle configurations (multiple wheelbases, track widths, cab heights, suspension displacement). A useful driveline sizing tool must tabulate output for each required instance.

Driveline Sizing Methodology

Defined at a high level, ***sizing of driveline components is the alignment of physical design attributes to the demands of rotational inertia and transmitted torque to meet customer expectation***. Individual component designs are typically handled at a systems and / or sub-systems level, while any torque considerations are individually balanced using a *combined torque* that addresses both impact and continuous fatigue torque. A critical first step in any driveline sizing activity is the gathering of comparative prior data, along with obtaining all vehicle level inputs to be used for analysis and sizing calculations. This comparative data is also useful in helping to determine input factors used in the torque and sizing formula calculations. Any legacy data must also be maintained as a large extensive database of sizing experience and proven benchmarking from existing, and / or previous designs. A basic context diagram for driveline sizing is shown in *Figure 4-1, Required Inputs and Context for Driveline Sizing,* which lists the high level inputs, as well as their black-box integration and flow through the system.

Prior to the development of comprehensive sizing tools, in house driveline engineering traditionally relied on input from up-stream suppliers to size each sub-system in series. Impact load and fatigue life calculations would be re-calculated at each level. Universal joint (UJ) & constant velocity joint (CVJ) durability life calculations were completed independently and were typically a 100% supplier dependent process. Additionally, there was a lack of any specific library data for vehicle life calculation & correlation studies, and no library of

stored vehicle Road Load Data (RLD) for rapid access to compare design options. Inefficiencies or miscalculations would often be cascaded throughout the system design process.

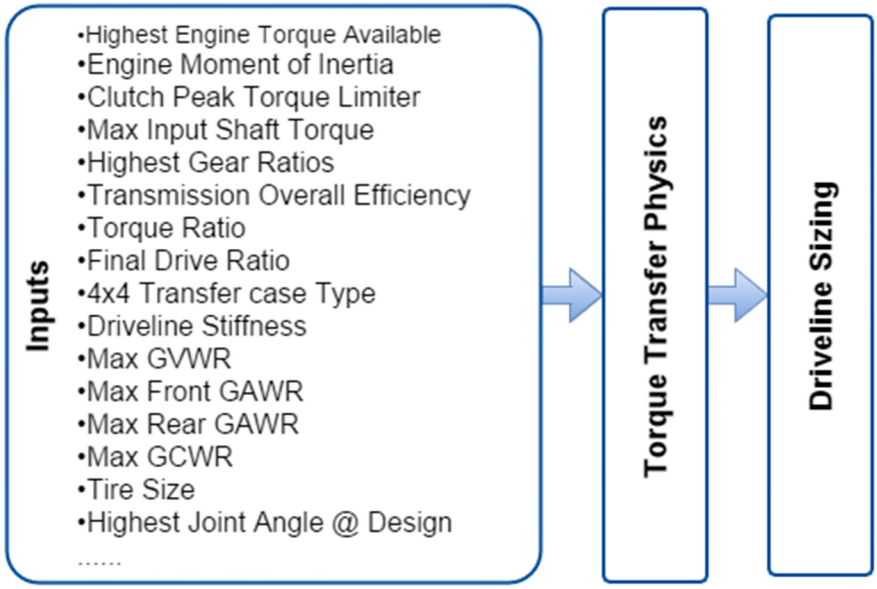


Figure 4-1, Required Inputs and Context for Driveline Sizing

With an identified need for improved engineering efficiency in driveline sizing, some organizations such as Ford Motor Company and Dana Corporation developed in-house Microsoft Excel based driveline sizing calculation tools to organize, integrate and automate the required inputs, torque transfer calculations and outputs. Although typically in spreadsheet format, a sizing tool could also be effective as reference files or a database based on benchmarking or industry standards.

Determining which specific components need to be sized, begins with selecting the best fit driveline configuration for a given vehicle segment. Will the vehicle be conceived as front-wheel-drive or rear-wheel-drive? Does

performance require four-wheel-drive? Does the customer segment require some type of traction control system? In addition to variations in driving dynamics, there are also significant differences in how an all-wheel drive (AWD) or four-wheel-drive (4x4) system reacts to a loss of traction. These decisions all influence the torque calculations for the system. Accurate sizing of components is not only critical to the successful engineering and service life of the vehicle, but is best established early in the development cycle, greatly assisting program flexibility by offering quick attribute changes when needed.

Sizing Tool Overview

The basic outputs of any driveline sizing tool are impact torque, fatigue torque and yield torque. *Figure 4-2, Driveline Sizing Flowchart* is an overview of the key inputs and logical calculations needed to generate the driveline impact torques, which are usually most critical for axle, driveshaft and halfshaft sizing.

The flow diagram represents the sizing calculation from a left to right progression, beginning with the input data on the left side. The torque transfer physics calculations and logical decisions are in the middle of the chart. Finally, the three critical output impact torque definitions are on the right side of the diagram. These outputs are used for sizing the specific sub-system components.

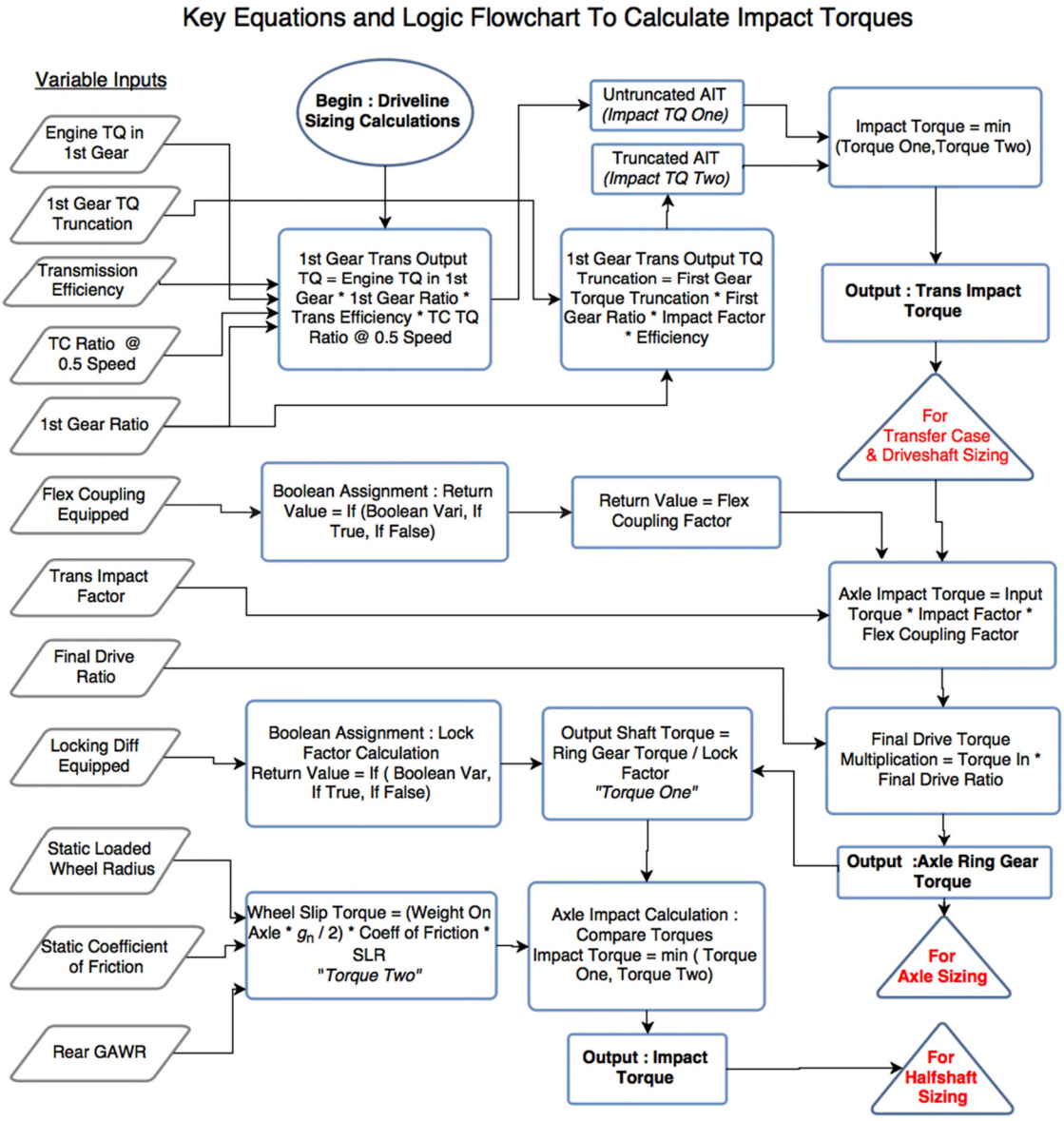


Figure 4-2, Driveline Sizing Flowchart

Torque Transfer Physics

Driveline sizing must be achieved only through a complete understanding of the physics behind torque transfer through the system and its effect on the driveline components. Engine torque is transmitted through the clutch (manual transmission applications) or the torque converter (automatic transmission

applications) to the transmission and through the driveline system to the wheels. This results in a propulsion force exerted on the vehicle which is limited by the traction at the wheels / tires. Wheel slip is the ultimate limit for the torque on the driveline and represents the maximum torque that can be transmitted through the driveline under any circumstance.

Torque can be defined as *impact* or *continuous* through vehicle life. Impact torque is the maximum seen by the driveline and is related to inertia loading. This torque is seen during a drag start, or pulling away from a stoplight with a very heavy trailer. Continuous torque is also known as *fatigue torque* and represents the average torque transmitted through the driveline during normal operation. Very high impact loads result in overload failures. High continuous torques result in fatigue, or wear out, failures. Regardless of configuration, proper driveline sizing considers both impact and continuous torque demands in combination to achieve a balanced design.

Power pack impact torque (*Tpp*) is the torque generated by the engine at the transmission output shaft. It occurs during wide open throttle or clutch slip conditions. The torque is an outcome of the inertia in the system and is higher than the maximum torque generated by the engine. The impact factor is a constant that is selected based on previous engineering experience and varies throughout the industry. The general formula for a rear wheel drive vehicle is:

$$Tpp = Engine\ Torque * First\ Gear\ Ratio * Transfer\ Efficiency * Impact\ Factor$$

Truncated transmission output torque (*TT*) is when torque must be reduced at the transmission due to physical hardware limitations within the system. When determined, the control system calls for reduced engine torque to the flywheel / flexplate, which in turn adjusts (lowers) torque at the transmission output. This is common in medium and heavy duty diesel trucks where the engine peak output torques can be very high. The general formula for a rear wheel drive vehicle is:

$$TT = First\ Gear\ Torque\ Limit * First\ Gear\ Ratio * Impact\ Factor * Efficiency$$

Axle ring gear impact torque (*Trg*) is torque seen by the rear axle due to axle ratio multiplication. The general formula for a rear wheel drive vehicle is:

$$Trg = Input\ Torque * Final\ Drive\ Ratio * Impact\ Factor * Flex\ Coupling\ Factor$$

Wheel slip torque (*TWS*) is the maximum torque that can be reacted by the wheel. It is the torque at which the wheel loses traction and begins to spin, thereby limiting the maximum torque that can be transmitted to the ground. It is dependent on the weight of the vehicle and the friction at the tire surface. It can be affected by vehicle dynamics. During drag start events a car's weight will shift and tend to squat, further increasing the downward force on the axle and tires. The general formula for a rear wheel drive vehicle is:

$$TWS = (Gross\ Vehicle\ Weight\ /\ 2) * Gravity\ Force * Surface\ Friction * Tire\ Radius\ /\ (Final\ Drive\ Ratio * Mechanical\ Efficiency)$$

These torque calculations apply to driveline impact torque. Similar calculations are available to determine fatigue torque and yield torque.

Required Inputs and Data

Determining and communicating the correct input data needed for the driveline sizing process is critical to ensure useful output. The question often asked of the driveline system engineer is *"What minimum information is needed to size the driveline?"* To avoid wasted time and effort, it is important for the system engineer to understand the minimum data required to complete the initial sizing torque calculation. Further output refinement is then implemented as an iterative process utilizing road load data (RLD) and legacy data from previous designs.

Typically the component engineer has the responsibility for sizing and delivering individual driveline components once the inputs and metrics data is sourced from PDL, powertrain line-ups, weight charts and tire drawings. A list of required sizing inputs is presented in *Table 4-1, Minimum Inputs Required for Driveline Sizing*. This represents the minimum data needed before starting a new driveline design, and is also useful for assessing any changes through the development process.

Table 4-1, Minimum Inputs Required for Driveline Sizing

Sizing Input	Typical	Notes
Vehicle Assumptions		
Driveline Configuration (Type)	4WD/4x4	What style of driveline is required?
Brake Traction Control Equipped	Yes	Brakes affect axle loads and tire slip.
Max GVWR	6000	How much can the vehicle weight?
Max Front GAWR	2800	How much load is on the front axle?
Max Rear GAWR	3200	How much load is on the rear axle?
Max GCWR	11000	How much can the vehicle tow?
Durability Road Load Data Set (RLD)	RLD	What is the durability requirement?
Engine		
Engine Torque in 1st Gear	280	Drag start torque.
Highest Engine Torque Available	400	Total max output torque.
Engine Torque in Rev. Gear	280	Max torque in reverse.
Engine Moment of Inertia	7.7	Inertia is related to impact load.
Transmission		
Transmission Type	Auto	Auto or manual transmission?
1st Gear Torque Truncation	350	Max torque out of transmission.
2nd Gear Torque Truncation	400	Max torque out of transmission.
Transmission Efficiency	0.93	Transmission losses.
Torque Converter Ratio @ Stall	2	Ratio determines start performance.
Torque Converter Ratio @ 0.5 Speed Ratio	1.6	Ratio determines start performance.
Transaxle FDR	3.7	Final drive ratio for FWD.
Flywheel Moment of Inertia - Manual Trans	6.8	Inertia is related to impact load.
Clutch Peak Torque Limiter - Manual Trans	100	Max torque out of transmission.
Number of Forward Gears	10	How many gears?
1st Gear Ratio	4.2	Torque multiplication.
2nd Gear Ratio	2.4	Torque multiplication.
Reverse Gear Ratio	3.4	Torque multiplication.
Driveline		
4x4 Transfer case type – Active/On Demand	Active	Determines torque bleed front / rear.
4x4 Transfer case Max Coupler Torque	1250	Max torque to front.
Flex Coupling Equipped	Yes	Attenuates impact.
Total Driveshaft Stiffness	600	Attenuates impact.
Highest Joint Angle @ Design	5	Joint sizing input.
Rear Axle Ratio	3.7	Final drive ratio for RWD.
Locking Differential Equipped	Yes	Increases halfshaft torque.
Wheels / Tires		
High Rotational Inertia Tire Equipped	Yes	Increases wheel slip impact.
Tire Size	265/55R18	Tire slip.
Max Tire SLR	13.75	Tire slip.
Tire Coefficient of Static Friction	0.92	Tire slip.

Beyond the minimum requirements, useful supplemental data can include:

Road Load Data (RLD), often referred to as durability data, is acquired through instrumented vehicles. To acquire RLD, a test vehicle is instrumented with load cells, accelerometers, strain gages, torque meters and force transducers

and run through a simulated durability course. The data is accumulated into a database and plotted as a torque histogram. Good RLD is typically not available until after the first prototype vehicles are built and tested, and surrogate data is often used for driveline sizing.

Supplier Joint Data is the background information provided by driveline suppliers and is often useful for selecting driveshaft and halfshaft components. Knowing available joint sizes, U-joint yoke geometries, shaft geometries and design and materials can be an invaluable pre-screening method when an engineer requires short production lead times or minimal investment.

Benchmarking Data is the library of competitive information used for establishing logical comparisons between initial design concepts and existing competitive designs. It is particularly useful for validating initial sizing calculations in setting cost or weight targets. Automotive manufacturers spend a significant amount of money and time performing accurate benchmarking studies to set appropriate targets.

5. Driveline Model Structure

"Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases." (Operations 2007)

According to the 2007 report of the International Council on Systems Engineering (INCOSE), model based systems engineering was very likely to replace the document-centric approach practiced by most systems engineers. By 2011 SysML was used by 20% of aircraft and defense companies and 7% of automotive manufacturers (Paredis and Davies 2011). It appears likely that MBSE will influence the future practice of automotive systems engineering once it is further integrated into the systems engineering product development process.

Functional Decomposition

The driveline system functional decomposition starts from the P-diagram. The team identified five basic behaviors, or operations, that a driveline system had to perform to meet the customer's basic needs. The system needed to 1) transmit torque, 2) direct torque left / right, 3) direct torque fore / aft, 4) multiply torque and 5) disconnect the secondary driveline. These basic functions are shown in *Figure 5-1, Functional Decomposition of Driveline System*. Each function is associated, or mapped, to at least one logical block. The function, or operation, is then inherited through generalizations.

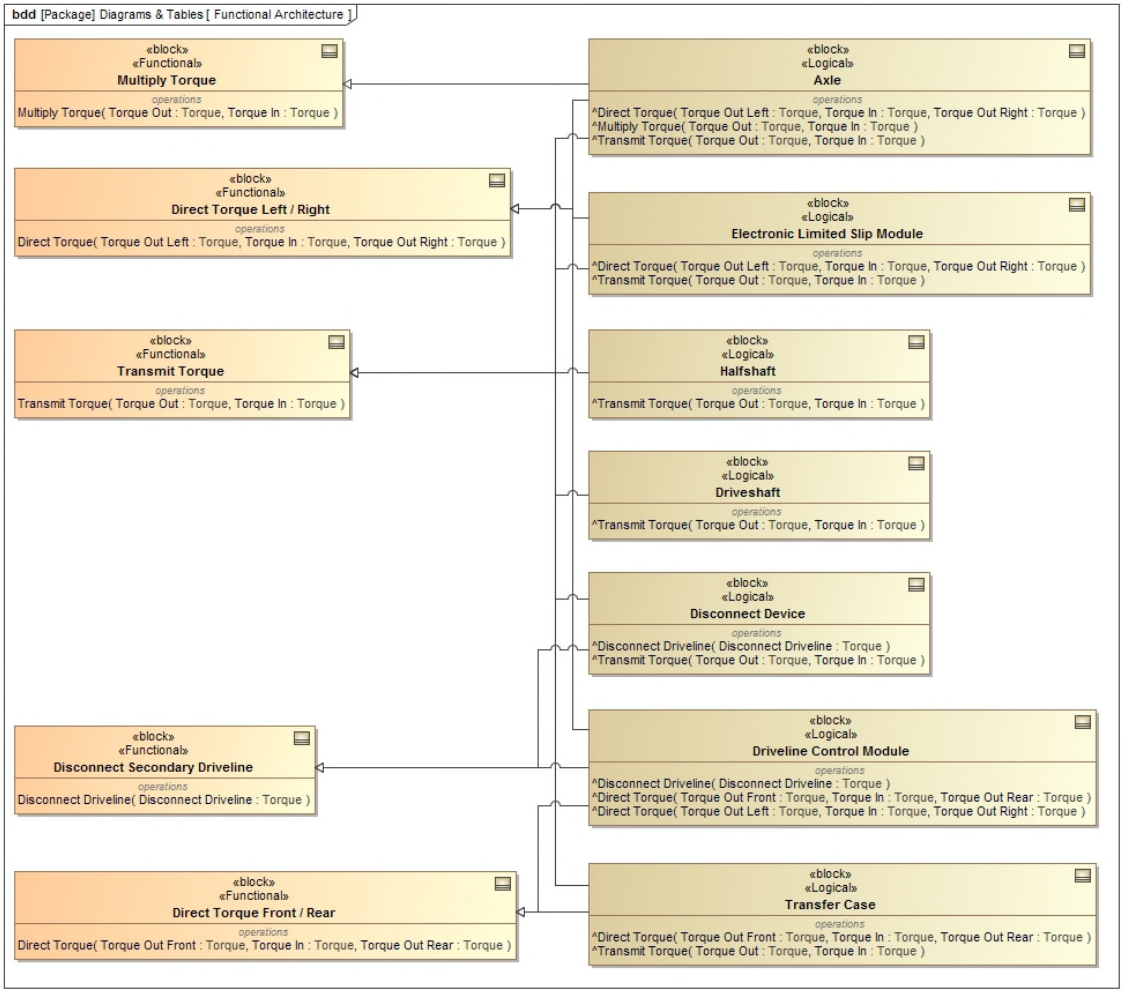


Figure 5-1, Functional Decomposition of Driveline System

Transmission of propulsion force from the powertrain to the wheels is the most basic, and most obvious, driveline function. At the functional level of abstraction the degree or amount of torque transferred is not necessarily relevant – all that matters is that all drivelines transmit torque. In our model this function is inherited by all mechanical sub-systems and components. It is important to note that ***no one component alone delivers or satisfies the transmit torque function***. This is an emergent behavior of the total driveline system.

The next two functions identified are similar / related. *Direct torque left / right* is a behavior of an axle or transaxle differential. The differential takes the incoming torque, turns it ninety degrees and splits it to the left and right wheels. It is important to note that functions are independent of the design concepts that accomplish the behavior. An open differential, limited slip differential, locking differential and electronic limited slip differential all accomplish the same basic function – direct torque left and right. But they accomplish the task with varying degrees of effectiveness from the perspective of the customer. *Direct torque fore / aft* is similar, but is associated with the logical block *Transfer Case* and is only applicable to AWD or 4x4 vehicles. In some design concepts these functions can be electronically controlled, and as such are also associated with the driveline control module.

The function *multiply torque* is associated with the axle or front-wheel-drive transmission final drive ratio. For efficiency, the rotational output speed from the transmission output is much higher than wheel speed. Angular velocity and torque are inversely related. As the driveshaft speeds go up, the torques carried by the driveline go down and the physical size of the shafts and joints can be made smaller. In our model, the *multiply torque* function is only associated with the axle.

The last identified function is to disconnect the secondary driveline. In practice, the secondary driveline in AWD or 4x4 systems is disengaged to reduce rotational losses, improve fuel economy and improve NVH. Of the five functions, this is the one least connected to customer expectations. It is possible

to develop a driveline system without a disconnect mechanism, but for the purposes of our model implementation this is considered to be a basic function of the system. The disconnect function is associated with the disconnect device and the driveline control module.

Logical Decomposition

The driveline system logical decomposition starts with the functional decomposition. Once the system functions are defined, the system engineer is able to parse the functional requirements into distinct logical blocks. This decomposition is directed through engineering judgment and experience. *Figure 5-2, Logical Architecture of IRS Rear Axle Driveline System* illustrates the logical architecture of an IRS driveline with available AWD. The IRS driveline block in the upper left is connected to the sub-systems and components using solid diamond arrows, indicating composition relationships. This diagram shows the value properties, reference properties and the operations associated with each logical block.

During the process it is important to describe the various sub-systems and components in as abstract or general a manner as possible. A good logical decomposition should remain untethered to any specific design concept, since it is unlikely that the first design concept will be the best design concept. *Figure 5-3, Logical Architecture of IRS Powertrain System* shows how the IRS driveline fits into the overall IRS powertrain system. The driveline is a sub-system of the powertrain, no different than the engine or transmission.

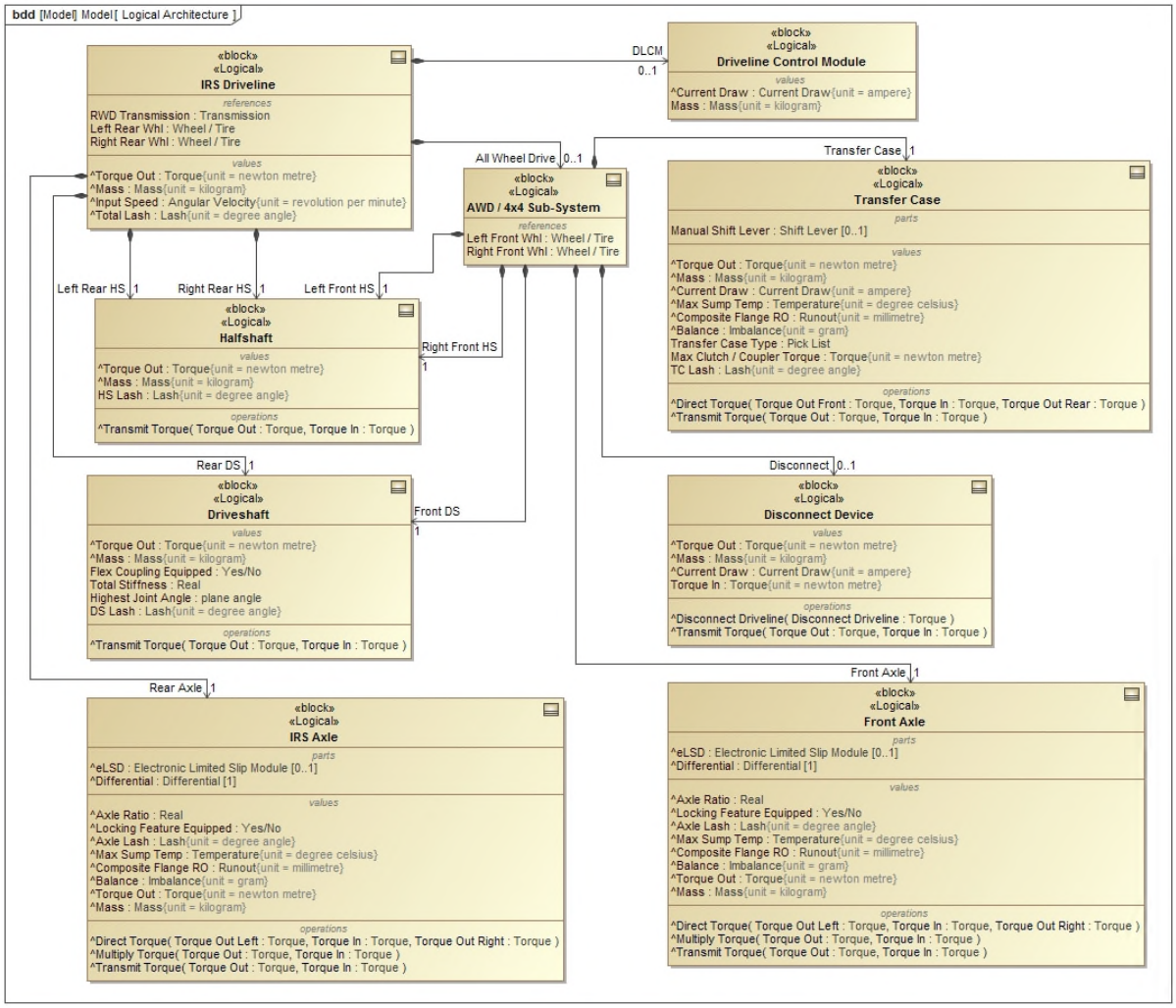


Figure 5-2, Logical Architecture of IRS Rear Axle Driveline System

This logical diagram provides the structure required to capture the vehicle inputs required to support our parametric equations for calculating driveline impact torques. A properly configured and bounded driveline system engineering model can be capable of defining the minimum customer requirements to size and specify the driveshaft, rear axle and halfshafts.

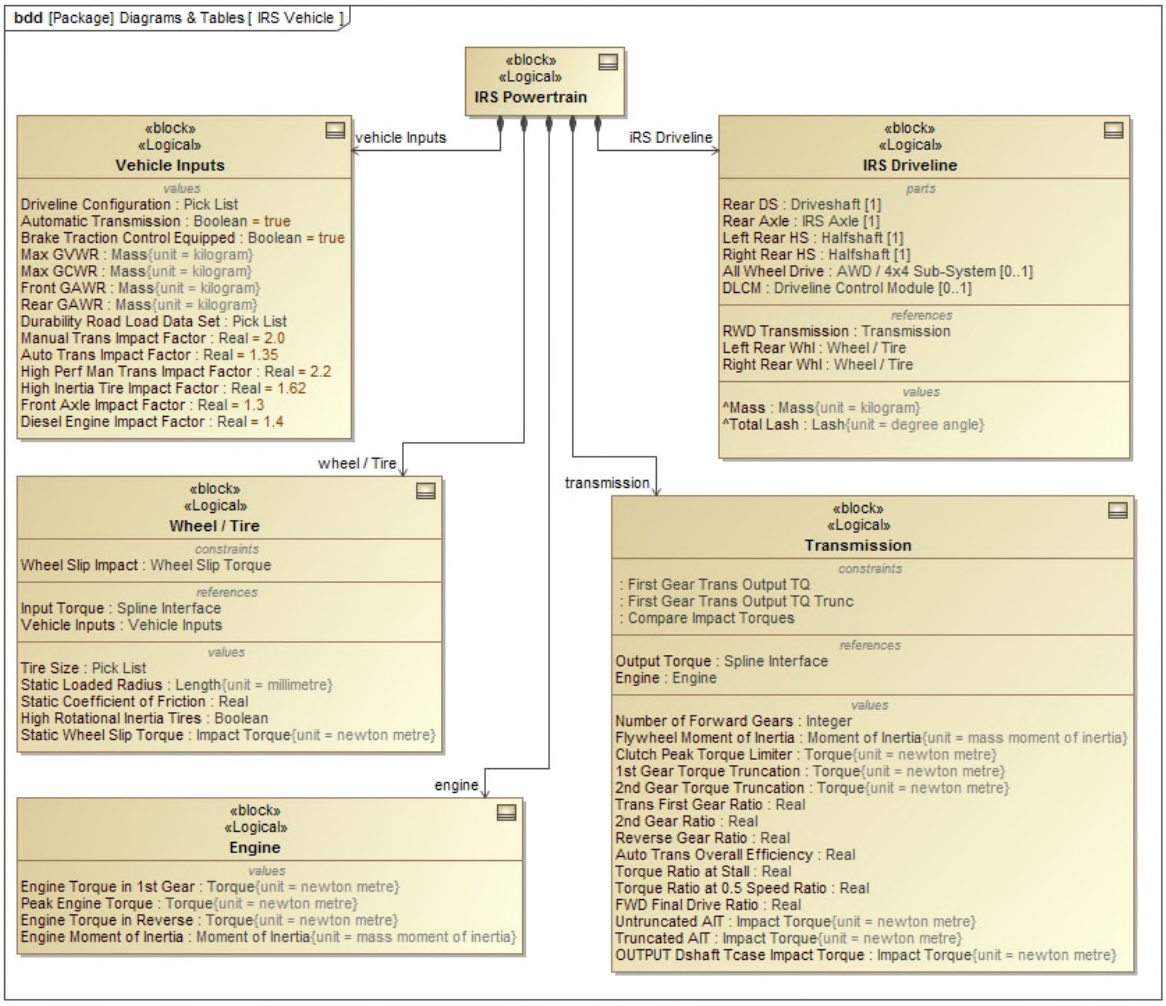


Figure 5-3, Logical Architecture of IRS Powertrain System

Creation of Internal Relationships

Internal relationships are created in SysML through the use of internal block diagrams (IBD). Internal block diagrams are based on UML composite structure diagrams and include restrictions and extensions as defined by SysML. An IBD captures the internal structure of a block in terms of properties and connections among properties. A block includes properties, so that its values, parts, and references to other blocks can be specified. An IBD created for a block (as a

model inner element) will only display the inner elements of the block, such as *parts*, *ports* and *connectors*. An IBD created for a package will display additional elements such as shapes, notes and comments.

Figure 5-4, Internal Block Diagram of AWD / 4x4 Sub-System shows the IBD for the generic all-wheel drive system. Note that it only includes parts that are part of the AWD sub-system. The outer boundary of the diagram is essentially an abstract representation of the outer boundary of the assembly. All properties and connectors that appear inside an IBD belong to, or are owned by, the block whose name is written in the diagram heading. That one particular block is the *context* of the diagram. SysML allows any property (part) to be shown in an internal block diagram to display compartments within the property (or part) symbol (No Magic Inc. 2015b).

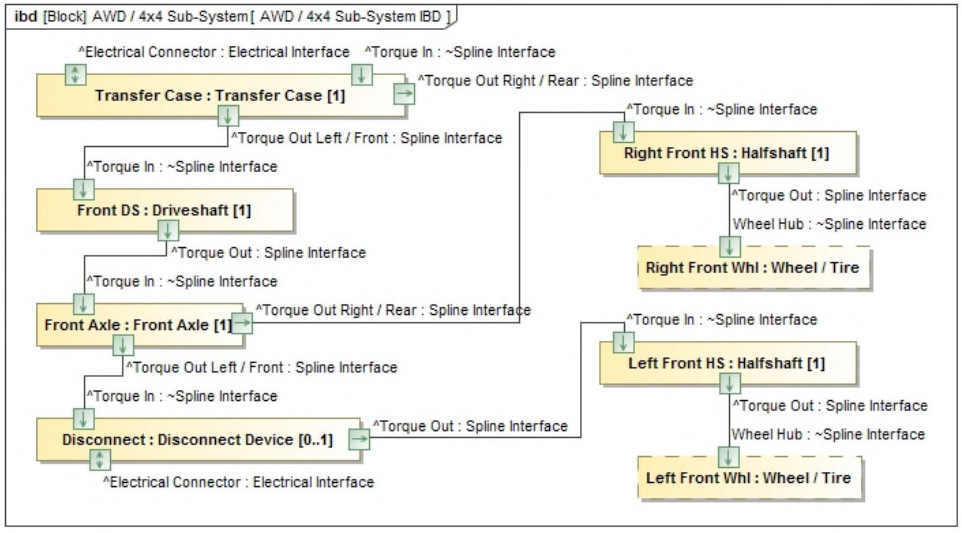


Figure 5-4, Internal Block Diagram of AWD / 4x4 Sub-System

Figure 5-5, Internal Block Diagram of IRS Driveline System is the IBD for the IRS AWD driveline, or independent rear suspension all-wheel drive. This is a primary rear wheel drive based system with the engine and transmission packaged longitudinally in the vehicle. The engine transmits torque to the transmission which then multiplies the torque according to the ratio of the gear that the transmission is operating in. The transmission is outside the driveline system control volume, and hence appears as a ***reference property*** in the internal block diagram, indicated by the dashed boundary. The AWD sub-system referenced earlier is a *part* of the IRS driveline system and appears inside this diagram.

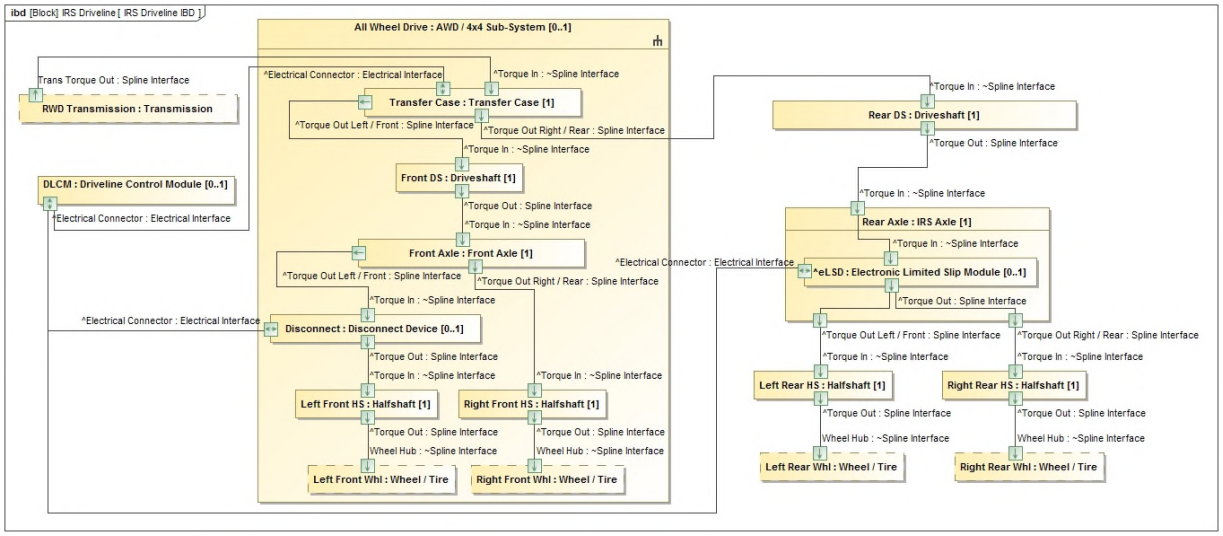


Figure 5-5, Internal Block Diagram of IRS Driveline System

The torque output from the transmission is fed into the transfer case through a spline interface which is shown as a port on the transmission with a flow property of *torque out*, and with a port on the transfer case with a flow property of *torque in*. The transfer case then splits the torque output to the front and rear

axles through the front and rear driveshaft. The front / rear torque split depends on driver demand, terrain conditions and vehicle dynamics. The transfer case has an electrical interface which connects to the driveline control module and is indicated separately on the IBD. Each electrical connection or data flow can be illustrated as a connector on the diagram. The connection to the rear driveshaft is also a spline interface and is shown as a port on the transfer case with a *torque out* flow property. The rear driveshaft block has a corresponding port which has a *torque in* flow property. SysML allows the system engineer to specify the flow direction of a port through a ***conjugation*** property and will ensure that the flow directions match -up.

The rear driveshaft transmits torque through another spline interface to the rear axle, which is shown by another *torque out* port. The corresponding *torque in* port on the rear axle then sends that torque directly to an electronic limited slip differential (eLSD), which is a part of the axle, so the port on the eLSD still shows *torque in*. The eLSD also has an electrical interface port, which is connected to the driveline control module. The differential multiplies the input torque by the axle ratio and then splits torque into two output torque components, one for the left rear halfshaft and the other for the right rear halfshaft. These are shown as spline interface ports with *torque out* flow properties. The amount of torque that is sent to each halfshaft depends on the traction at each wheel, and this is accomplished by the eLSD and controlled by the driveline control module. The halfshafts then transmit torque to the wheel hubs through spline interfaces which have *torque in* and *torque out* flow property ports.

The transfer case output torque to the front driveshaft can be described as a subsystem, which can be called the all-wheel drive subsystem. The architecture is primarily rear wheel drive and hence the front drive system becomes part of the all-wheel drive system. If the architecture was primarily front wheel drive, then the rear drive system would have become part of the all-wheel drive sub-system. Here the *torque out* port from the transfer case to the front driveshaft is a spline interface with a *torque out* flow property. The corresponding *torque in* port is also a spline interface shown on the driveshaft. The driveshaft then transmits that torque through another spline interface to the front axle, which is shown as a port with *torque out* as the flow property. This output torque is then routed through a disconnect device, which outputs torque to the left front halfshaft, through similar spline interfaces with *torque out* and *torque in* flow properties.

The disconnect system disconnects torque to the right halfshaft in concert with the front axle. A characteristic of a differential is that torque to the left and right is always balanced (equal). As any winter driver probably knows, if one drive wheel is on ice, and one wheel is on dry pavement, the vehicle doesn't move. The differential disconnects the left hand halfshaft thereby dropping the wheel torque to zero. Since there is no reaction torque, the torque to the right wheel also drops to zero, the differential will begin to over-spin, and the axle ring gear will stop.

Finally, the four wheels and tires are captured on the IBD as *reference properties* since they are outside of the IRS AWD system. The electrical

connections between the driveline controller and the various electrical components are shown in an extremely simplified manner for illustrative purposes only. In a more detailed IBD the connections would detail both electrical power and information flows.

6. Requirement Management

As discussed in Chapter 3 and applied in Chapter 5, a complex system is represented graphically in MBSE as a series of logical blocks with defined relationships. The ultimate goal of MBSE is to efficiently and reliably guarantee that the customer's needs are satisfied with high quality, reliable, efficient solutions delivered on time throughout a system's entire life cycle. In practice, this means documenting and verifying that the system meets the customer's requirements. The methodology to accomplish this goal is requirements traceability through SysML.

Import Requirements

As mentioned in earlier chapters, document-based systems engineering results in reams of paper and electronic data. At Ford Motor Company, a basic IRS driveline system has over three-hundred documented requirements, including system specifications, sub-system specifications, design rules and basic assumptions. But the three-hundred number is misleading. This is only the number of ***unique*** requirements that were identified after importing the Ford Motor Company control documents into the system model. The original number was over five-hundred requirements

If a Ford system engineer was required to certify all of the IRS driveline requirements, he or she would have to go through all five-hundred requirements individually. In some cases the various requirements are actually conflicting. As requirements change, they need to be coordinated so that the changes are

captured across each instance, but they often are not, resulting in inconsistencies. And the total number of requirements are likely to grow. New requirements are likely to be written in the future to accommodate changing technology as automotive drivelines adopt electronic slip control, electric secondary drive and supplemental electronic content.

The reason the Ford system requirements and specifications are so cumbersome is that they were developed and managed in a piecemeal fashion. Many were developed from customer usage, to deliver better functional performance or improved durability. Some were developed at the manufacturing level to improve ease of assembly, to carry a lower number of unique parts or to achieve modularity. Some were developed at the customer service level to improve serviceability or ensure compatibility with commonly available parts, such as fluids that require frequent change intervals. Some were developed by top management for geopolitical or business reasons. These *reasons for requirements* align closely with the categories of SysML requirements covered in *Table 2-2, Available SysML Requirement Types*.

In the case of Ford Motor Company's document-based system, requirements with multiple test cases are repeated for each test case. This created a situation where the same requirement was duplicated for every unique test case, thereby increasing the complexity and making the system even less friendly to the engineer.

In order to import requirements from a repository which contains redundant, conflicting and duplicate requirements, data parsing is required. The team had to go through a rigorous and iterative process of reading every one of the five hundred or so requirements and then classifying them into SysML requirement types. After the requirements were classified, separate *comma separated value,* or CSV, files were created for each requirement type containing all the requirements that were classified for the particular type. The critical data columns identified for creating these files contained a unique identifier for each requirement (UID), the requirement name, the text of the requirement and the priority level. Individual CSV files were created for the *functional, performance, interface, design constraint, physical, usability* and *business requirement* types.

The MagicDraw software user interface makes it relatively easy to import requirements once they are classified. The software has an ***import CSV*** feature that allows the user to import multiple fields of data in CSV format. Using the CSV import feature, requirements are imported into a *target package* which in this case was the requirements package in the model. In MagicDraw parlance a *package* is equivalent to a file folder. Within the requirements package, *smart packages* were created for each of the requirement types. Each requirement type CSV file was then imported into the corresponding smart package for the requirement type. This level of organization makes it easy for the user to parse the system requirements. *Figure 6-1, Process Flow Diagram for Requirements Import* illustrates the import process the team followed.

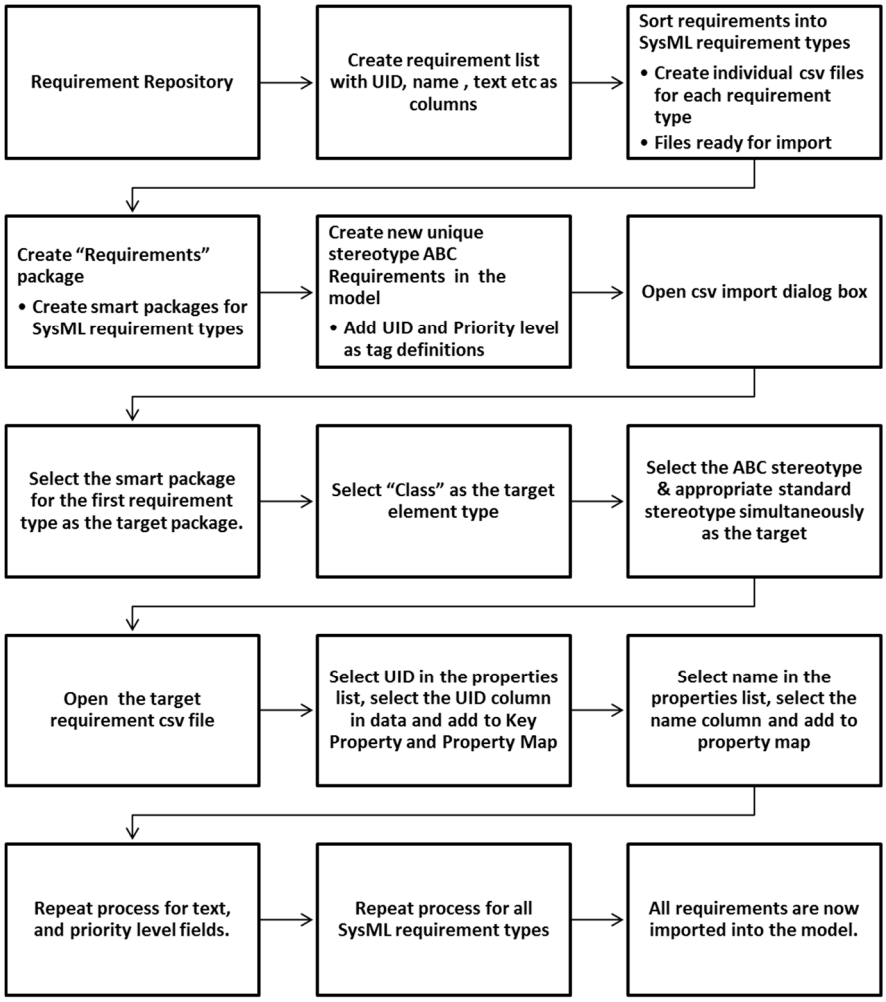


Figure 6-1, Process Flow Diagram for Requirements Import

The import process requires the selection of a *target element type*, which in the case of requirements is called a *class*. A class is a definition for a resource. It includes information that describes the features of an entity and how it can be used. In software terms, code is written as a set of classes and references to behaviors defined by the classes (No Magic Inc. 2015b).

The next step in the process requires the selection of a *target stereotype*. A stereotype defines a new kind of model element by adding properties, constraints

or semantics to an existing kind of model element (Delligatti 2013). Available stereotypes include the standard SysML requirement types, such as *functional requirements* and *performance requirements*. In our case, an additional stereotype was required with unique identifier and priority level tag definitions. Selecting the new stereotype in addition to the standard requirement stereotype corresponding to the type of requirement to be imported allowed the unique identifier to be something other than the name of the requirement. In the case of the driveline system, this was the requirement number. This allowed requirements listed with the same name, such as *Driveline Angle*, to be imported as separate requirements with different requirement numbers.

After all the above parameters are specified in the CSV import dialog, the target file with the specific type of requirement can be selected and then opened. After the dialog box opens the file, the requirement number column is assigned the unique identifier tag, the name column the name tag, the text column the text tag and so forth. The process is repeated for each type of requirement (each type of requirement has its own file) and each requires the corresponding stereotypes to be selected along with the user defined stereotype that was created. The imported requirements' stereotypes can be changed without much effort, which means that even if the system modeler made a mistake in classifying the requirement before the import, he or she can easily fix it in the model.

Requirements are an important part of the entire system model and this is evident when the *satisfy* and *verify* relationships are considered. Logical elements have to satisfy every requirement and test cases are used to verify every

requirement. Hence, the ***proper classification of requirements in the system model is crucial for their association to the proper logical elements and test cases***. Some generic test cases such as a *design review* can be created for almost every requirement, but a performance requirement requires a specific test case with an objective numerical value for verification. The logical elements that satisfy the requirements also require careful attention; otherwise the software will report errors.

Creation of Test Cases

System requirements must be verified to ensure that the system performance meets customer expectations. What is typical industry-wide is that requirements are verified by tests. In the language of SysML, tests are called ***test cases***. To ensure system performance every requirement in SysML must be verified by at least one test case. In the case of our driveline system model, many of the requirements were imported from the Ford corporate requirement repository into SysML. The imported requirements had associated *verification tests*; therefore, the same repository was used to generate the CSV list of test cases for import into our driveline system model.

Once the test case list was identified, the actual process followed for the import of test cases into MagicDraw is very similar to the process followed for the requirements import. When a test case list is generated, care must be taken to import the correct associated data. In our model, we imported the unique Ford test identification number (test ID), test name, test description, prototype type

and associated requirements. The Ford test ID was used as the SysML UID. *Figure 6-2, Process Flow Diagram for Creation of Test Cases* lays out the process followed for test case importation and incorporation into the driveline system model.

The first step to import the test cases into MagicDraw is to save the test case list as a CSV file. Second, create a *Test Case* package in the model. After the test case package is created, a new stereotype needs to be created for test cases. This new stereotype is used to add the definitions *Test Case UID* for test ID number and *Prototype Type* for the level of prototype used for the test. Then open the CSV import dialog box in MagicDraw. In the dialog box, select the *Test Case* package as the target package. Once the package is selected, select *Activity* as the target element type. Then select the standard SysML *Test Case* stereotype and concurrently select the recently created unique test case stereotype. Finally, select the test case CSV file and open it.

Figure 6-2, Process Flow Diagram for Creation of Test Cases

The next phase of the import involves mapping, which is assigning the appropriate SysML properties to the relevant columns of data from the CSV file. In the property list select the *Test Case UID* tag and the *Test ID* column in the CSV data and add to the key property field and to the property map. This will delete all duplicate test cases to prevent redundancy. Second, select *Name* in the property list and the *Test Name* column in the CSV data and add to the property map. Third, select *Prototype Type* from the properties list and select the

Prototype Type column in the CSV data file and add to the property map. Finally, select *Owned Comment* from the properties list and select *Test Operating Conditions (Description)* from CSV data file. Once mapping is completed, the test case import is finished. All of the test cases will be documented and available for association with the previously imported system requirements.

Verification Matrix

The ***verification matrix shows the verify relationships*** from test cases to the appropriate requirements. At this point, all the requirements and test cases have been imported into the model. The next step involves creation of the verify relationships from the test cases in the model to the appropriate requirements. When there are hundreds of requirements and test cases, as in the case of the driveline system model, the process of making the verify relationships is a very arduous one. Each requirement is verified by one or more test cases, and each verify connection can be made using the requirement verify matrix, or using a table format.

A test case table was created with all the test cases within the model. This method of making the verify relationships ensures accuracy, as each row is selected and the verify relationships are made individually. The model has one hundred and eighty two individual test cases, and as a result it took a significant amount of time to make verify relationships from each test case to the

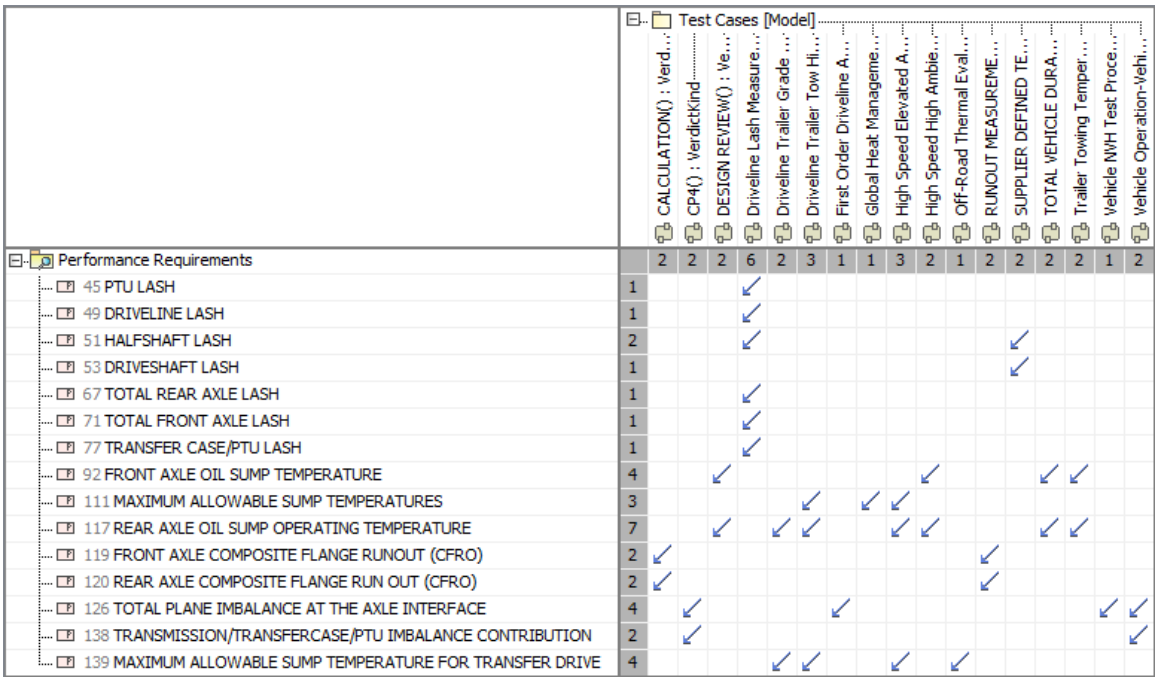
appropriate requirements. Several test cases satisfied multiple requirements as is evident from *Table 6-1, Requirements Verified by Test Case*.

Table 6-1, Requirements Verified by Test Case

Requirement Type	Number of Unique Requirements	Total Number of Test Cases (For Each Rqmt. Type)
Functional Requirements	0	0
Design Constraint Requirements	284	471
Performance Requirements	15	36
Physical Requirements	4	5
Interface Requirements	0	0
Business Requirements	9	11
Usability Requirements	14	18

Over five hundred verify relationships were made in the process of veryifyng requirements. Design constraints were the most numerous type of requirement as there are two hundred and eighty four design constraints in the system model. All of the verify relationships for design constraints addedup to four hundred and seventy one relationships. This is easily visualized in the MagicDraw software, but hard to reproduce as a picture in a word document. *Table 6-2, Verification Matrix for Performance Requirements* illustrates the MagicDraw verification matrix for the driveline model performance requirements.

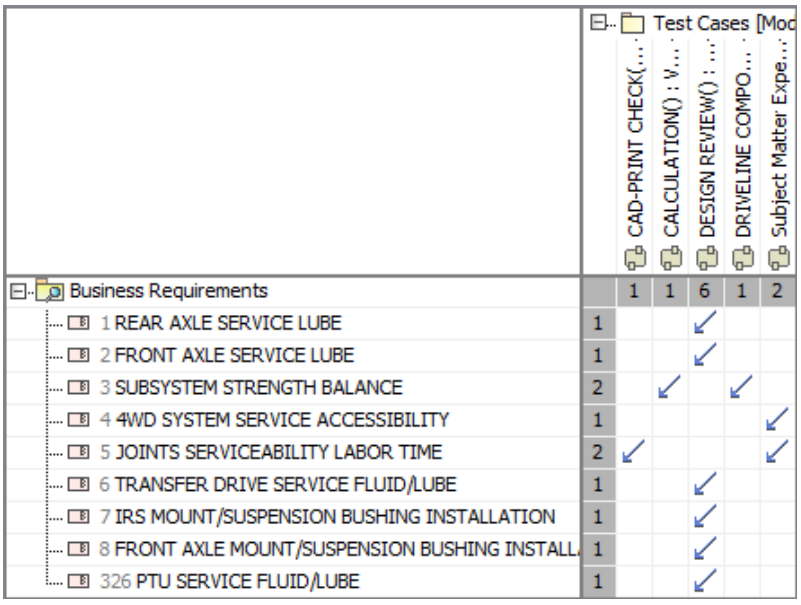
Table 6-2, Verification Matrix for Performance Requirements



Performance Requirements		Test Cases [Model]: CALCULATION() : Verd...	CP4() : VerdictKind	DESIGN REVIEW() : Ve...	Driveline Lash Measure...	Driveline Trailer Grade ...	Driveline Trailer Tow Hi...	First Order Driveline A...	Global Heat Manageme...	High Speed Elevated A...	High Speed High Ambie...	Off-Road Thermal Eval...	RUNOUT MEASUREME...	SUPPLIER DEFINED TE...	TOTAL VEHICLE DURA...	Trailer Towing Temper...	Vehicle NVH Test Proce...	Vehicle Operation-Vehi...
		2	2	2	6	2	3	1	1	3	2	1	2	2	2	2	1	2
45 PTU LASH	1				↙													
49 DRIVELINE LASH	1				↙													
51 HALFSHAFT LASH	2				↙									↙				
53 DRIVESHAFT LASH	1													↙				
67 TOTAL REAR AXLE LASH	1				↙													
71 TOTAL FRONT AXLE LASH	1				↙													
77 TRANSFER CASE/PTU LASH	1				↙													
92 FRONT AXLE OIL SUMP TEMPERATURE	4			↙							↙				↙	↙		
111 MAXIMUM ALLOWABLE SUMP TEMPERATURES	3						↙		↙	↙								
117 REAR AXLE OIL SUMP OPERATING TEMPERATURE	7			↙		↙	↙			↙	↙				↙	↙		
119 FRONT AXLE COMPOSITE FLANGE RUNOUT (CFRO)	2	↙											↙					
120 REAR AXLE COMPOSITE FLANGE RUN OUT (CFRO)	2	↙											↙					
126 TOTAL PLANE IMBALANCE AT THE AXLE INTERFACE	4		↙					↙									↙	↙
138 TRANSMISSION/TRANSFERCASE/PTU IMBALANCE CONTRIBUTION	2		↙															↙
139 MAXIMUM ALLOWABLE SUMP TEMPERATURE FOR TRANSFER DRIVE	4					↙	↙			↙		↙						

Table 6-3, Verification Matrix for Business Requirements is another example of a verify relationship matrix, but for the business requirements. The slanted arrows indicate a relationship, or link between the performance requirements on the X-axis and the test cases on the Y-axis.

Table 6-3, Verification Matrix for Business Requirements



There are other ways to vizualize the verify relationship in the system model, such as a table. The corresponding table for performance requirements is shown in *Table 6-4, Verify Table for Performance Requirements*. Due to size, this table is only a partial tabulation of the complete requirement validation.

Table 6-4, Verify Table for Performance Requirements

#	Name	Verified By	Ford UID
1	PTU LASH	Driveline Lash Measurement() : VerdictKind	RQT-070800-015306/2
2	DRIVELINE LASH	Driveline Lash Measurement() : VerdictKind	RQT-050000-009097/9
3	HALFSHAFT LASH	SUPPLIER DEFINED TEST() : VerdictKind Driveline Lash Measurement() : VerdictKind	RQT-050401-012386/6
4	DRIVESHAFT LASH	SUPPLIER DEFINED TEST() : VerdictKind	RQT-050101-009167/13
5	TOTAL REAR AXLE LASH	Driveline Lash Measurement() : VerdictKind	RQT-050204-015457/16
6	TOTAL FRONT AXLE LASH	Driveline Lash Measurement() : VerdictKind	RQT-050304-011338/8
7	TRANSFER CASE/PTU LASH	Driveline Lash Measurement() : VerdictKind	RQT-070700-018143/4
8	FRONT AXLE OIL SUMP TEMPERATURE	TOTAL VEHICLE DURABILITY (TABLES 5A-5G) TEST RESULTS() : Ve DESIGN REVIEW() : VerdictKind High Speed High Ambient Axle Temperature Test() : VerdictKind Trailer Towing Temperature Stabilization() : VerdictKind	RQT-050304-011361/3
9	MAXIMUM ALLOWABLE SUMP TEMPERATURES	Global Heat Management and Cooling Efficiency() : VerdictKind High Speed Elevated Ambient Driveline Temperature Verification() Driveline Trailer Tow Highway-City Temperature Verification() : Ver	RQT-070800-015295/2
10	REAR AXLE OIL SUMP OPERATING TEMPERATURE	TOTAL VEHICLE DURABILITY (TABLES 5A-5G) TEST RESULTS() : Ve DESIGN REVIEW() : VerdictKind High Speed Elevated Ambient Driveline Temperature Verification() Driveline Trailer Tow Highway-City Temperature Verification() : Ver Driveline Trailer Grade Temperature() : VerdictKind High Speed High Ambient Axle Temperature Test() : VerdictKind Trailer Towing Temperature Stabilization() : VerdictKind	RQT-050204-015525/9
11	FRONT AXLE COMPOSITE FLANGE RUNOUT (CFRO)	CALCULATION() : VerdictKind RUNOUT MEASUREMENT (AXLDR)() : VerdictKind	RQT-050304-011344/6
12	REAR AXLE COMPOSITE FLANGE RUN OUT (CFRO)	CALCULATION() : VerdictKind RUNOUT MEASUREMENT (AXLDR)() : VerdictKind	RQT-050204-015511/10
13	TOTAL PLANE IMBALANCE AT THE AXLE INTERFACE	CP4() : VerdictKind First Order Driveline Analysis() : VerdictKind Vehicle Operation-Vehicle Sensitivity to Driveline Imbalance-Cruise Vehicle NVH Test Procedure Vehicle Operation - Cruise - Smooth Ro	RQT-050000-009113/6
14	TRANSMISSION/TRANSFERCASE/PTU IMBALANCE CONTRIBUTION	CP4() : VerdictKind Vehicle Operation-Vehicle Sensitivity to Driveline Imbalance-Cruise	RQT-050000-009100/9
15	MAXIMUM ALLOWABLE SUMP TEMPERATURE FOR TRANSFER DRIVE	High Speed Elevated Ambient Driveline Temperature Verification() Driveline Trailer Tow Highway-City Temperature Verification() : Ver Off-Road Thermal Evaluation of the Driveline() : VerdictKind Driveline Trailer Grade Temperature() : VerdictKind	RQT-070700-018006/22

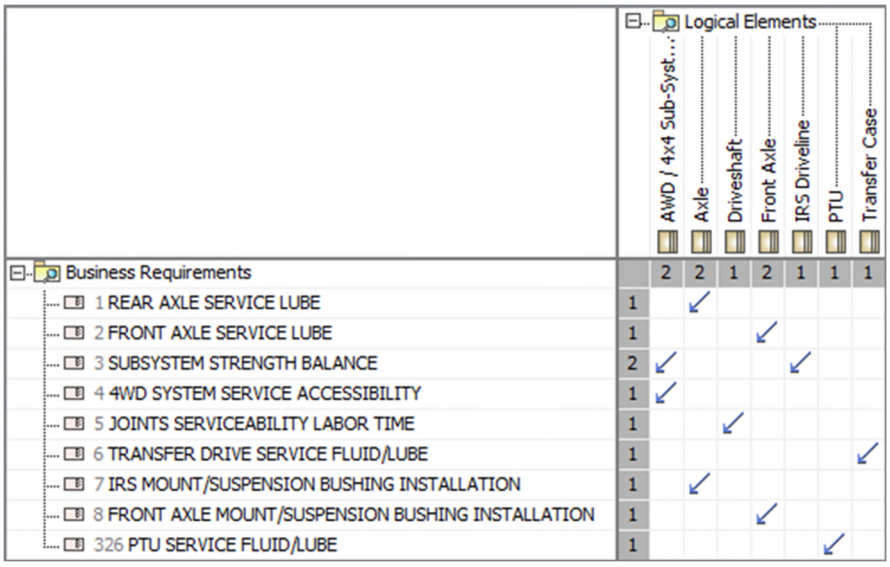
It is possible to input the verify relationships graphically, by drawing connectors between the requirement and test case blocks, textually, by creating the relationship in the specification window, or via the verification matrix by clicking in the appropriate box and selecting *create relationship*. All of these methods are equivalent when creating the relationship abstraction. Once all relationships are created, it is a very easy check to confirm that every requirement is linked to at least one test case and that every test case verifies at least one requirement.

Satisfy Matrix

Requirements are eventually satisfied by physical elements, and in the case of the driveline system, a physical element could be a specific model of transfer case or front axle. In MBSE this relationship is created at the logical level of abstraction, prior to any physical parts being designed. The ***satisfy matrix indicates that each requirement is mapped to at least one logical block through a satisfy relationship*** that indicates that the logical block is required to deliver or meet the requirement. Each requirement is then connected to at least one logical element that satisfies the particular requirement.

The majority of the requirements are design constraints and the satisfy matrix is very large and not easily shown on a word document. Using MagicDraw however, it is easy to parse through the satisfy matrix for design constraints, and it is easy for any sub-system or component engineer to understand which design constraints his or her system needs to satisfy. The same holds for all requirements. One example of a satisfy matrix for logical elements that satisfy business requirements, is given in *Table 6-5, Satisfy Matrix for Business Requirements*.

Table 6-5, Satisfy Matrix for Business Requirements



Another example for a satisfy matrix shows the satisfy relationships between a set of logical blocks which contain value properties and performance requirements. *Table 6-6, Satisfy Matrix for Performance Requirements* shows the satisfy matrix for performance requirements.

Satisfy relationships can also be created and viewed in a tabular format. The table format can be used to separate the requirements out by sub-system or component as illustrated in *Table 6-7, Satisfy Table for Performance Requirements*. The table shows all the requirements that are satisfied by the driveline system , along with the unique identifiers for the requirements.

Table 6-6, Satisfy Matrix for Performance Requirements

	Logical Elements													
		Axle / T												
		Axle Lash : Lash	Balance : Imbala...	Composite Flang...	Max Sump Temp...	Total Lash : Lash	DS Lash : Lash	HS Lash : Lash	PTU Lash : Lash	TC Lash : Lash
Performance Requirements		2	2	2	4	1	1	1	2	1
45 PTU LASH	1								1 ↙	
49 DRIVELINE LASH	1					1 ↙				
51 HALFSHAFT LASH	1							1 ↙		
53 DRIVESHAFT LASH	1						1 ↙			
67 TOTAL REAR AXLE LASH	1	1 ↙								
71 TOTAL FRONT AXLE LASH	1	1 ↙								
77 TRANSFER CASE/PTU LASH	2								1 ↙	1 ↙
92 FRONT AXLE OIL SUMP TEMPERATURE	1		1		↙					
111 MAXIMUM ALLOWABLE SUMP TEMPERATURES	1		1		↙					
117 REAR AXLE OIL SUMP OPERATING TEMPERATURE	1		1		↙					
119 FRONT AXLE COMPOSITE FLANGE RUNOUT (CFRO)	1		1	↙						
120 REAR AXLE COMPOSITE FLANGE RUN OUT (CFRO)	1		1	↙						
126 TOTAL PLANE IMBALANCE AT THE AXLE INTERFACE	1		1 ↙							
138 TRANSMISSION/TRANSFERCASE/PTU IMBALANCE CONTRIBUTION	1		1 ↙							
139 MAXIMUM ALLOWABLE SUMP TEMPERATURE FOR TRANSFER DRIVE	1		1		↙					

Table 6-7, Satisfy Table for Performance Requirements

#	Name	Satisfied By	Ford UID
1	VEHICLE DURABILITY SIMULATION	Driveline System	RQT-050101-009126/14
2	ALLOWABLE OPERATING TEMPERATURE	Driveline System	RQT-050000-009066/13
3	DESIGN FOR MAXIMUM DRIVELINE TORQUES	Driveline System	RQT-050000-011640/15
4	FASTNERS TORQUE-ANGLE CHARACTERISTICS	Driveline System	RQT-050101-009183/4
5	CRASH WORTHINESS	Driveline System	RQT-050000-009086/14
6	POWERTRAIN BENDING FOR NVH	Driveline System	RQT-050000-009102/10
7	HIGH SPEED DURABILITY	Driveline System	RQT-050000-009081/9
8	DRIVELINE/PROPSHAFT RADIATED NOISE	Driveline System	RQT-050000-009112/4
9	MODAL SEPARATION IN GEAR MESHING FREQUENCY RANGE	Driveline System	RQT-050000-009094/11
10	TOTAL PLANE IMBALANCE AT THE POWERPLANT INTERFACE	Driveline System	RQT-050000-009092/16
11	SUBSYSTEMS COMPATIBILITY	Driveline System	RQT-050000-009065/7
12	DRIVELINE TOTAL DURABILITY	Driveline System	RQT-050000-009063/12
13	DRIVELINE IMPACT DURABILITY	Driveline System	RQT-050000-009064/15
14	DRIVELINE JOINTS SPECIFICATION	Driveline System	RQT-050000-009120/4
15	SYSTEM SERVICEABILITY	Driveline System	RQT-050000-011656/3

There are a total of three hundred and twenty six individual requirements that are satisfied by logical element blocks. A better illustration of the number of requirements satified are given in *Table 6-8, Requirements Satisfied by Logical*

Block. The table shows all the logical elements in the driveline model that satisfy requirements, and the requirements are separated into the seven different SysML requirement types.

Table 6-8, Requirements Satisfied by Logical Block

Logical Elements	No. of Functional Reqmt. Satisfied	No. of Des. Constraint Reqmt. Satisfied	No. of Perf. Reqmt. Satisfied	No. of Physical Reqmt. Satisfied	No. of Interface Reqmt. Satisfied	No. of Business Reqmt. Satisfied	No. of Usability Reqmt. Satisfied
AWD / 4x4 Sub-System		16		1		2	
AWD Sub-System		4		1		2	
Axle		54				2	2
Axle Lash:Value Properties::Lash			2				
Axle Ratio:Real							
Axle / T-Case Inheritance							
Balance:Value Properties::Imbalance			2				
Composite Flange RO:Value Properties::Runout			2				
Max Sump Temp:Value Properties::Temperature			4				
Beam Axle		3		1			
Controls Inheritance Block		1					
Coupling		3					
Disconnect Device		4					
Driveline Control Module		1					
Driveline System		14					1
Total Lash:Value Properties::Lash			1				
Driveshaft		109				1	5
DS Lash:Value Properties::Lash			1				
Electronic Limited Slip		1					
Electronic Locking		1					
Front Axle		11				1	2
Halfshaft		21					1
HS Lash:Value Properties::Lash			1				
IRS Driveline		1				1	
Mechanical Limited Slip		2					
PTU		17		1		1	
PTU Lash:Value Properties::Lash			1				
Shift Lever		5					2
Transfer Case		20		1		1	1
TC Lash:Value Properties::Lash			1				
Transmission		7					
Wheel / Tire		2					

7. Parametric Relationships

In systems engineering, design involves making decisions between solution alternatives. Alternatives are most often compared and evaluated based on engineering metrics, such as weight or performance. The general process is to generate alternatives (ideation), evaluate alternatives (engineering analysis) and finally, decide between alternatives (interpretation of results). SysML provides a language to express and perform mathematical system analysis through parametric diagrams. Parametric diagrams show mathematical relationships between the blocks of the system model. They act as constraints on the system design.

Sizing Inputs in System Model

The main benefit and purpose of the driveline sizing tool discussed in Chapter 4 is to obtain impact torque definition and joint sizing. But it also serves as an upfront component and system optimization tool for cost, weight, package size and component quality. In effect, the sizing tool is a method for analysis between different design solutions, meaning it is a perfect candidate for modeling in a parametric diagram. If properly constructed, the SysML model can perform the same basic functions of existing driveline sizing tools if it contains the key interface parameters for the driveline system.

To perform driveline sizing within the SysML model it is important to define the sizing calculation inputs as value properties within the model. These value properties must be associated and linked to the correct logical constructions at

the sub-system and sub-component level. *Table 7-1, Parametric Inputs to Sizing Model* defines the logical association for each required sizing input within the model along with the correct value type. In SysML a value type is the unit of measure. For our model we have elected to use SI units, though the existing Ford Motor Company sizing tool uses English units.

Table 7-1, Parametric Inputs to Sizing Model

Parametric Input	Logical Block Ownership	Value Type
Vehicle Assumptions		
Driveline Configuration (Type)	Vehicle Inputs	Pick List
Brake Traction Control Equipped	Vehicle Inputs	Yes / No
Max GVWR	Vehicle Inputs	Kg
Max Front GAWR	Vehicle Inputs	Kg
Max Rear GAWR	Vehicle Inputs	Kg
Max GCWR	Vehicle Inputs	Kg
Durability Road Load Data Set (RLD)	Vehicle Inputs	Pick List
Engine		
Engine Torque in 1st Gear	Engine	N-m
Highest Engine Torque Available	Engine	N-m
Engine Torque in Rev. Gear	Engine	N-m
Engine Moment of Inertia	Engine	MMOI
Transmission		
Transmission Type	Transmission	Pick List
1st Gear Torque Truncation	Transmission	N-m
2nd Gear Torque Truncation	Transmission	N-m
Transmission Efficiency	Transmission	Real #
Torque Converter Ratio @ Stall	Transmission	Real #
Torque Converter Ratio @ 0.5 Speed Ratio	Transmission	Real #
Transaxle FDR	Transmission	Real #
Flywheel Moment of Inertia - Manual Trans	Transmission	MMOI
Clutch Peak Torque Limiter - Manual Trans	Transmission	Real #
Number of Forward Gears	Transmission	Integer
1st Gear Ratio	Transmission	Real #
2nd Gear Ratio	Transmission	Real #
Reverse Gear Ratio	Transmission	Real #
Driveline		
4x4 Transfer case type – Active/On Demand	Transfer Case	Pick List
4x4 Transfer case Max Coupler Torque	Transfer Case	N-m
Flex Coupling Equipped	Driveshaft	Yes / No
Total Driveshaft Stiffness	Driveshaft	N-m/rad
Highest Joint Angle @ Design	Driveshaft	Degrees
Rear Axle Ratio	Axle	Real #
Locking Differential Equipped	Axle	Yes / No
Wheels / Tires		
High Rotational Inertia Tire Equipped	Wheel/Tire	Yes / No
Tire Size	Wheel/Tire	Pick List
Max Tire SLR	Wheel/Tire	Pick List
Tire Coefficient of Static Friction	Wheel/Tire	Real #

To be useful, these inputs must be associated or related to logical constructions within our SysML model. *Figure 7-1, SysML Diagram of Required Inputs for Driveline Sizing* illustrates the mapping of these needed parametric inputs as value properties associated with logical blocks based on the property's logical ownership.

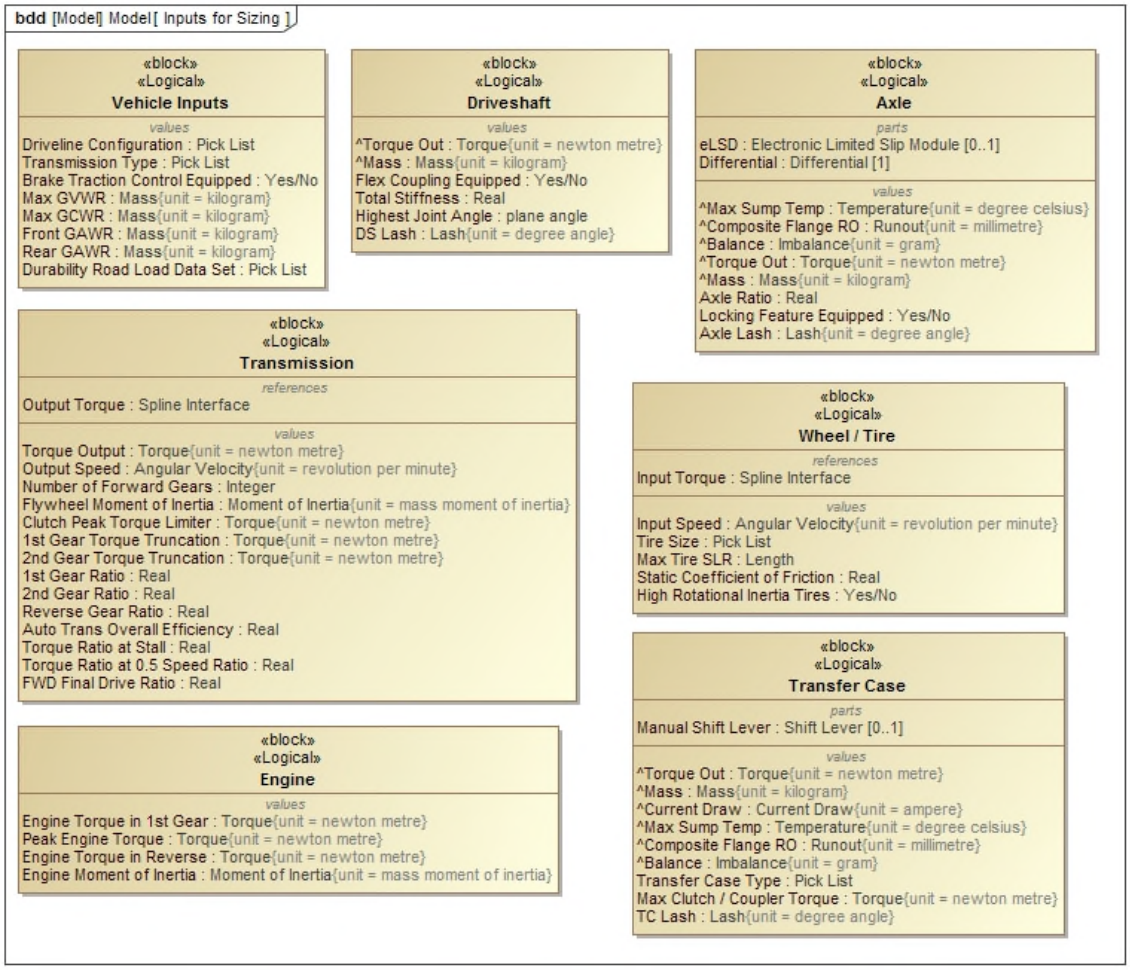


Figure 7-1, SysML Diagram of Required Inputs for Driveline Sizing

The impact torque sizing inputs are the *value properties* associated with the appropriate logical blocks. For example, the value property *Engine Torque in 1st*

Gear is owned by the *Engine* logical block. *Static Coefficient of Friction* is owned by the *Wheel / Tire* logical block. At logical architecture levels, these value properties are *to be defined* variables that are not associated with any specific engine or tire instance. They are defined later on as specific *instances*.

Parametric Constraint Modeling

One of the more useful modeling constructs offered in SysML is the parametric constraint used in parametric diagrams. Parametric constraints specify equivalence relationships between logical blocks. *Figure 7-2, SysML Parametric and Requirement Diagrams* shows the relationship of parametric and requirement diagrams in the overall context of all diagram types within SysML. Parametric diagrams are a graphical means of representing mathematical or logical relationships between model elements. They are defined in a similar manner to IBDs, but they use internal relationships with constraint parameters instead of part parameters. They are restricted to connecting only through ***binding connectors***, typically with a parametric constraint at one end of the connection. ***Binding connectors imply equality***.

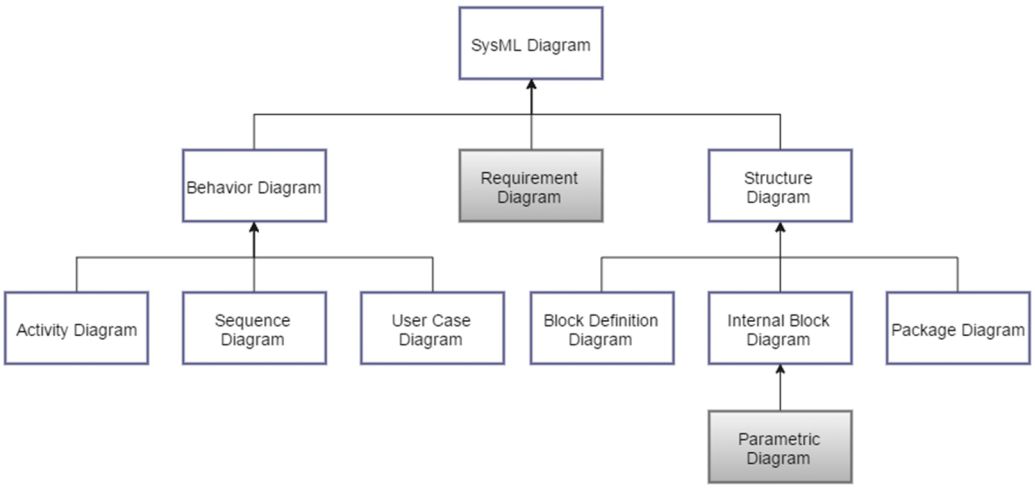


Figure 7-2, SysML Parametric and Requirement Diagrams

The key element found on a parametric diagram is the constraint block, which is used to constrain the properties of one or more other blocks. Constraint blocks are owned by a Block Definition Diagram or Package Diagram. Constraint blocks consist of constraints (any expression such as $\{\tau = F * d\}$) and constraint parameters (such as τ, F and d). It is best to create each constraint at the time you define the constraint block. An equation defined within the constraint block can be any mathematical relationship (the constraint) between variable properties (the parameters). Equations in constraint blocks govern the relationship for the value properties of a model. A constraint parameter is used to ***bind*** or connect a value property (an attribute owned by a block) from the outside to a variable within the constraint equation (Paredis and Davies 2011).

Parametrics Relationships for Driveline Sizing

A goal of our SysML model was to show proof of concept for the calculation of impact, yield and fatigue torques for driveline sizing. This required the

creation and integration of parametric relationships. We began by creating individual constraint blocks to model the torque transfer physics of impact loads. There are a total of eight different constraint blocks used to model impact torque, consisting of torque flow equations, logic flow and Boolean assignments for build content that can relate / display internal relationships to the parent constraint block as well as other blocks through the binding connections.

Figure 7-3, SysML Parametric Diagram for Driveshaft represents the driveshaft constraint block created and displayed in a parametric diagram. This constraint takes the Boolean value for *Flex Coupling Equipped* and calculates the flex coupling factor used in one impact torque calculation. The value for *Flex Coupling Equipped* is one of the value properties found in *Table 7-1, Parametric Inputs to Sizing Model*. The function returns the value 0.95 if the driveline has a flex coupling and 1.0 if the driveline does not have a flex coupling.

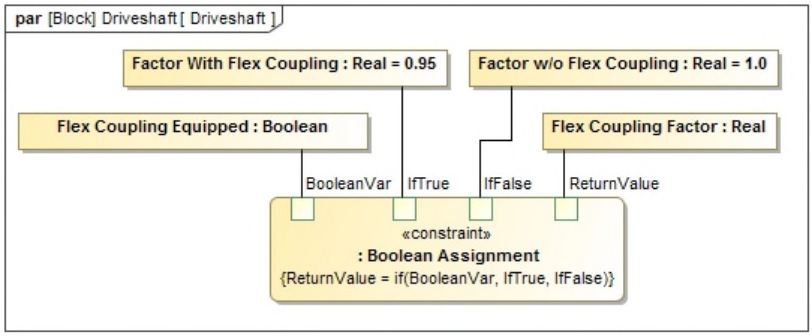


Figure 7-3, SysML Parametric Diagram for Driveshaft

For our proof of concept sizing exercise, we aligned all input value parameters to that of a proposed CD622 program, which provided us the advantage of known values for verification. As seen in the driveshaft diagram,

Boolean assignments as well as logic flow can be handled in SysML through a *state* of a system, which is specified in terms of the values of its properties. Thus a change in state results in a different set of constraint equations to be recalculated. This was accommodated by specifying unique constraints that are conditioned on the value of the property (No Magic Inc. 2015b).

A parametric diagram was also created to represent / contain the overall driveline sizing model for our IRS driveline instance application. The large parametric diagram simultaneously displays individual constraint blocks as well as their unique relationships between respective input values and other constraint blocks, as well as all binding connections. The comprehensive IRS driveline sizing parametric diagram is shown in *Figure 7-4, SysML Parametric Diagram for Impact Torque Calculation*. This is a very complex, nested parametric that requires inputs from six different logical blocks. Once again, all inputs are captured in *Table 7-1, Parametric Inputs to Sizing Model*.

At this point ***instances*** are created to capture the potential powertrain inputs based on program assumptions. *Figure 7-5, SysML BDD for Powertrain Instances* shows the component instances defined to create the various powertrain installations for our sizing proof of concept. We created three engine models, two transmissions, two driveshafts, two tires, two axle ratios and two sets of vehicle inputs to define three particular vehicle programs. Once the blocks are created, the parametric input parameters are defined within the value properties of the blocks. These specific component instances are used to build

and define powertrain instances based on the parametric diagrams constructed for the IRS Powertrain.

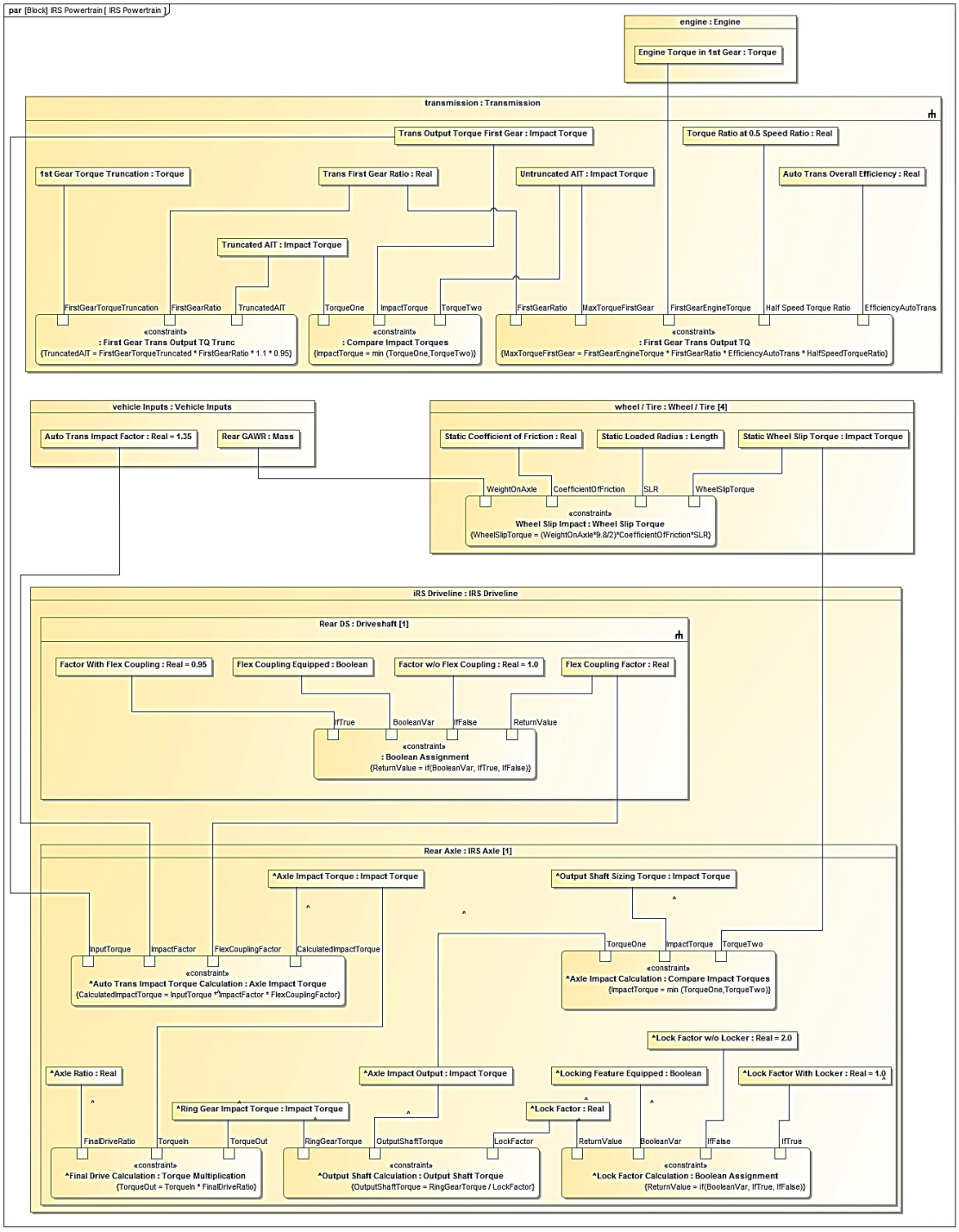


Figure 7-4, SysML Parametric Diagram for Impact Torque Calculation

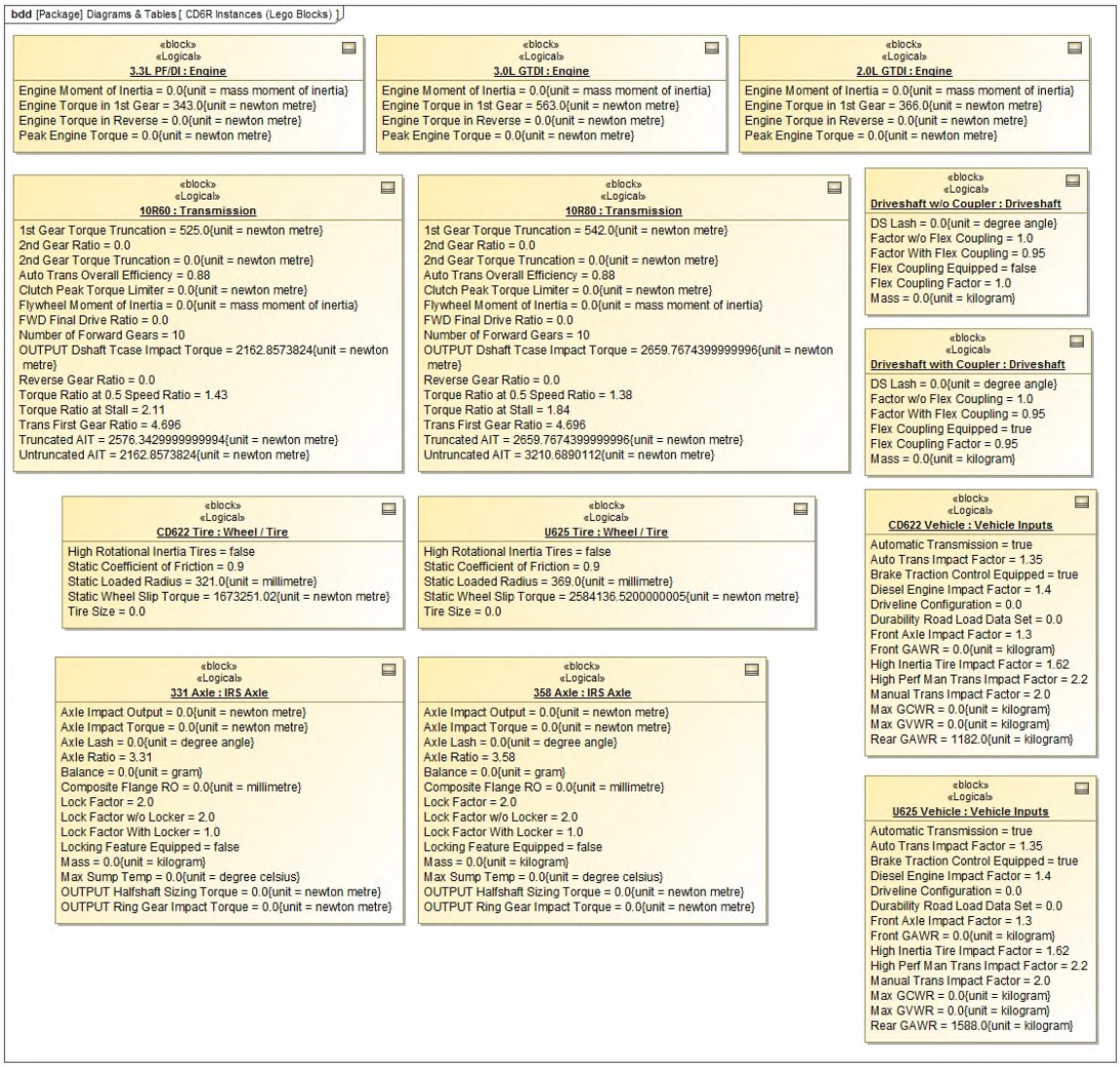


Figure 7-5, SysML BDD for Powertrain Instances

The component instances are used like interchangeable bricks to build up the vehicle instances. The components are mixed and matched to create instances based on the system engineer's needs. Then a simulation is run to calculate the various system outputs. SysML is capable of exporting the simulation results in table form. Our three proof of concept runs are captured in *Table 7-2, Instance*

Table for Impact Torque Output. Each run of the simulation outputs the sizing torques for driveshaft, axle and halfshaft.

Table 7-2, Instance Table for Impact Torque Output

#	Name	Transmission : Transmission	Wheel / Tire : Wheel / Tire	Engine : Engine	OUTPUT Dshaft Tcase Impact Torque : Impact Torque	OUTPUT Halfshaft Sizing Torque : Impact Torque	OUTPUT Ring Gear Impact Torque : Impact Torque
1	CD622 CM1 Powertrain	10R60 : Transmission	CD622 Tire : Wheel / Tire	2.0L GTDI : Engine	2576.342999999...	5756.194347749...	11512.38869549...
2	U611 CM1 Powertrain	10R80 : Transmission	U625 Tire : Wheel / Tire	3.0L GTDI : Engine	2659.767439999...	6105.961617821...	12211.92323564...
3	U625 CM2 Powertrain	10R80 : Transmission	U625 Tire : Wheel / Tire	2.0L GTDI : Engine	2659.767439999...	6427.328018759...	12854.65603751...

Thus, with the adaptation of the driveline sizing tools into an MBSE methodology, we have created what is effectively a real-time sizing tool that is responsive and flexible enough to keep up with all design changes throughout the product development process. As program inputs change, ***the driveline sizing tool can automatically recalculate the outputs***. If the outputs are then compared against the component design criteria, such as comparing the calculated halfshaft sizing torque against the designed halfshaft ultimate torque, SysML will automatically flag any inconsistencies. If the program changes a tire size, thereby changing the wheel slip torque, the modeling tool can be used to re-run calculations to ensure the change does not impact the halfshaft, axle or driveshaft useful life.

Notes on Modeling Parametric Diagrams

While we were able to complete the proof of concept for the parametric modeling tool, it significantly stretched the limits of the MagicDraw software. As of version 18.2, MagicDraw parametric diagrams did not support reference properties and the team ran into several simulation bugs that prevented seamless

operation of the calculation tool. Due to software limitations, the simulation could not progress more than two levels down in the system model. This required the team to construct intermediate steps and made the parametric model unnecessarily complex. It required significant effort and an upgrade to MagicDraw version 18.3 Beta to get the model to export the sizing calculations correctly, and even then it required significant engineering effort to store the data. Some of these issues may have been due to the lack of experience we (and our advisor) have with parametric modeling (MagicDraw is routinely used to conduct complex simulations). However, it may also be due to our non-software backgrounds and nuances in the simulation engine more apparent to engineers with a programming background.

Consequently, our assessment is that the current release version of MagicDraw is unsuitable for driveline sizing. However, there is no conceptual roadblock preventing the incorporation of the sizing tool into a future version of the SysML tool, once the software catches up to our modeling efforts. As discussed earlier, MBSE is a developing field. The software tools available to system engineers are being improved constantly and MagicDraw is already working on allowing reference properties to be used in parametric simulations. According to Mike Vinarcik, *"MagicDraw routinely incorporates feedback from users as it enhances capabilities and refines the user interface."* Companies that are early adopters to SysML are likely to have significant influence in the direction of future software development efforts, particularly in efforts to make MBSE tools less opaque to non-programmers.

8. Benefits of Applied MBSE

Object oriented modeling was developed for and has traditionally been applied to software applications, but this paper demonstrates that it can be equally applicable to a fully mechanical system. Executing this project demonstrated that MBSE can provide significant value and improved effectiveness and efficiency when applied to typical automobile related systems engineering problems.

In our limited application, we observed improvements in at least three areas. We saw a reduction in requirement redundancy and consistency across all levels of abstraction; streamlined communication of requirements by making all key input and output parameters available to all model users; and the ability to continuously update and manage component design inputs through parametric relationships with vehicle level inputs.

It is important to note that this driveline system model was completed in just nine weeks by three powertrain engineers with no previous SysML or MagicDraw modeling experience. The MagicDraw application is reasonably intuitive and the driveline system model's complexity and scope increased as the team's familiarity and skill level grew. Learning the SysML language was fairly straightforward, and even with minimal instruction the learning curve was quick. This may have been due to the fact that SysML relationships are logically consistent and make sense to the trained engineer.

Improved Communication

Our comprehensive SysML model provides a rational architecture for the sizing and definition of the driveline system. It begins with the customer's needs, translates those into system requirements and delivers an efficient, optimized solution that tracks and links all system requirements to defined test cases. Implementation of MBSE will improve engineering productivity by linking requirements across all sub-systems and improving tracking and communication of requirement changes. The reduction of requirement redundancies and automatic validation of test case verification could result in the elimination of entire tracking departments in Ford Motor Company. Some other advertised benefits of MBSE are improved design consistency, precision, traceability, subsystem integration and design evolution (Pearce and Friedenthal 2013).

Sizing tools developed based on document-based requirements have limitations on scalability. When there is a new vehicle level requirement identified, the changes required to typical sizing tools can be extensive and not easily accomplished without complete tear-up. New customer requirements may drive new technology development and the implementation of this new technology is a typical example. The physics equations or parametrics describing the new technology or system usually cannot be directly integrated into a typical sizing tool without extensive tear up and coding. In many cases this leads to development of new sizing tools, which may or may not be compatible with traditional systems. This can lead to the existence of multiple sizing tools which can lead to confusion. In the case of a system model however, the new

requirement can be added, and the parametrics added into the SysML model without much effort. This requirement is then cascaded throughout the system making the process much smoother and it also ensures that every component receives any updated or new requirement automatically.

Management of Requirements

MBSE supports requirement management through reduced redundancy, better traceability and linkages between requirements and their methods of verification. The MBSE implementation required significant upfront work to define the logical structure of the mechanical system and the key interfaces. Achieved through the process of logical decomposition, it proved valuable and forced the team to question existing assumptions implicit in the existing document based system engineering currently used at Ford.

Requirements are a key element in the success of a system and its development and are considered by some authors to be *"the cornerstone of systems engineering"* (Salado and Nilchiani 2014). During the elicitation process of requirements, careful categorization to minimize overlap between them is critical. It serves to better partition and bound the problem, while searching for an optimal solution. Ensuring the adequacy of the problem definition should be a critical step in the development process. Excessive specifications along with overlapping of requirements will increase effort, time and the probability of inconsistent information, while potentially reducing the solution space.

As mentioned earlier, the importation of over five-hundred driveline requirements from a control document into the MBSE model required new categorization into three-hundred ***unique*** requirements, better aligned to behaviors and operations that define customer requirements. This was a painstaking iterative process of elicitation refinement. But it also served to root out excessive overlap of constraints in the current requirement lists, much of which existed due to legacy information. As previously mentioned, these constraints were often created piecemeal and at many different levels of development, testing and verification. Through our integration efforts into SysML categories, we discovered a clear need and benefit of improved elicitation and partitioning of existing requirements. Improved management of categorization models offers reduced redundancy, increased quality, and an overall efficient systems engineered solution.

Parametric Input Cascade and Control

Through parametric relationships, top level assumption changes are immediately cascaded down and can be verified against existing component variable properties. If engine torque in first gear goes up, it will immediately be calculated into transmission output torque and compared against the axle maximum input torque limit. Changes in tire properties can be linked to and compared against halfshaft joint design limits automatically. If the input assumptions exceed design limits the SysML model will immediately throw an error code to alert the system and component engineers that their attention is

required. All design data can be stored and managed in one place at an abstract level.

Conclusion

This is believed to be the first application of model based mechanical systems engineering at Ford Motor Company and was intended to be a test, or proof of concept. Based on the success of our limited model, it is the authors' opinion that MBSE will be a significant tool for future automotive system designs. The state of SysML and MBSE appears to be similar to the implementation of CAD applications in the 1980's, or CAE in the 1990's. The MBSE market is slowly developing and SysML is beginning to penetrate industry through early adopters. Effectively, MBSE is an emerging market that is likely to grow in the future as its potential benefits are recognized by engineers.

Companies have not yet identified the benefits of model based system engineering, but when they do, the market will explode. Companies are likely to pursue enterprise implementations to reap the benefits in communication and requirement management across their organizations. At Ford, it is likely that SysML or something similar will be integrated into the existing Teamcenter CAD/CAE tools. If history is any guide, early adopters are likely to absorb the costs, but also guide the future development of this new technology.

References

Balmelli, Laurent. 2007. “An Overview of the Systems Modeling Language for Products and Systems Development -- Part 3: Modeling System Behavior.” *Journal of Object Technology* 6 (6): 149–77.

Carter, Mike (Ford Motor Company). 2015. “Rear Drive Housed Axle Subsystem P-Diagram,” 2.

Delligatti, Lenny. 2013. *SysML Distilled: A Brief Guide to the Systems Modeling Language*. Upper Saddle River, NJ: Pearson Education, Inc.

Kurt Niebuhr. 2014. “How-to-Choose-the-Right-Axle-Ratio-for-Your-Pickup-Truck @ Www.edmunds.com.” *Edmunds.com*. http://www.edmunds.com/car-buying/how-to-choose-the-right-axle-ratio-for-your-pickup-truck.html.

No Magic Inc. 2015a. “Executive Overview: Accelerating The Model Driven Enterprise.” http://www.nomagic.com/files/brochures/letter/MagicDraw_ExecOverview_2012.pdf.

———. 2015b. “SysML Plugin 18.1 User Guide.”

Operations, Technical. 2007. “Systems Engineering Vision 2020.” *Systems Engineering* 1 (September).

Paredis, Chris, and Kevin Davies. 2011. “System Analysis Using SysML Parametrics : Current Tools and Best Practices Acknowledgments.” *Georgia Institute of Technology Model Based Systems Engineering Center*, 62. https://openmodelica.org/images/docs/modprod2011-tutorial/modprod2011-tutorial4-Chris-Paredis-SysML-Parametrics.pdf.

Pearce, Paul, and Sanford Friedenthal. 2013. “A Practical Approach For Modelling Submarine Subsystem Architecture In SysML.” *Submarine Institute of Australia Science Technology & Engineering Conference*.

Ryen, Ed. 2008. “Overview of the System Engineering Process Prepared by,” no. March. https://www.dot.nd.gov/divisions/maintenance/docs/OverviewOfSEA.pdf.

Salado, Alejandro, and Roshanak Nilchiani. 2014. “Categorizing Requirements to Increase the Size of the Solution Space : Moving Away from NASA and ESA ’ S Requirements Categorization Models.” In *SECESA 2014*, 9.

Stuttgart, Germany.

Seymour, Samuel J., and Steven M. Biemer. 2011. *Systems Engineering Principles and Practice*. 2nd ed. Hoboken, NJ: John Wiley & Sons, Inc.

Wellmann, T., Govindswamy, K., Braun, E., and Wolff, K. 2007. “Optimizing Vehicle NVH Characteristics for Driveline Integration,” 1–15. http://www.fev.com/fileadmin/fev-resources/Publications/NVH/Optimizing_Vehicle_NVH_Characteristics_for_Driveline_Integration_01.pdf.