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
"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

For virtual development of distributed systems all teams have to work together.
Simulation
“system” debugging for simulation purpose
Debug Models
(computational issues)
Build Virtual Vehicle
Trace and Collect Component Models

WHERE ARE CAE ENGINEERS SPENDING THERE TIME
Modeling and Simulation
fully integrated into carline development
Debug Models
(computational issues)
Build Virtual Vehicle
Trace and Collect Component Models

Initial State
Desired State

FASST is an enabler to work more effective
FASST (FORD AUTOMATED SYSTEM SIMULATION TOOLCHAIN) leverages modern tools & standards.

**GitHub**
- 40 million+ Global Users
- Ford: 10,000+ Users

**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**

**FASST**
- 500+ Members / Passive Users
- 100+ Active Users
- ~30 Members on “DevOps” Team
FASST: BUILDING THE SKELETON MODEL

Vehicle controls architecture

Bill Of Models

Skeleton/System Model

Ford
Go Further

AUTOMOTIVE CONFERENCE 2020

MathWorks

Vector

HS-CAN 1
HS-CAN 2
FD-CAN
Private-CAN

![Diagram of FASST model building process]

<metadata>

Program_Name: ADAS
Feature_Name: Feature
Variant_Name: Model Parts

<components>

BRAKES: FASST/BRAKES
CAMERA: FASST/CAMERA
BODY: FASST/BODY
STEERING: FASST/STEERING
FEATURE: FASST/FEATURE
DISPLAY: FASST/DATA
DRIVER: FASST/DRIVER

ADAS_Feature

WDW Hardware
Interface Builder
K3SS/APP
Model/Architecture

GWM
BODY
SMART CAN
SMART CAN
DATA CAN
DATA CAN
DATA CAN
DATA CAN
DATA CAN
DATA CAN
DATA CAN
DATA CAN
DATA CAN
DATA CAN
DATA CAN

Veh

DISPLAY

Ford

No Model

Vehicle

Driver
FASST VIRTUAL VEHICLE BUILD: CREATING A VEHICLE MODEL

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

Exploratory testing

Test Plan

Virtual development
Build system/skeleton model & Include components

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
- Model BOM

FASST
- ECU Architecture (DBC)
- Plant Interface Sheet

Vehicle Plant Models
- ECU Models & Components

Cloud-Based Distributed Version Control
Model BOM
ECU Architecture (DBC)
Plant Interface Sheet
ECU Models & Components
Vehicle Plant Models
## IT ALL STARTS WITH THE BILL OF MODELS

<table>
<thead>
<tr>
<th>Program Name</th>
<th>Feature Name</th>
<th>Vehicle Dynamics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variant Name</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Model BOM*

<table>
<thead>
<tr>
<th>Model Parts</th>
<th>GitHub Organization and Repository</th>
<th>Branch/Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRAKES</td>
<td>FASST/BRAKES</td>
<td>BehvM</td>
</tr>
<tr>
<td>CAMERA</td>
<td>FASST/CAMERA</td>
<td>BehvM</td>
</tr>
<tr>
<td>BODY</td>
<td>FASST/BODY</td>
<td>SkeM</td>
</tr>
<tr>
<td>STEERING</td>
<td>FASST/STEERING</td>
<td>ReqlM</td>
</tr>
<tr>
<td>FEATURE</td>
<td>FASST/FEATURE</td>
<td>Feature1</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>FASST/DISPLAY</td>
<td>Display1</td>
</tr>
<tr>
<td>DRIVER</td>
<td>FASST/Driver</td>
<td>Driver1</td>
</tr>
</tbody>
</table>

### Additional sections (not shown) for:
- Build Options
- System Model
- Test Procedures
- Optional Tools
- Documentation
- Test Results
- Miscellaneous

### Network / ECU Architecture

<table>
<thead>
<tr>
<th>Network Parts</th>
<th>GitHub Organization and Repository</th>
<th>Branch/Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ford_PowerTrain</td>
<td>FASST/VDBS</td>
<td>BehvM</td>
</tr>
<tr>
<td>ADAMS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### DBC

<table>
<thead>
<tr>
<th>DBC Organization</th>
<th>Branch/Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>FASST/DBC</td>
<td>Latest_Architecture</td>
</tr>
</tbody>
</table>
FASST is an automation wrapper for several linked tools that builds the full system vehicle model from components in the GitHub cloud.
ANATOMY OF A FASST 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

Version: 1.4
ed: Thu May 28 01:35:27 2020 by RBASK

Virtual ECU
“Breadboard”
MIL, SIL, & HIL
Ready

Driver Model
Signal Monitors Displays
Vehicle Model
Shortcut Buttons to FASST Tools
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

Vehicle Plant Option Flexibility

VEHICLE PLANT MODEL FLEXIBILITY IS CRITICAL
THE FASST PLANT VEHICLE BUILD PROCESS

The Plant Interface Builder automates the connections within FASST.
SCALING TO PRODUCTION:
UNLEASHING THE POWER OF CONTINUOUS INTEGRATION (CI)

NICK ADAMS
CONTINUOUS INTEGRATION: JENKINS TO VALIDATE COMPONENTS

Component ECU Repos

GitHub

Jenkins

FASST Garage

Component ECU Repos

- BRAKES
- CAMERA
- STEERING
- BODY

流程图示意图展示了不同ECU的编译和单元测试过程。

- F-150 Brakes
- F-150 Camera
- F-150 Steering
- F-150 Body

测试结果显示

- Compile → Unit Tests

至少10款车型 × 30款ECU

测试每天运行

250+ 测试模型均可通过浏览器查看

AUTOMOTIVE CONFERENCE 2020
FASST Garage is the enabler for finding the right model based on the use case and user’s needs.
CONTINUOUS INTEGRATION: JENKINS TO AUTOMATE VEHICLE BUILDS

ECU Network (DBC)
Component ECU Repositories
-

Bill of Models Library

GitHub

Jenkins

F-150 Lane Assist
FASST → Compile → Unit Tests → Feature Tests

F-150 Park Assist
FASST → Compile → Unit Tests → Feature Tests

F-150 Trailer Assist
FASST → Compile → Unit Tests → Feature Tests

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

AUTOMOTIVE CONFERENCE 2020
INNER SOURCING
FOR MODEL DEVELOPMENT

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

- George E.P. Box

NATE ROLFES
WHAT IS INNER SOURCE?

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

**Skeleton Model**
- Interfaces (I/O) with Changeable Outputs (No behavior or logic)

**Behavior Model**
- Design Intent Behavior
  - “Descriptive, well structured, concise, unambiguous, readable, easy to understand.”

**Requirement Model**
- Executable representation of functional design requirements and specifications
  - “Capture the functional requirement in a clear and executable manner.”

**MODEL FIDELITY: SIMPLIFIED AND FUNCTIONAL MODELS**

The three diagrams are for the same model to demonstrate scaled fidelity!
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!
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 throughout 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
With special thanks to the entire FASST team: