

# Ford

WELCOME

Model-Based Agility

with

FASST 

*Ford Automated System  
Simulation Toolchain*



Robert ter Waarbeek  
Raghu Baskaran  
Steven Foster  
Nick Adams  
Nate Rolfes

...and the Global FASST team !

**VIRTUAL VEHICLE built in MINUTES instead of MONTHS**

# AGENDA

- **Why & What is FASST** Robert ter Waarbeek
- **How does it work** Raghu Baskaran
- **Creating full Vehicle Simulation** Steven Foster
- **Scaling to production** Nick Adams
- **Inner Sourcing for Model Development** Nate Rolfes
- **Closing Statement** Robert ter Waarbeek

# WHAT IS FASST?

# WHY DO WE NEED IT?



## ROBERT TER WAARBEEK

# WHY DO WE NEED FASST?

*“Software and digital systems provide tremendous power in building complex systems not previously possible.*

***But this increase in power comes with a price – large software systems are fiendishly difficult to get correct.***

*The difficulty of building such software is often underestimated by engineers.”*

– Nancy Leveson, Professor of Aeronautics and Astronautics at MIT

Widely recognized as a preeminent expert in system and software safety



**Automotive Vehicles are extremely complex mass produced mechatronics systems**

**with rising complexity its is essential to detect system issues early in development.**

# THE CHALLENGE OF FULL VEHICLE SIMULATION

**ADAS Feature**  
EASE  
Matlab 2012a 32bit  
In-house + Supplier C

Status Q1-2017

**Steering controls**  
Chassis  
Matlab 2011b 32bit +target link  
Supplier A + In-house

**Vehicle Dynamics**  
Models  
VehDyn CAE  
ADAMS  
converted into:  
CarSim  
Dspace ASM  
IPG-Carmaker  
Mathworks-VDDBS

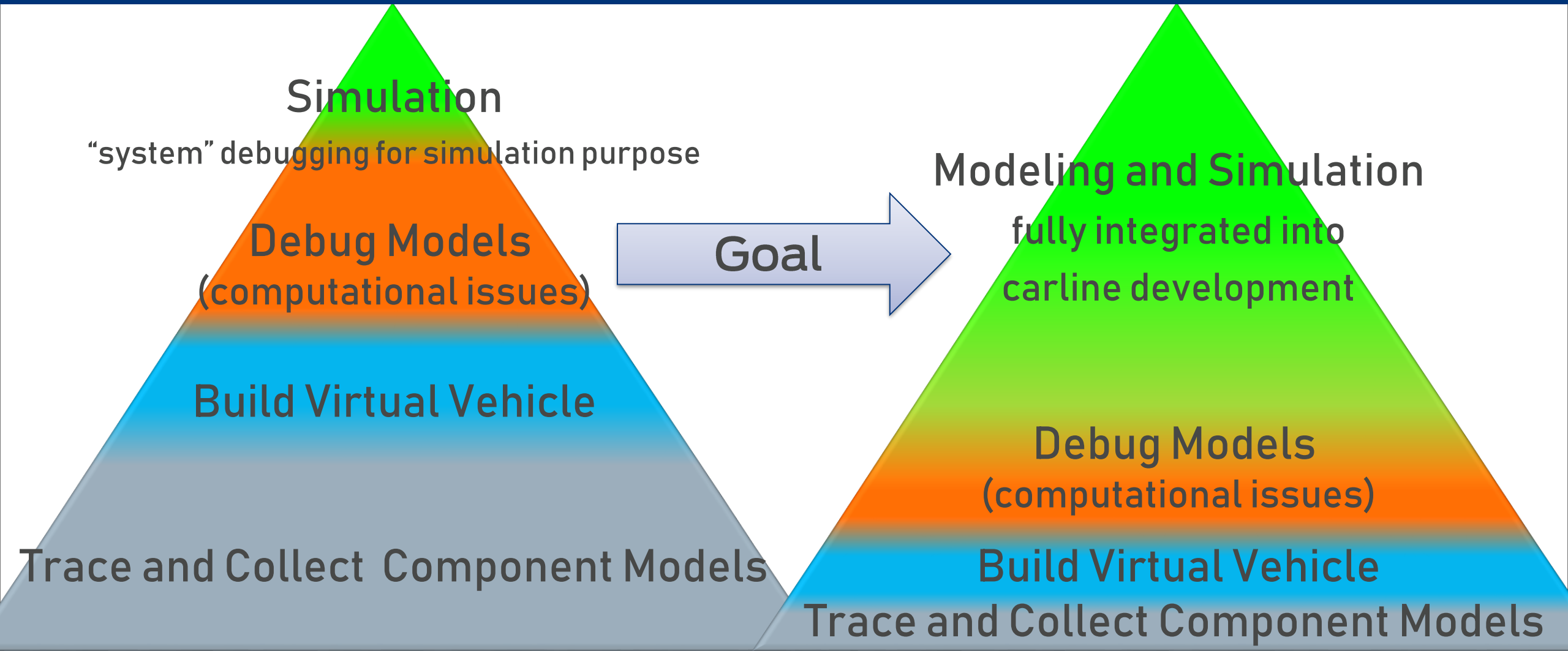
**Powertrain Models**  
Powertrain  
Matlab 2015b 64bit + MBD  
In-house

**Brake Controls**  
Chassis  
Matlab 2014b 32bit  
Supplier B + In-house

ADAS  
Sensor  
Models  
EASE  
CarSim  
Carmaker  
Unreal

For virtual development of distributed systems all teams have to work together

# SIMULATION INEFFICIENCY



Initial State

Desired State

# FORD AUTOMATED SYSTEM SIMULATION TOOLCHAIN (FASST)

## GitHub

40 million+ Global Users

Ford: 10,000+ Users

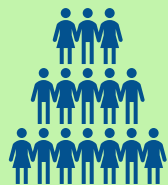


Most widely used source control management tool

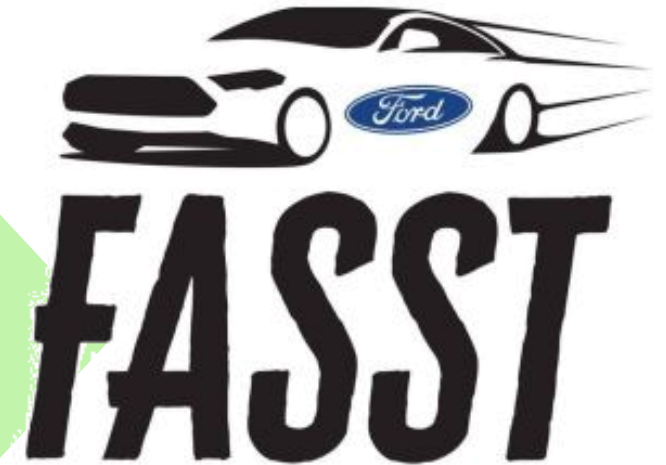
## MATLAB® & SIMULINK®

3 million+ Global Users / 4,500 Employees / 31 Offices Globally

Ford: 7,000+ Matlab / 4,000+ Simulink Users



A Smart Cross Organizational team



500+ Members / Passive Users  
100+ Active Users  
~30 Members on "DevOps" Team

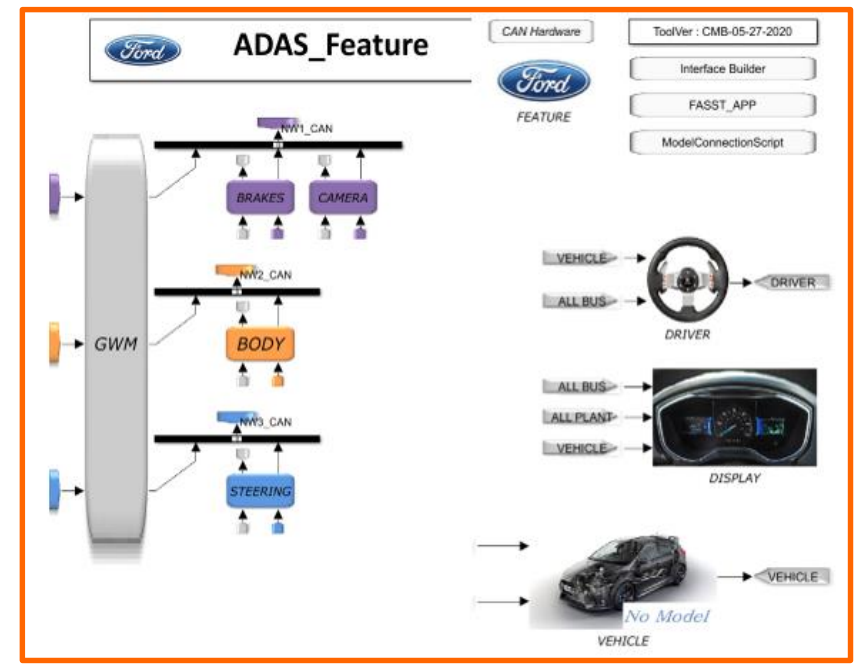
# FASST: BUILDING THE SKELETON MODEL



Program_Name	ADAS	
Feature_Name	Feature	
Variant_Name		
Model Parts	GitHub Organization and Repository	Branch/Tag
BRAKES	FASST/BRAKES	BehvM
CAMERA	FASST/CAMERA	BehvM
BODY	FASST/BODY	SkellM
STEERING	FASST/STEERING	BehvM
FEATURE	FASST/FEATURE	Feature1
DISPLAY	FASST/DISPLAY	Display1
DRIVER	FASST/DRIVER	Driver1



Program_Name	ADAS
Feature_Name	Feature
Variant_Name	
Model Parts	GitHub Organization and Repository
BRAKES	FASST/BRAKES
CAMERA	FASST/CAMERA
BODY	FASST/BODY
STEERING	FASST/STEERING
FEATURE	FASST/FEATURE
DISPLAY	FASST/DISPLAY
DRIVER	FASST/DRIVER



Vehicle controls architecture

Bill Of Models

Skeleton/System Model



# FASST: CREATING A VEHICLE MODEL

## GitHub

MODEL REPOSITORY  
Populate ECU contents from functional software model developers

Brakes

Camera

Steering

...

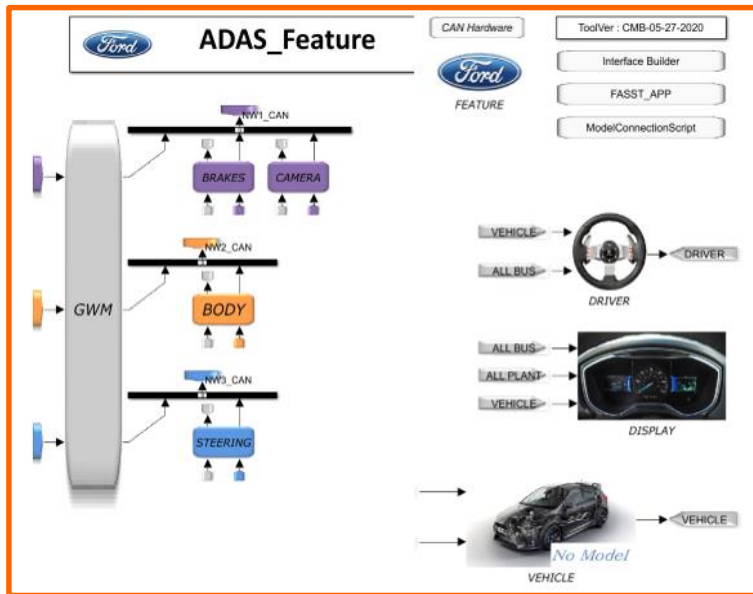
VEHICLE

- **VDBS**
- CarSim
- Carmaker
- ...

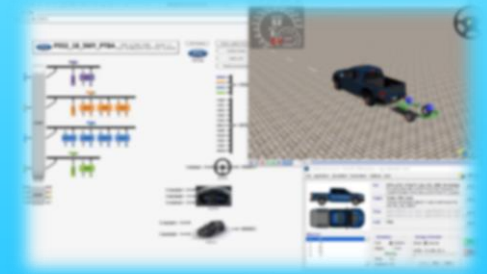
Skeleton/System Model

Component Models

Vehicle models



# FASST: VIRTUAL DEVELOPMENT



There is no single virtual vehicle



DVM,DVP  
with metrics

or

Explorative  
testing

Test Plan



Virtual  
development

# FORD AUTOMATED SYSTEM SIMULATION TOOLCHAIN (FASST)

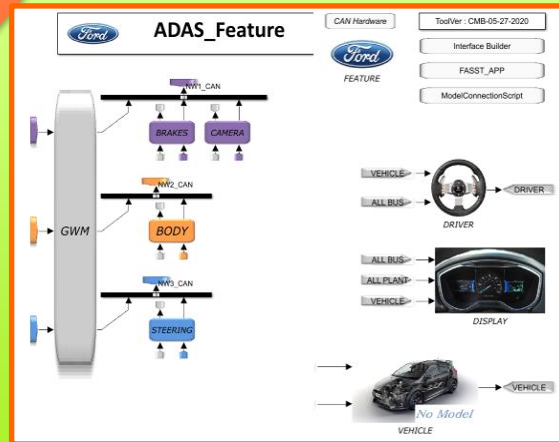
Vehicle controls architecture

Feature model BOM (Bill Of Models)



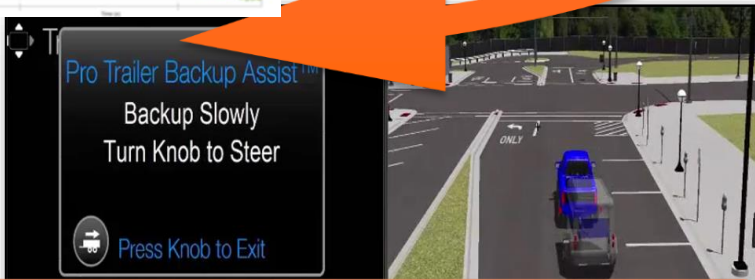
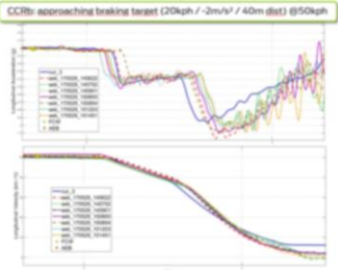
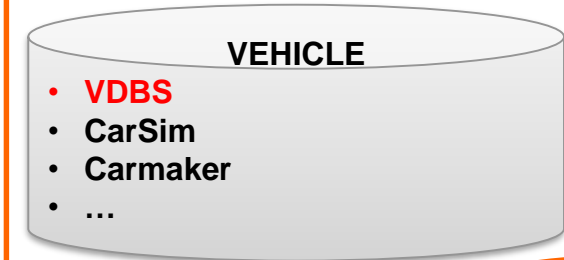
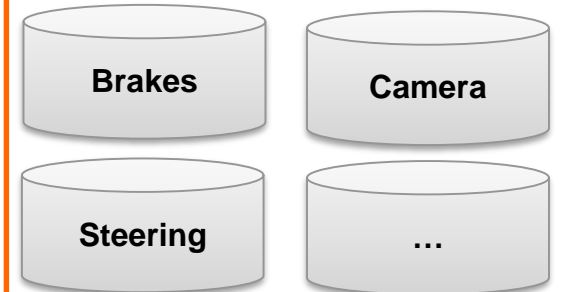
# FASST

Build system/skeleton model & Include components



# GitHub

MODEL REPOSITORY  
Populate ECU contents from functional software model developers



Virtual development & Verification

FASST reduced virtual vehicle build from months into minutes

# HOW DOES IT WORK?

## THE MECHANICS OF BUILDING A FASST MODEL

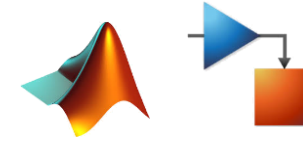


**RAGHU BASKARAN**

# WHAT DO WE NEED TO BUILD A FASST MODEL?

## GitHub

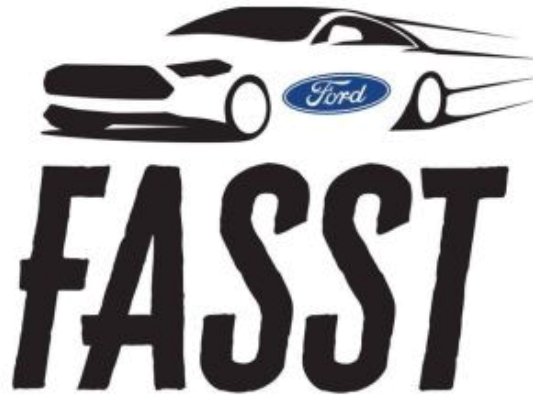
Cloud-Based  
Distributed  
Version Control



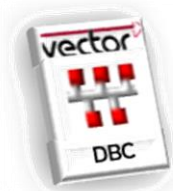
ECU Models &  
Components



Model  
BOM



Vehicle Plant  
Models



ECU Architecture  
(DBC)

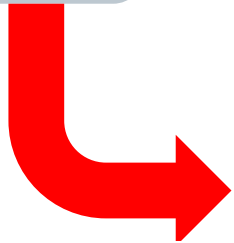


Plant Interface  
Sheet

# IT ALL STARTS WITH THE BILL OF MODELS



Model BOM\*



<metadata>		
Program_Name	ADAS	
Feature_Name	Feature	
Variant_Name		
Model Parts	GitHub Organization and Repository	Branch/Tag
<components>		
BRAKES	FASST/BRAKES	BehvM
CAMERA	FASST/CAMERA	BehvM
BODY	FASST/BODY	SkelM
STEERING	FASST/STEERING	ReqM
FEATURE	FASST/FEATURE	Feature1
DISPLAY	FASST/DISPLAY	Display1
DRIVER	FASST/DRIVER	Driver1
</components>		
<vehicle>		
Vehicle	FASST/Vehicle	BehvM
CarSim		
CarMaker		
VDBS	FASST/VDBS	BehvM
Ford_PowerTrain		
ADAMS		
</vehicle>		
<network>		
DBC	FASST/DBC	Latest_Architecture

Program Name  
Feature Name  
Variant Name

ECU Component Models

Location in GitHub

Model Fidelity or Release Tag

Vehicle Dynamics Model Selection

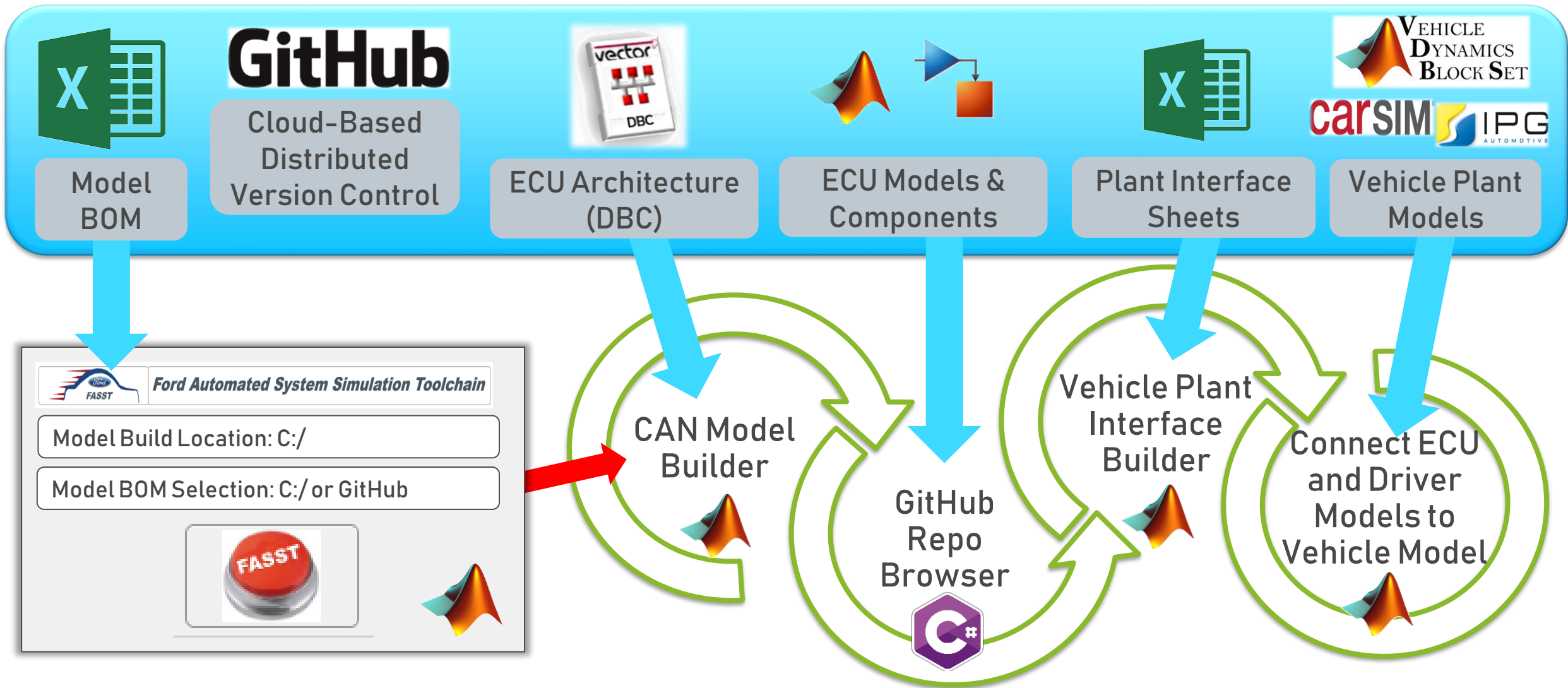
Network / ECU Architecture

\*Additional sections

(not shown) for:

- Build Options
- System Model
- Test Procedures
- Optional Tools
- Documentation
- Test Results
- Miscellaneous

# THE FASST "ONE CLICK" SYSTEM MODEL BUILD



**FASST is an automation wrapper for several linked tools that builds the full system vehicle model from components in the GitHub cloud**

# FASST DEMONSTRATION VIDEO





# ANATOMY OF A FASST MODEL


Program & Feature Name



## ADAS\_Feature

ADAS\_Feature Version: 1.4  
ed: Thu May 28 01:35:27 2020 by RBASKA

CAN Hardware



FEATURE

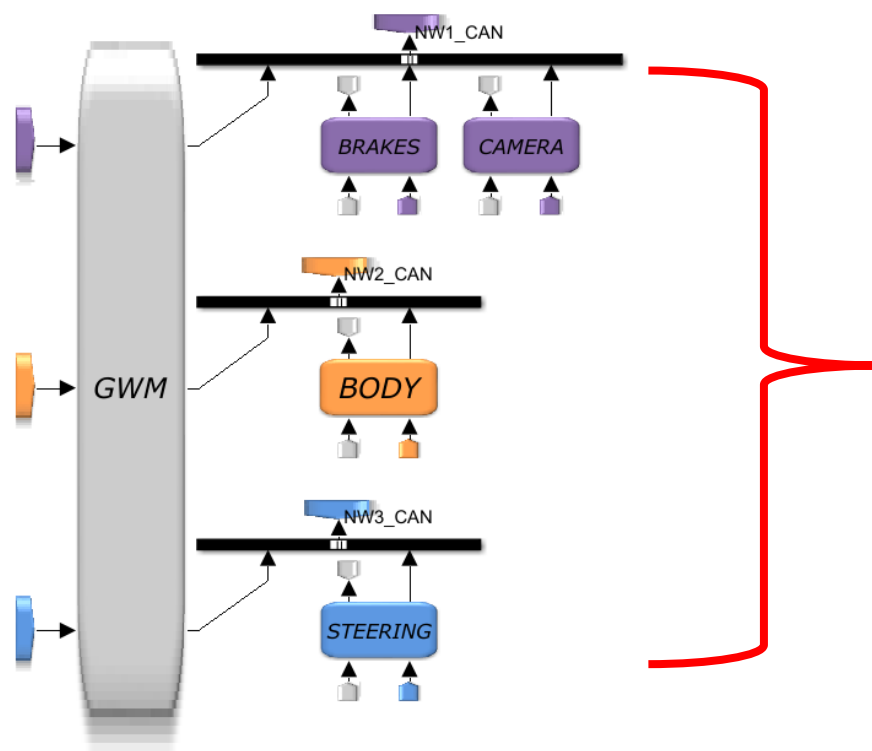
ToolVer : CMB-05-27-2020

Interface Builder

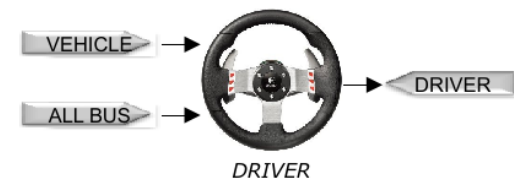
FASST\_APP

ModelConnectionScript

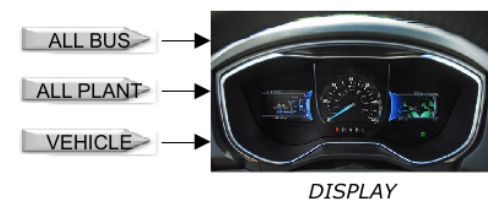
Shortcut Buttons to FASST Tools



Virtual ECU  
"Breadboard"  
MIL, SIL, & HIL  
Ready



Driver Model



Signal Monitors Displays



Vehicle Model

FASST generates an ECU framework model that can accept "plug and play" components to capture as many use cases as possible


# ANATOMY OF A FASST MODEL

Program & Feature Name


**ADAS\_Feature**

*ADAS\_Feature* Version: 1.4  
*ed: Thu May 28 01:35:27 2020 by RBASKA*

CAN Hardware


 FEATURE

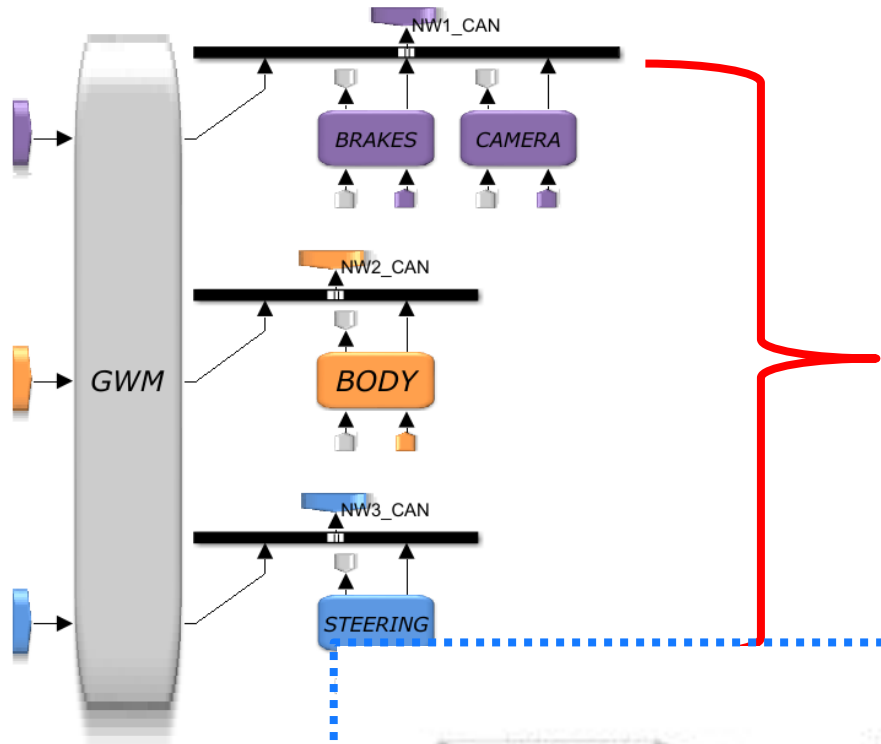
ToolVer : CMB-05-27-2020

Interface Builder

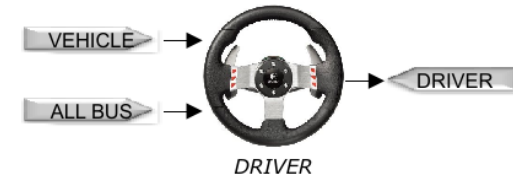
FASST\_APP

ModelConnectionScript

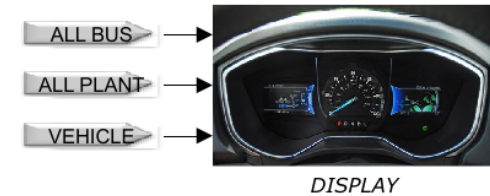
Shortcut Buttons to FASST Tools



Virtual ECU "Breadboard" MIL, SIL, & HIL Ready



Driver Model



Signal Monitors Displays



Vehicle Model

# CREATING FULL VEHICLE SIMULATION: CONNECTING SYSTEM COMPONENTS TO THE VEHICLE MODEL



**STEVEN FOSTER**

# VEHICLE PLANT MODEL FLEXIBILITY IS CRITICAL

## Why Support Multiple Vehicle Plant Models?

Simulation Speed

Licensing Cost / Availability

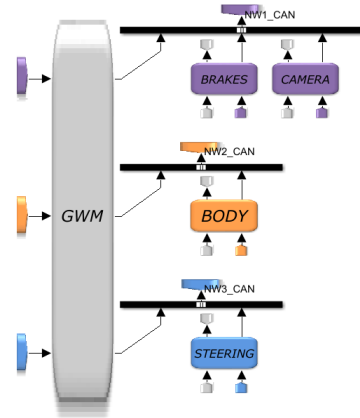
User Experience

Department Preference

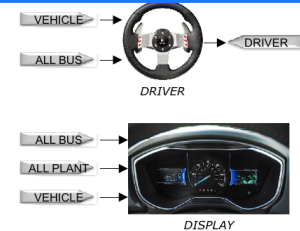
Model Fidelity

Ford **ADAS\_Feature** ADAS\_Feature Version: 1.4  
ed: Thu May 28 01:35:27 2020 by RBASKI

CAN Hardware ToolVer: CMB-05-27-2020  
Interface Builder  
FASST\_APP  
ModelConnectionScript



**FASST with Vehicle Dynamic Blockset**



SIMULINK



VDBS



NO MODEL



CARMAKER



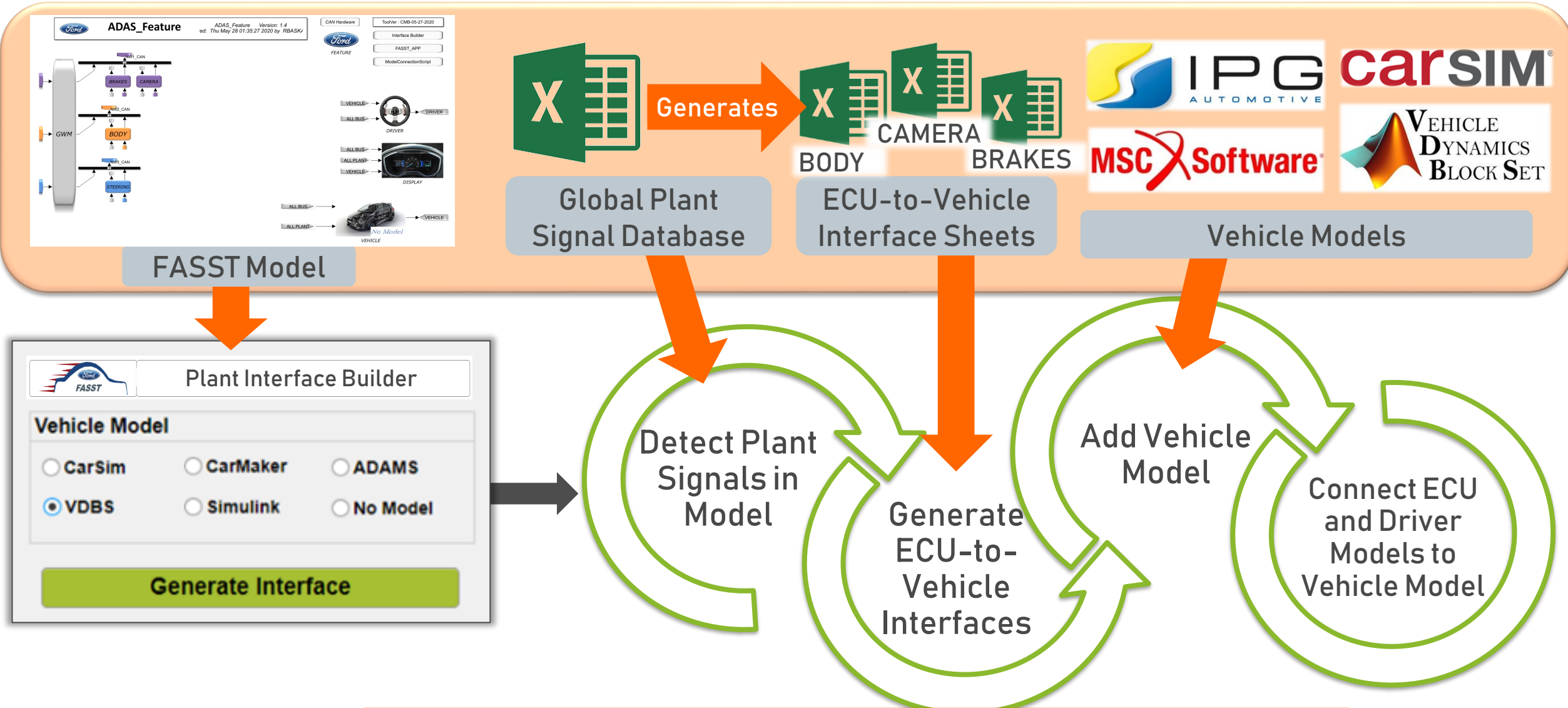
ADAMS



CARSIM

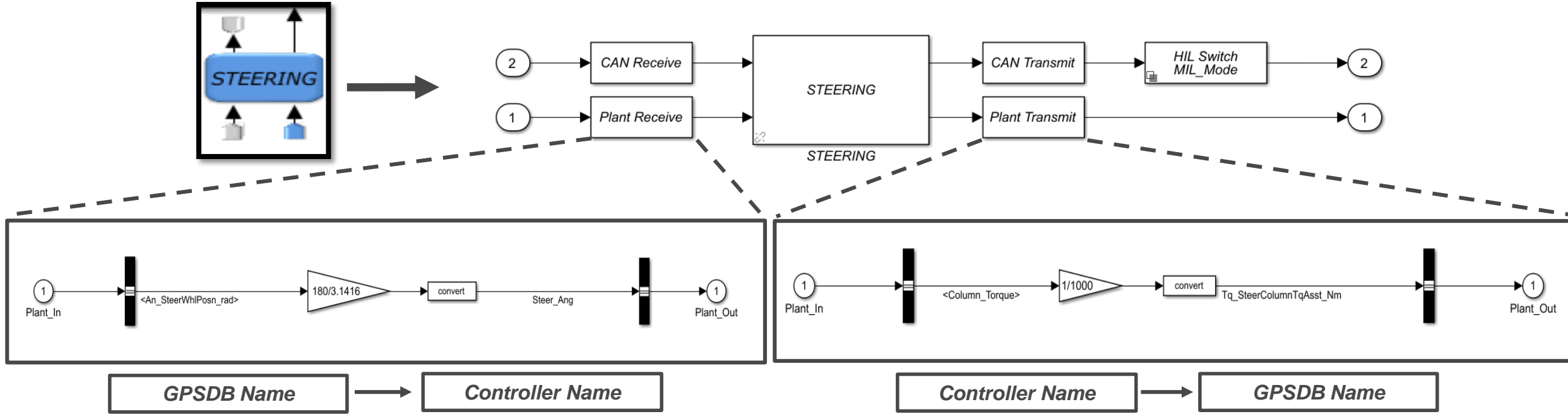
**Vehicle Plant Option Flexibility**

# THE FASST PLANT VEHICLE BUILD PROCESS



The Plant Interface Builder automates the connections within FASST

# THE VEHICLE PLANT TO ECU CONNECTION INTERFACE



STEERING

## ECU-to-Vehicle Interface Sheets

Interface	GPSDB Name	Controller Name	Unit Gain (GPSDB -> Controller)	Unit Gain (Controller -> GPSDB)	VDBS Name	Unit Gain (GPSDB -> VDBS)	Unit Gain (VDBS -> GPSDB)
Receive from Plant	An_SteerWhlPosn_rad	Steer_Ang	$180/\pi$ (rad to deg)	$\pi/180$ (rad to deg)	AngIn	1 (rad to rad)	1 (rad to rad)
Transmit to Plant	Tq_SteerColumnTqAsst_Nm	Column_Torque	1000 (Nm to Nmm)	1/1000 (Nmm to Nm)	TrqIn	1 (Nm to Nm)	1 (Nm to Nm)



Global Plant Signal Database



ECU Models & Components



VEHICLE DYNAMICS BLOCK SET



Go Further

ADAS\_FEATURE - Simulink

SIMULATION | DEBUG | MODELLING | FORMAT | APPS

FILE | LIBRARY | PREPARE | SIMULATE | REVIEW RESULTS

MBO\_Utilities | ADAS\_FEATURE

Hide/Show Explorer Bar | ADAS\_FEATURE

Zoom | Fit to View | Sample Time | Annotation | Image | Area

Viewmark this View | Viewmarks | Hide/Show Model Browser

Ready | View 1 warning | 114% | FixedStepDiscrete | 7:34 AM 6/3/2020

## ADAS\_FEATURE

ADAS\_FEATURE Version: 1.6  
 red: Tue Jun 02 12:27:05 2020 by SFOSTE

FEATURE

# Interface Builder Demo

# SCALING TO PRODUCTION:

UNLEASHING THE POWER OF CONTINUOUS INTEGRATION(CI)



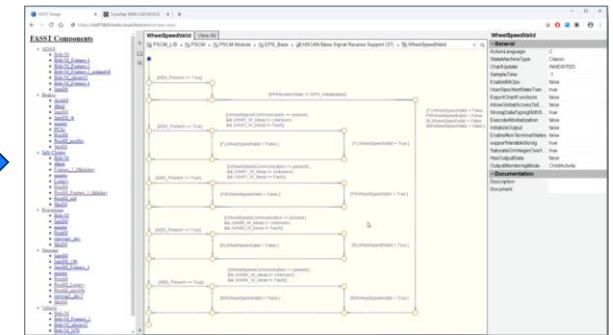
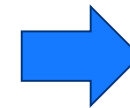
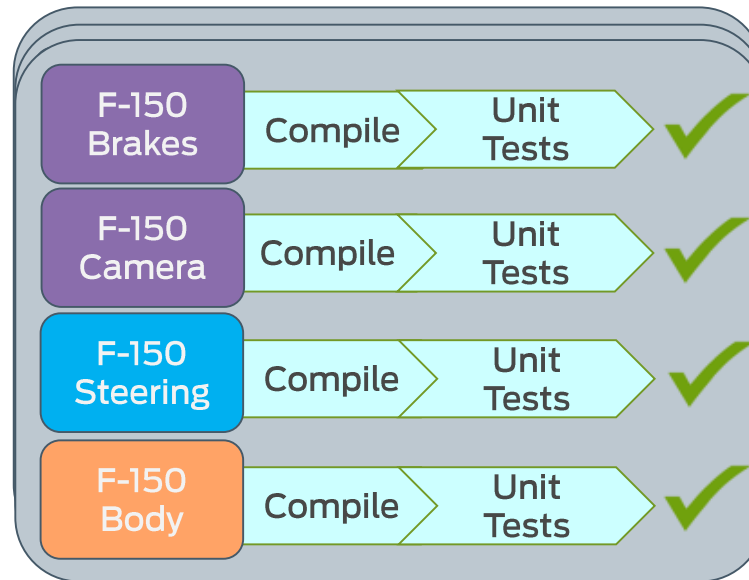
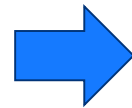
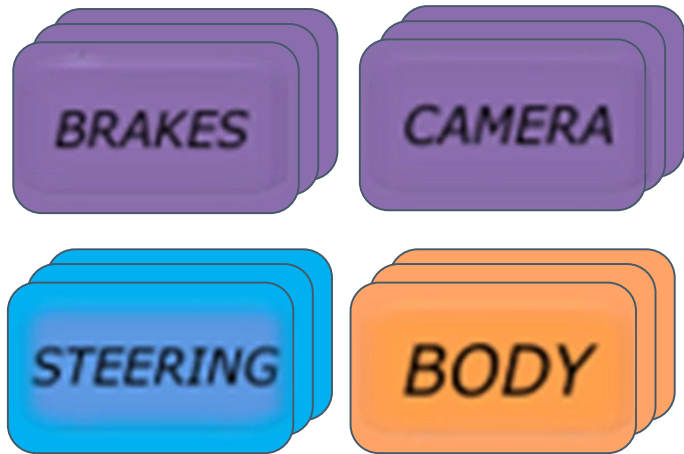
**NICK ADAMS**



# CONTINUOUS INTEGRATION: JENKINS TO VALIDATE COMPONENTS



## Component ECU Repos



250+ tested models are viewable from a browser

At least 10 carlines × 30 ECUs  
Tests run nightly

# THE FASST GARAGE TO BROWSE COMPONENTS

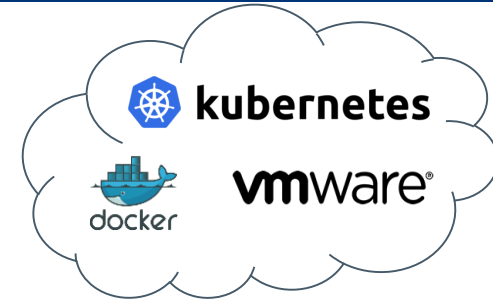
The screenshot displays the FASST Garage web interface. On the left, a tree view titled "FASST Components" lists various modules under categories like ADAS, Brakes, Info Cluster, Powertrain, Steering, and Vehicle. The main area shows a state machine diagram for the "WheelSpeedValid" component. The diagram starts with a state where "[ABS\_Present == True]". Transitions are triggered by "[EPSSystemState == EPS\_Initialization]". The diagram branches into four parallel paths for front-left (FL), front-right (FR), rear-left (RL), and rear-right (RR) wheels. Each path has a state where "[ABS\_Present == True]" and transitions based on "[WheelSpeedCommunication == present]... && (WhiFr\_W\_Meas != Unknown)... && (WhiFr\_W\_Meas != Fault)". The states are labeled "{FLWheelSpeedValid = False;}" and "{FLWheelSpeedValid = True;}" (and similarly for FR, RL, RR). A final state is labeled "{(FLWheelSpeedValid = False; FRWheelSpeedValid = False; RLWheelSpeedValid = False; RRWheelSpeedValid = False;)}".

On the right, the "WheelSpeedValid" configuration panel is shown. It includes a "General" section with properties like ActionLanguage (C), StateMachineType (Classic), ChartUpdate (INHERITED), SampleTime (-1), EnableBitOps (false), UserSpecifiedStateTran... (true), ExportChartFunctions (false), AllowGlobalAccessToE... (false), StrongDataTypingWithS... (true), ExecuteAtInitialization (false), InitializeOutput (false), EnableNonTerminalStates (false), supportVariableSizing (true), SaturateOnIntegerOverf... (true), HasOutputData (false), and OutputMonitoringMode (ChildActivity). Below this is a "Documentation" section with fields for Description and Document.

# CONTINUOUS INTEGRATION: JENKINS TO AUTOMATE VEHICLE BUILDS



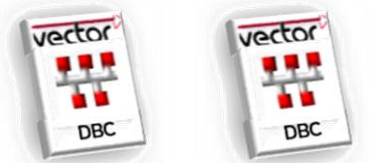
## Jenkins



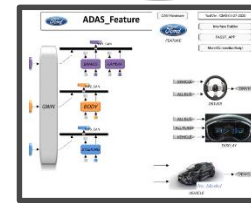
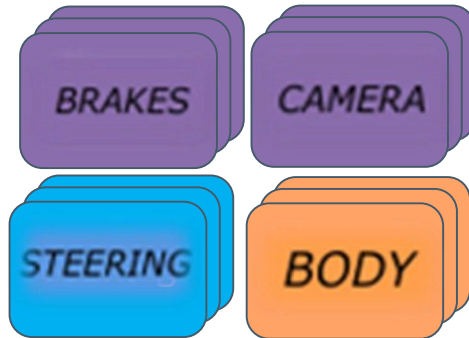
Bill of Models Library



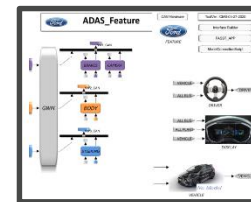
ECU Network (DBC)



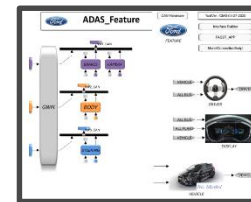
Component ECU Repositories



### F-150 Lane Assist



### F-150 Park Assist



### F-150 Trailer Assist



Dozens of BOMs are built and tested in parallel for a combined total of 40+ hours each day

One pipeline takes between 10 min to over an hour

# INNER SOURCING FOR MODEL DEVELOPMENT



**NATE ROLFES**

**“All Models Are Wrong...  
Some are Useful”**

*- George E.P. Box*

# FASST AND THE POWER OF INNER SOURCING

## Inner Source

is the adoption of **open source software development** best practices and establishment of **open source culture within an enterprise.**

### Collaboration

Maximize the pool of engineering brainpower for advancing a project, meeting user needs, or finding and fixing bugs. **Never start from scratch, always build upon others work!**

### Communication

Transparent, self-documenting, and “searchable” problem solving and decision-making creates trust & alignment in the goals and **makes it easy for new users to get on-board and start contributing!**

### Egalitarian

Users are Developers & Developers are Users leads to a culture void of “politics” as recognition is inherently merit-based. Can work around organizational barriers and **provide everyone the opportunity to influence the project direction and success!**

### The “Plausible Promise”

“Your program can be crude, buggy, incomplete, and poorly documented. What it must not fail to do is (a) run, and (b) convince potential co-developers that it **can be evolved into something really neat in the foreseeable future.**”

*- Eric S. Raymond, The Cathedral and the Bazaar*

**Establishing standard interfaces, terminology, and metrics around model types & capabilities is critical to gain traction for inner source**

# MODEL FIDELITY: SIMPLIFIED AND FUNCTIONAL MODELS

## Skeleton Model

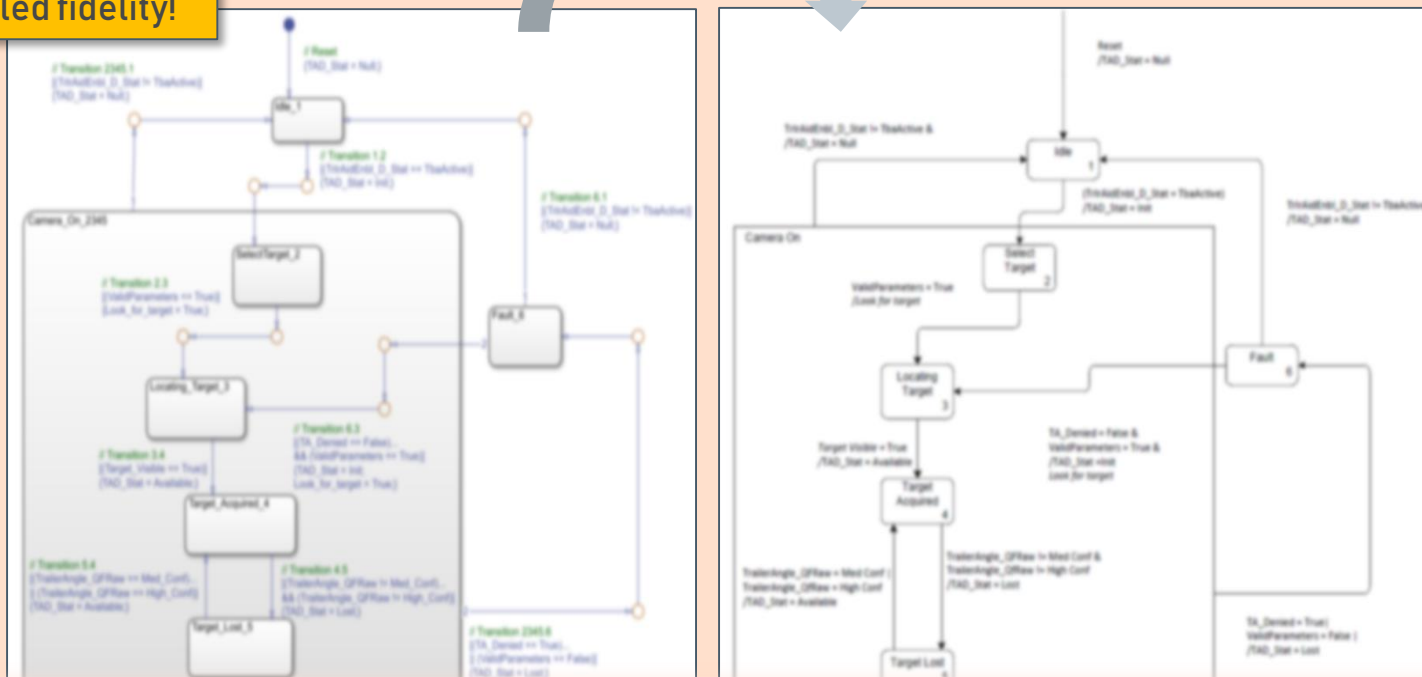
Interfaces (I/O) with Changeable Outputs (No behavior or logic)



The three diagrams are for the same model to demonstrate scaled fidelity!

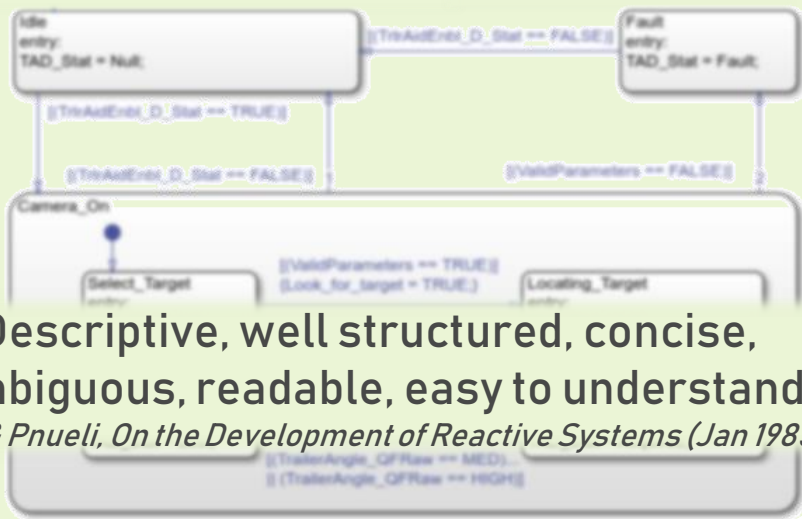
## Requirement Model

Executable representation of functional design requirements and specifications



## Behavior Model

Design Intent Behavior



“Descriptive, well structured, concise, unambiguous, readable, easy to understand.”  
Harel & Pnueli, *On the Development of Reactive Systems* (Jan 1985)

“Capture the functional requirement in a clear and executable manner.”

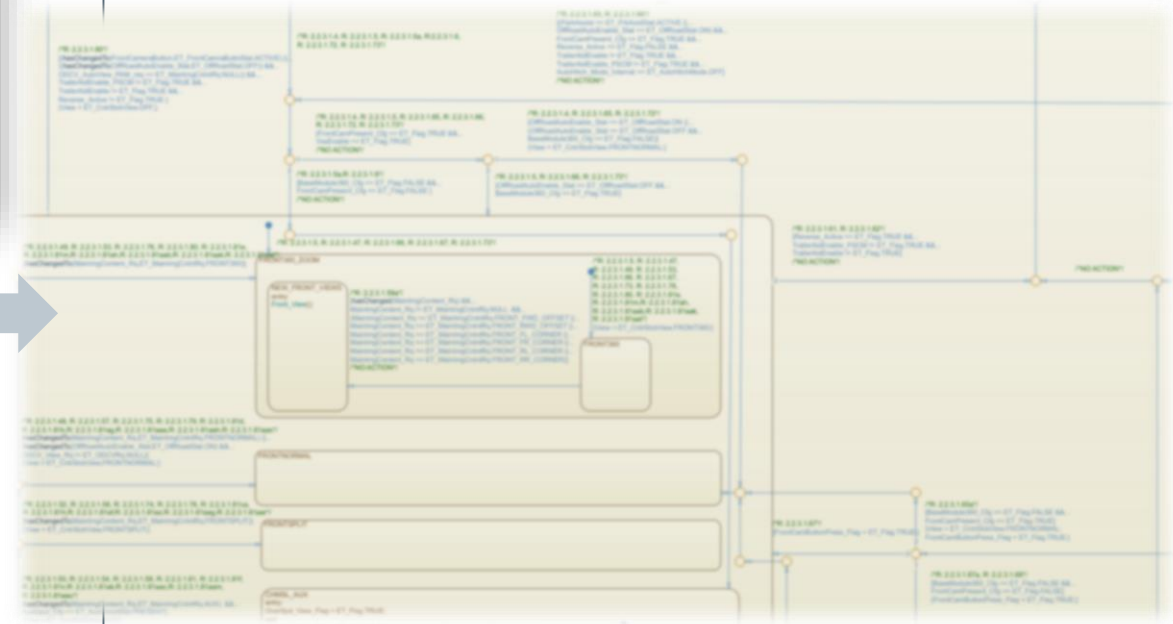
*Lee & Friedman, Requirements Modeling & Automated Requirements-Based Test Generation (2014)*

Evidence indicates that over 60% of System software issues emerge from these three phases!

# MODEL FIDELITY: THE PRODUCTION IMPLEMENTATION MODEL

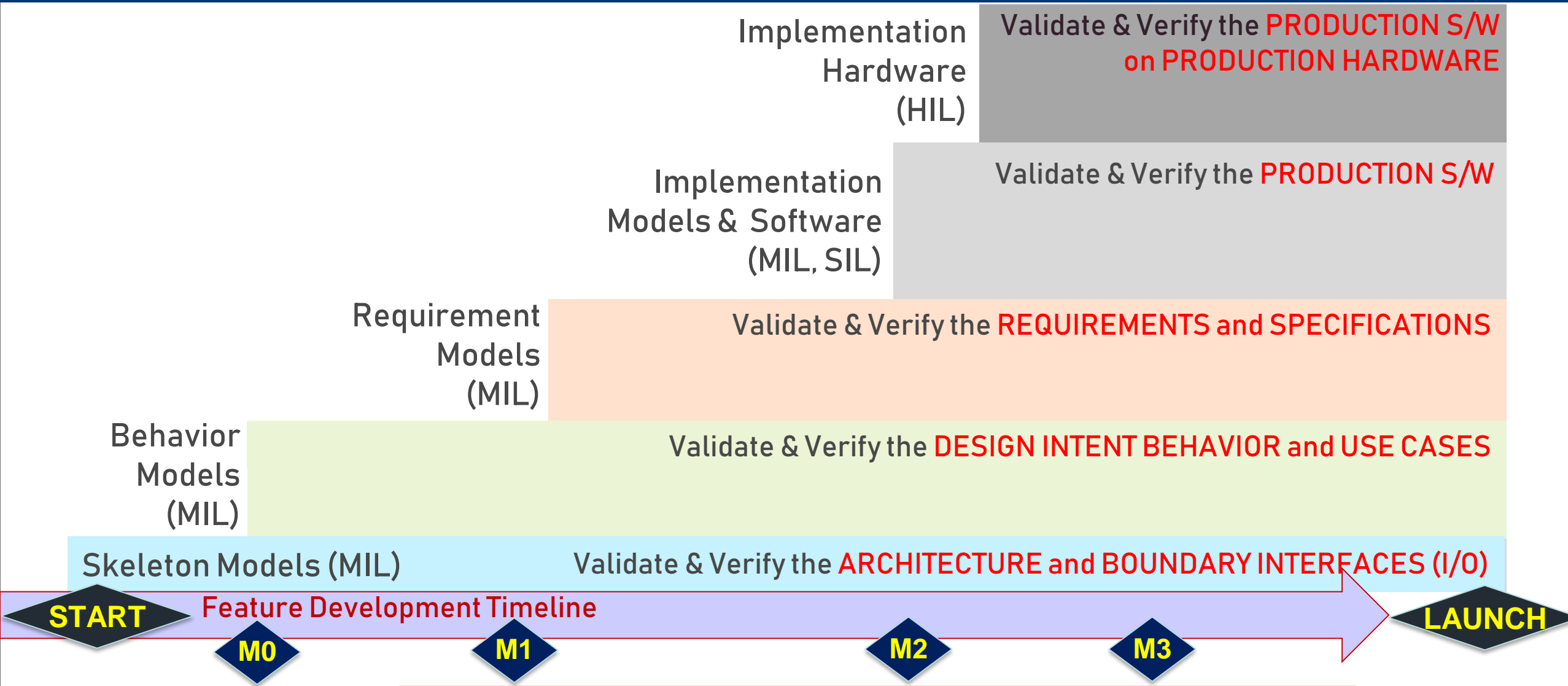
## Implementation Model / Production Code

Model-Based Implementation of the production code model.  
Typically only available for in-house model-based code.  
Difficult to obtain for supplier written code (use HIL instead).



System simulations which utilize Implementation Models can take dozens of minutes to compile and don't simulate at real-time speeds!

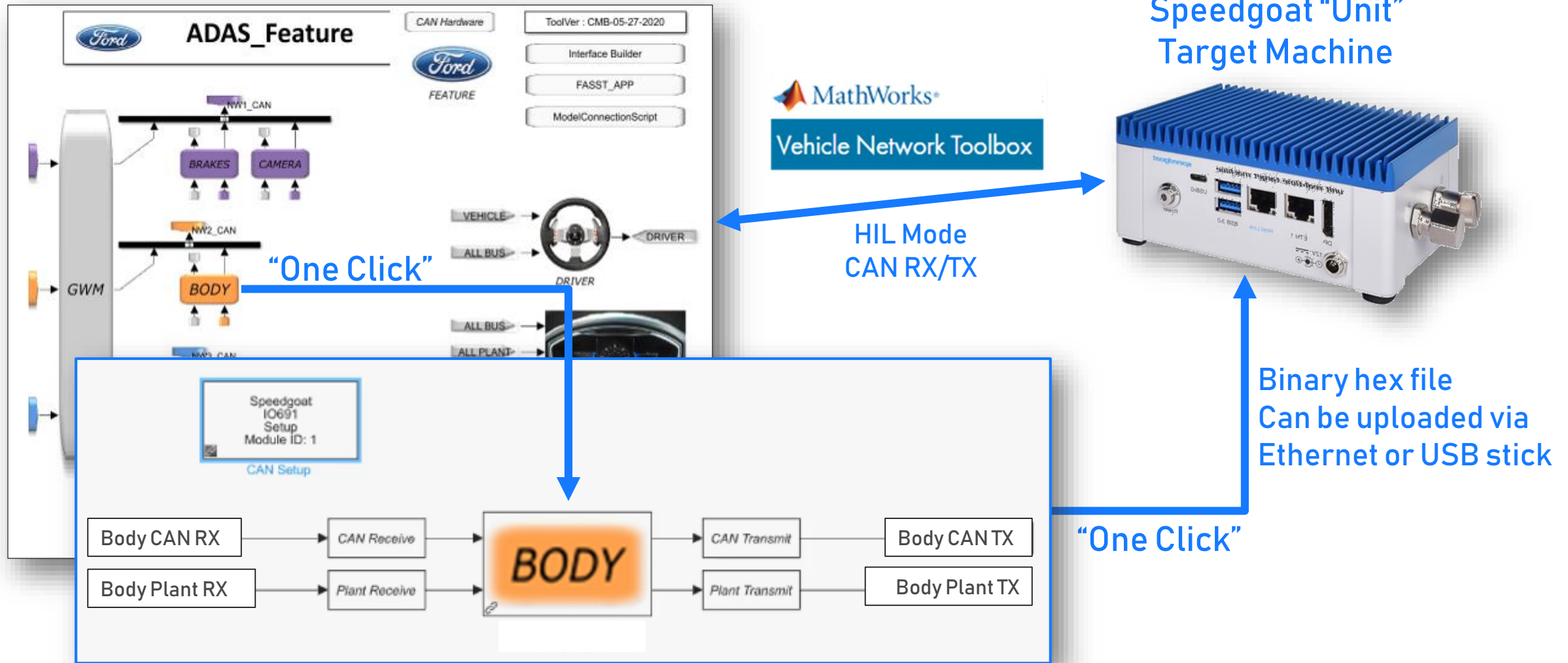
# THE IMPLEMENTATION MODEL CLIFF! vs the MODEL FIDELITY STAIRSTEP



Continuous Modeling Integration using scaled modeling fidelities is critical to ensure the systems work correctly the first time!



# FASST "TO GO": QUICK DEPLOYMENT TO SPEEDGOAT



FASST "To Go" Provides flexible options for offloading model components to real-time hardware as well as an easy method to pass protected binary files to breadboard teams and suppliers

# Summary

- FASST toolchain...
  - ... is developed in collaborative, modern, inner source and agile fashion, together with the MathWorks
  - ... helps to detect system Issues through out the development
  - ... reduced Virtual Vehicle build time from months to minutes
  - ... the automated processes eliminate modeling mistakes
  - ... in combination with CI enables scaling up modeling and simulation to enterprise level.
- The challenge of “*All models are wrong, but some of them are useful*” will always stay
- “Plug and Play” components are a critical key to success

FASST doesn't solve all the issues,  
but makes the daily life of an engineer more effective and enabled  
Cross Organizational Collaboration





THANK YOU

*Ford*

YHV 810