Taking advantage of plant modelling in the software development process

David Stamm
Chief Engineer – Systems
david.stamm@pi-innovo.com

Pi Innovo

Vehicle electronics innovators

The Problem

- Significant improvements in the quality and availability modeling and simulation resources, but are infrequently utilized by software developers
- Modeling and simulation is viewed as separate activities to the controls development process
- Software development lifecycles are long. Fixing design errors based on field failure data is prohibitively expensive
- Vehicle testing is difficult to schedule and is also very expensive

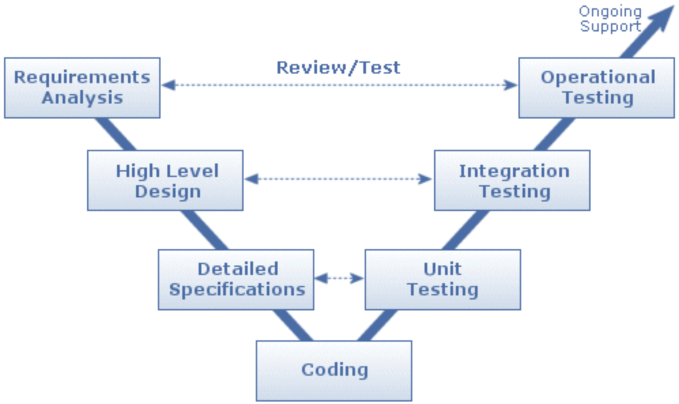
Constraints

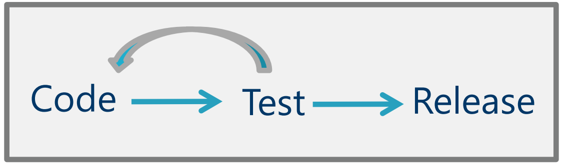
- To further constrain the broader adoption
  - Organizational structure
  - Internal & External budgets
  - Software development lifecycle model
  - Knowledge base for physics based modeling
  - Tools and co-simulation platform availability
  - Finding the right level of fidelity in the model
  - Calibrating and correlating the model

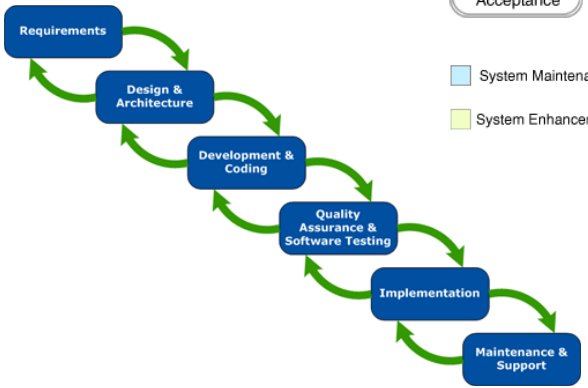
Realities

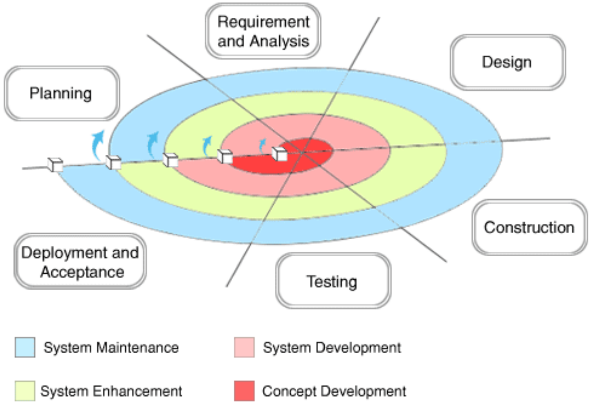
- Trust issues. Is the plant correct? Is it well correlated?
- Separate controls and simulation groups. No data sharing. Engineering silos
- People model a system larger than needed at too high of a fidelity
- The simulations take too long to run and the controls engineers can't be bothered to wait minutes for a single run to complete.
- Some evidence of using simulation, but it isn't fully integrated into the controls development process.

Common Software Development Lifecycles







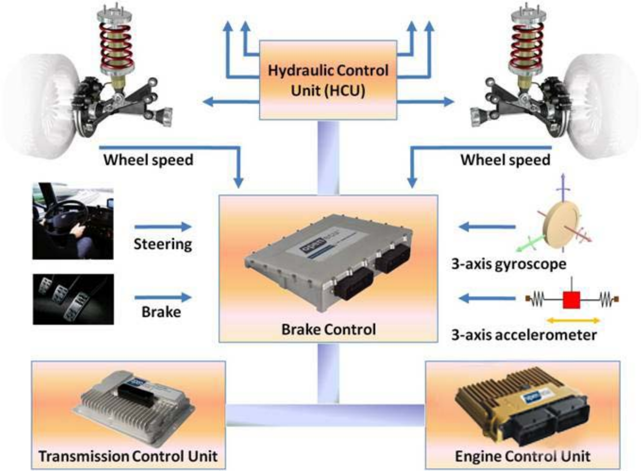


Case Study Details

- For the purpose of the discussion here all projects were conducted under a CMMI Level3 compliant development process.
- The software development lifecycles were tailored to support modeling and simulation during the software design, implementation, and software testing stages.
- Project subject matter in case studies varied from an active suspension system, vehicle rollover warning system, and a ABS/TCS/ESC system
- Each project used varying levels of fidelity in their plant models.

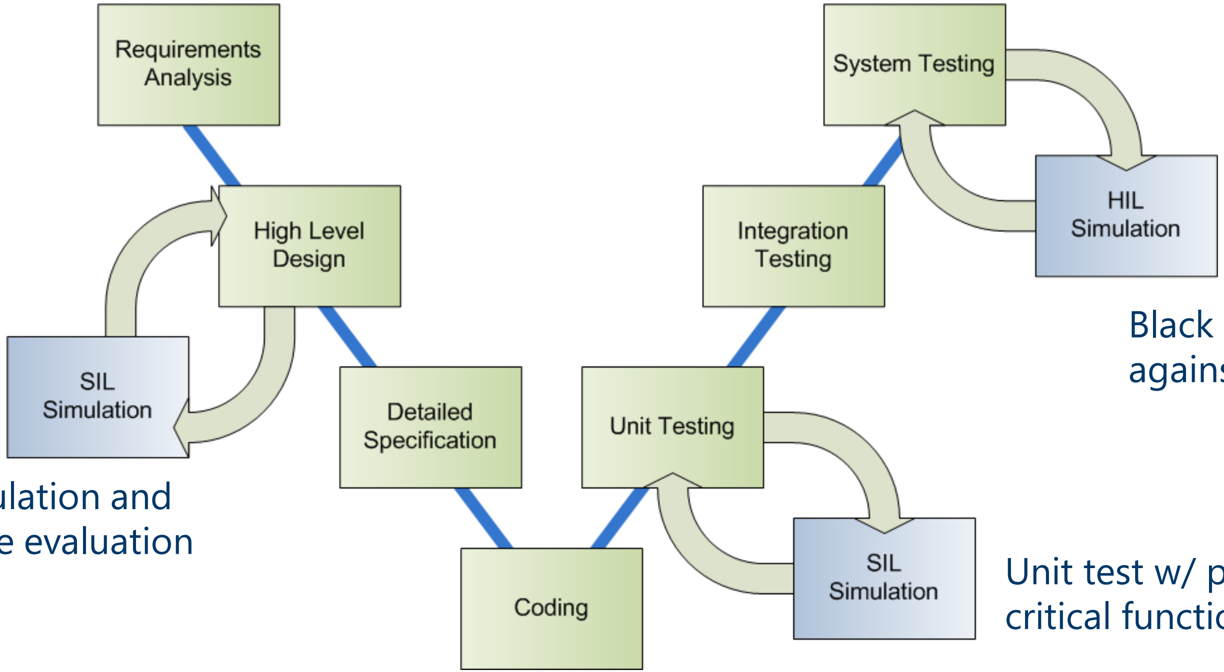
Case One – ABS/TCS/ESC System

- Customer contracted Pi Innovo to develop the controls for a combined ABS, Traction Control (TCS), and Electronic Stability Control (ESC) system.
- Minimal existing intellectual property was re-used. Clean sheet design.



Modified V-Model

Release to vehicle test



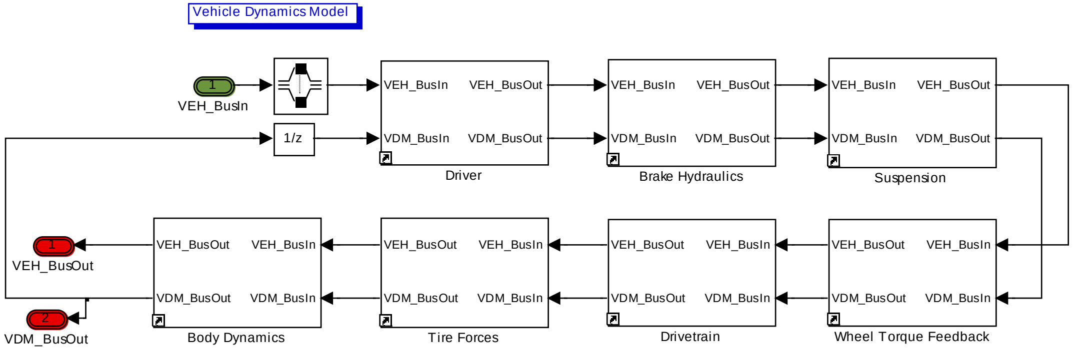
Black box testing against use cases

Unit test w/ plant for critical functions

Ad hoc simulation and performance evaluation

Case One – Test Configuration SIL

- Software-in-the-loop (SIL) testing conducted during design phase to support robust requirements definition of the software features.
- Re-utilized SIL testing during change management.
- Closed loop simulation in PC environment. Quasi real-time.
- Identical plant model to HIL simulation

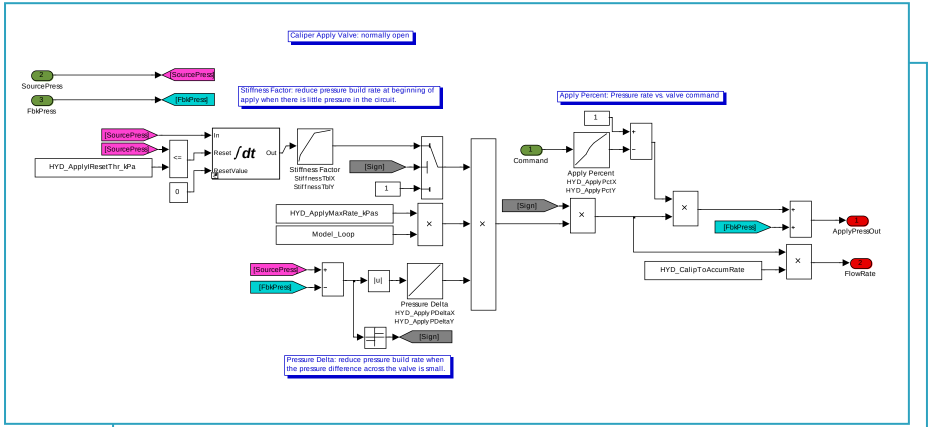


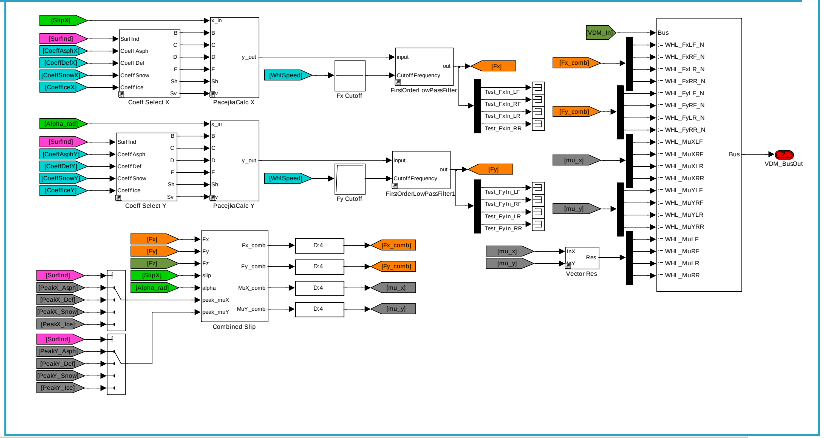
Case One – Test Configuration HIL

- Autosim Hardware-in-the-Loop (HIL) system.
- Physics based plant model (Simulink) running on HIL
- Control logic running on target ECU
- Closed loop, real-time simulation
- Option: add actual brake hydraulic components in the loop

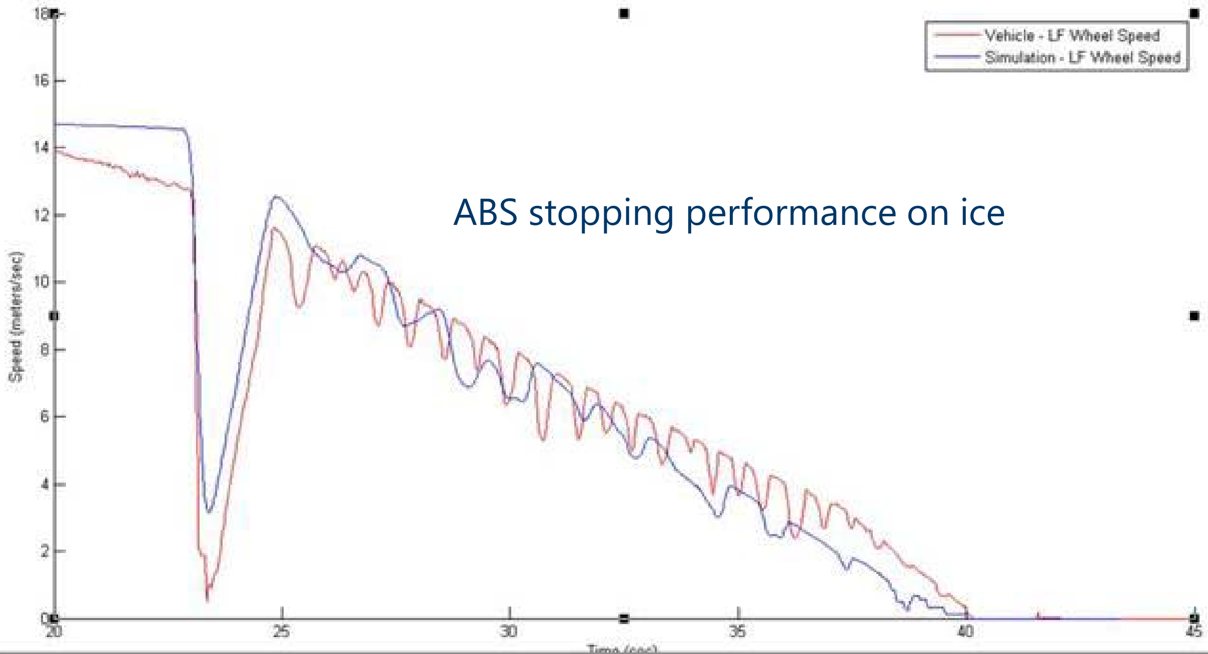
Case One – Plant Model

- Considerations
  - Driver steering effects
  - Brake hydraulics:
    - pump flow, valve flow, tank flow
    - accumulator pressure
    - brake caliper volume, stiffness, valve dynamics
  - Suspension dynamics: weight transfer due to pitch and roll
  - Drivetrain efficiency, inertia, and load
  - Pacejka Tire model
  - Vehicle Dynamics: Yaw, slip angles, roll, pitch
  - Pseudo random sensor noise





Simulation Output Results



ABS stopping performance on ice

Case One – Plant Model

- Certain dynamics were not considered
  - Fidelity requirements were geared to support basic controls software development
- Specific exclusions were:
  - Temperature & wear effects
  - Component variation
  - Caliper and valve friction
  - Brake pad friction vs. time vs. temperature
  - Internal leakage of hydraulics
  - Rough road models
  - Suspension stiffness & damping modeled with simple gains and low pass filters to provide overall bulk response to pitch and roll.

Case One – Medium Fidelity Plant Model

Pro

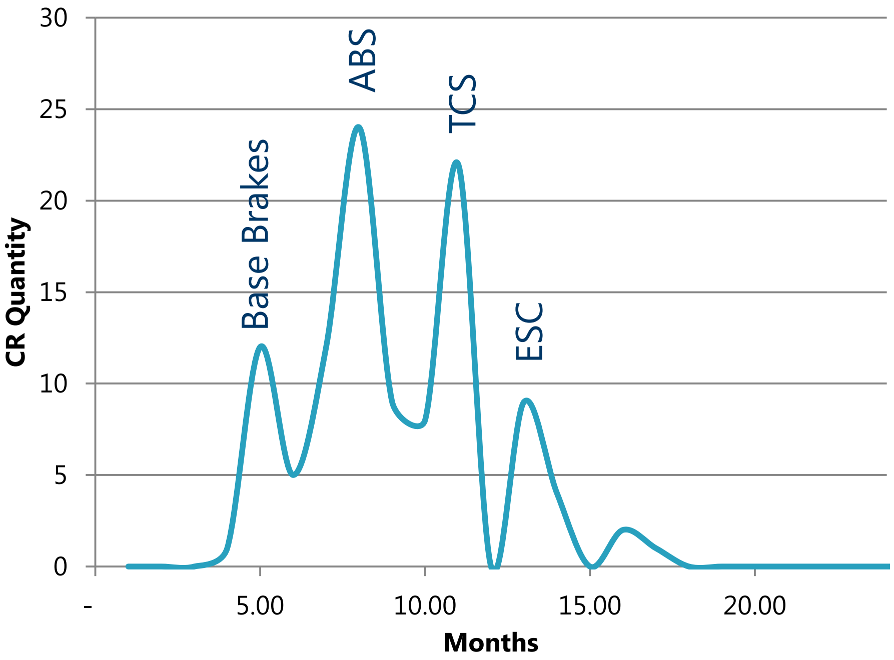
- Validate earlier in the development process
- Many control options evaluated in a short period of time
- Fits the traditional work habits of most software developers. Code first, ask questions (and provide documentation) later
- Baseline calibration more representative
- Eliminates the obvious design blunders
- Reduces the occurrence of controls design changes coming from field testing

Con

- Debugging often involves plant and control.
- Correlation to real world difficult to do if you are modeling a new system
- More time spent analyzing data over low fidelity output
- Requires more computational power to ensure plant runs at correct rate compared to controls
- Must augment SW development lifecycle to properly account for simulation

Case One – Project Metrics

- 1.5 years duration
- 25,000 man-hours
  - includes controls development and extensive vehicle testing
- 109 total Change Requests
- 144,000 total lines, 70,000 LOC
- Unit test in 10-60 min
- Black Box regression testing runtime of 3hrs
- First time pass of FMVSS126 test

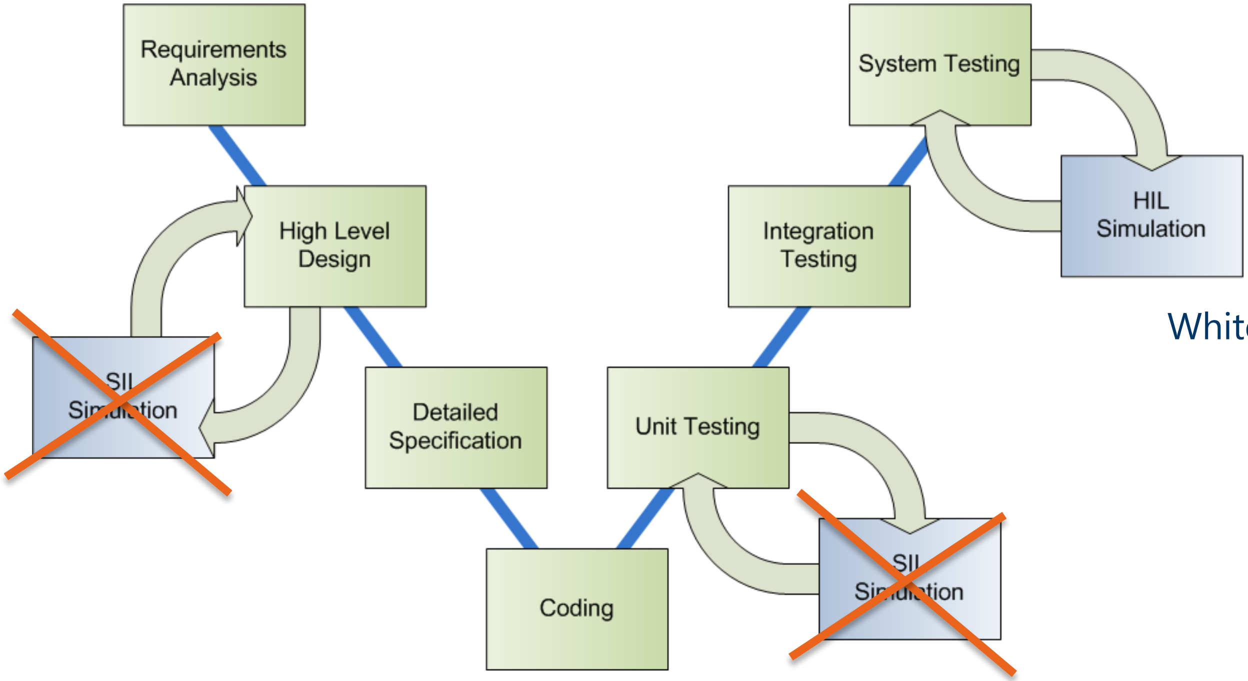


Case Two – Active Suspension System

- Customer contracted Pi Innovo to develop a control system for a novel active suspension system.
- Similar in scope and size to Case One.
- Clean sheet design. No prior intellectual property.
- New suspension technology for active components, minimal prior published work on the subject.

Case Two - Modified V-Model

Release to vehicle test



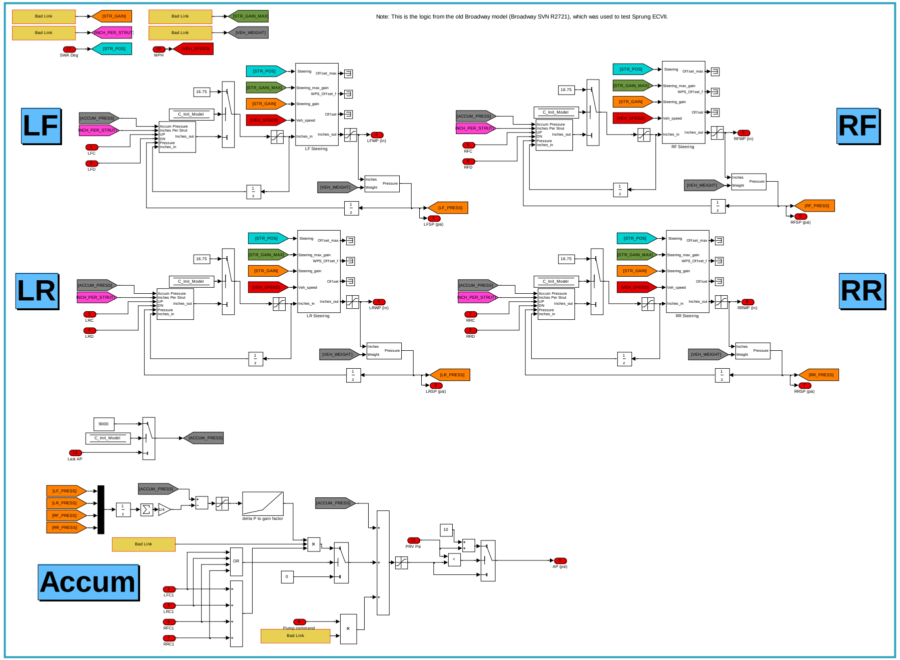
White Box Testing

Case Two – Test Configuration HIL

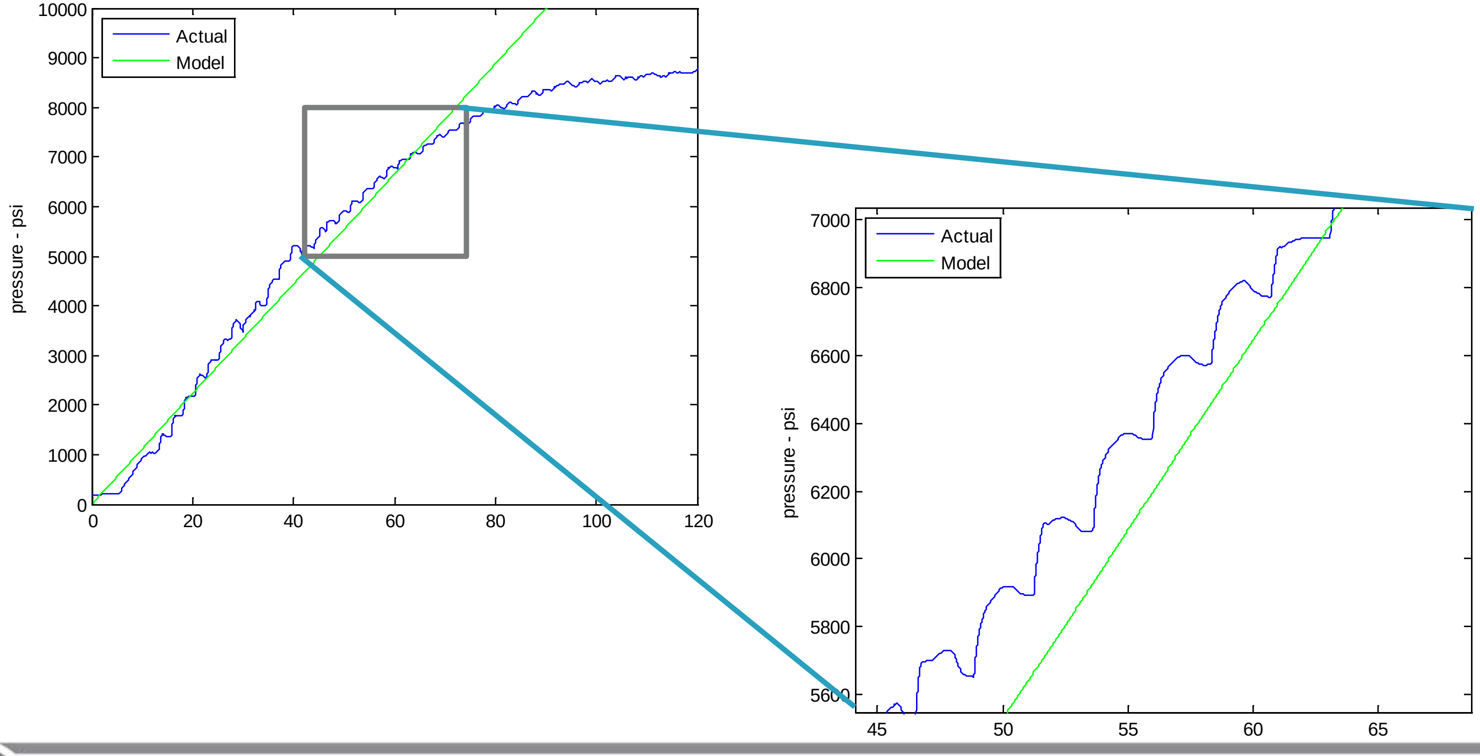
- Autosim Hardware-in-the-Loop (HIL) system.
- Low fidelity non-physics based plant model (Simulink) running on HIL
- Control logic running on target ECU
- Closed loop real-time simulation

Case Two – Plant Model

- No vehicle dynamics
- Hydraulics & vehicle statics focus. Non-physics based.
- *'for every millisecond the valve is open, increment pressure by 'x' kPa'*
- *'for every millisecond the valve is open, increment ride height by 'y' mm'*



Simulation Output Results



Case Two – Low Fidelity Plant Model

Pro

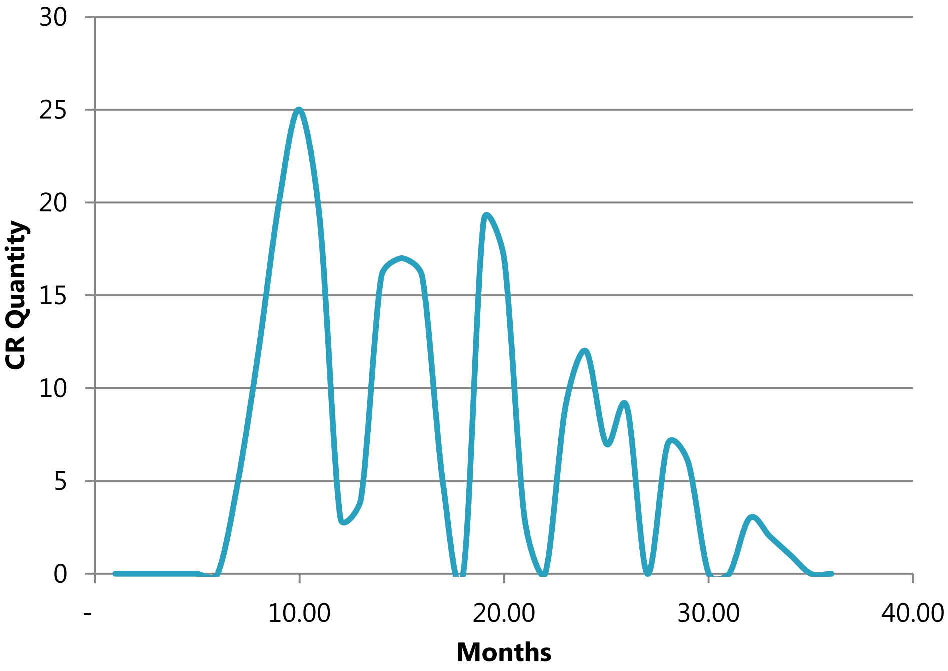
- Effort to develop is low
- Effort to validate/correlate is low
- Computationally simple / low overhead
- Evaluate bulk response of controls

Con

- Does not fully validate the design
- Does not provide a baseline calibration
- Effort to test with low fidelity model same as medium fidelity
- Costly. Can not validate design until post-release vehicle testing.

Case Two – Project Metrics

- 3 years duration
- 30,000 man-hours
  - includes controls development extensive vehicle testing
- 237 Change Requests
- 92,000 total lines, 46,000 LOC
- White Box regression testing runtime of 15hrs



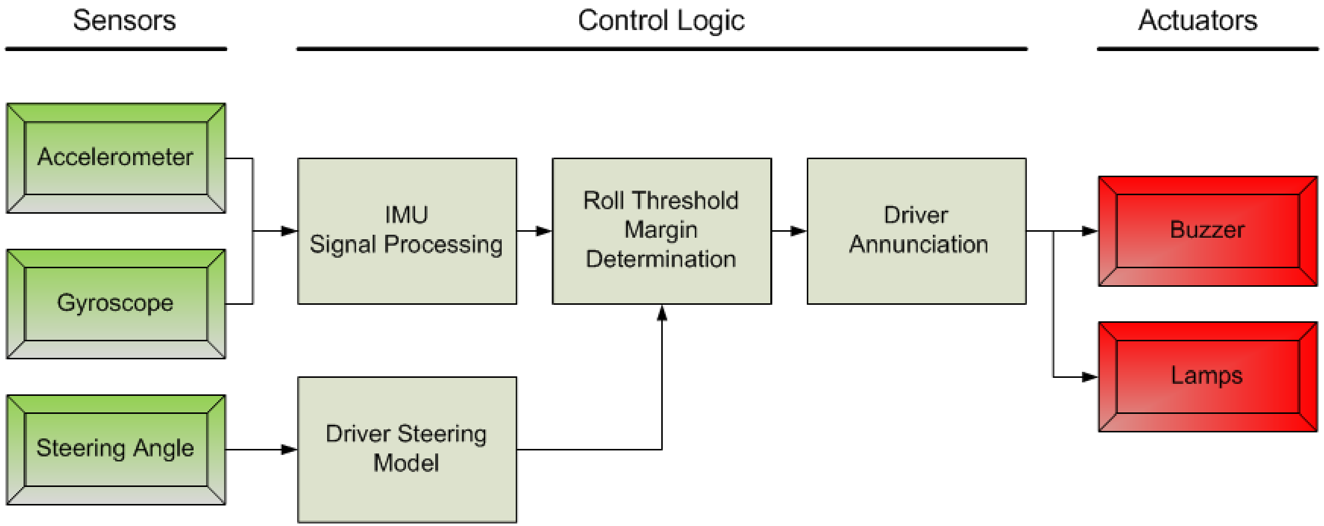
Case Three – Rollover Warning System

- No HIL test environment
- No SIL test environment
- Vehicle testing only

- Proof of concept prototype / technology demonstrator
- Show the effectiveness of warning drivers about impending rollover. Using accelerometer and gyroscope data, combined with a predictive element based on driver steering input as rollover prediction was made and annunciated to the driver.

Case Three – Rollover Warning System

- Driver steering and vehicle attitude model
- Simple threshold triggering for annunciation



Case Three - No Plant Model

- Suitable for feasibility studies or proof of concept projects
- Not recommended for safety critical systems of any kind
- No means to evaluate the design prior to release to vehicle test
- No ability to pre-calibrate the controls prior to release
- Only suitable if the actual test property is available to the software developers
- Only suitable to programs with ample time and budget

Case Three – Software Revision Metrics

- There are no software metrics
- Work commenced on Nov 4th with only high level user needs statement
- Functional prototype delivered Nov 17th

- 43 software versions over 13 days
- Desktop and vehicle testing only

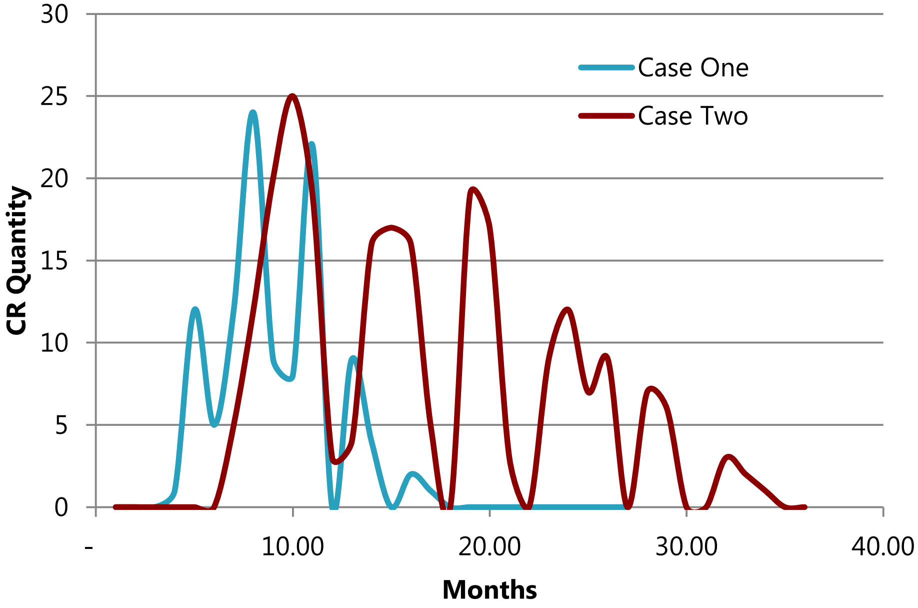
- Successful customer demonstration Nov 18th in vehicle.

Summary Questions

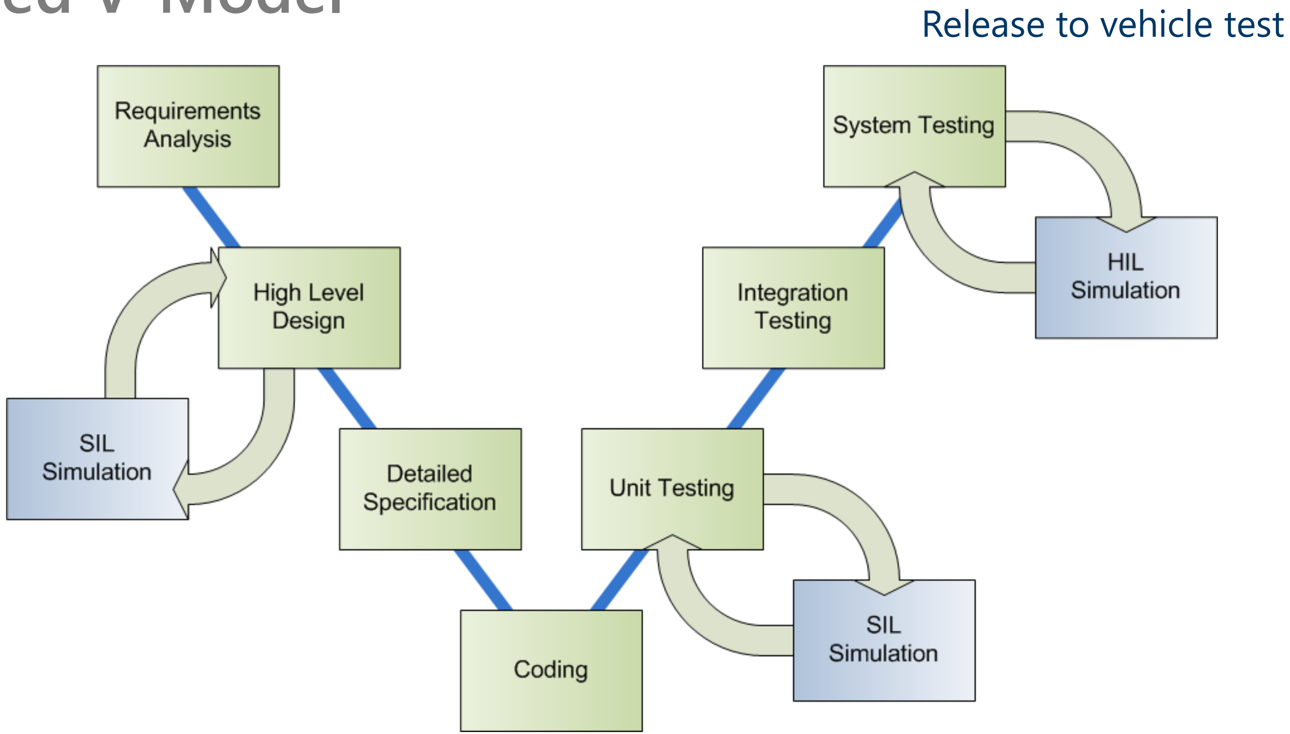
- Why was success variable between these examples?
- Was simulation the only differentiating factor in these examples?
- Is there a place for low fidelity non-physics based plant models?
- When should you choose to use physics based models in software development?
- What were the experiences of the project engineers and their thoughts?

Summary

- Case one was more efficient and effective than Case two
- Longer cycles between design validation (in-vehicle testing)
- 2x number of CRs
- 2x duration



Modified V-Model



Summary

- Case one was able to achieve high first time quality and pass federal compliance tests on the first try
- Case two project duration (and cost) was negatively impacted by not being able to validate the design choices early in development
- Case two costs were ~25% greater than Case one
- Case two duration was 2x longer than Case one
- Low fidelity, non-physics based models increase costs over medium fidelity physics based model when used in software development **
- Not every controls project needs a plant model – check your goals

Pi Innovo

Vehicle electronics innovators

Thank you for your attention.

Pi Innovo LLC
47023 W. Five Mile Road
Plymouth MI 48170-3765
United States of America

+1 734 656 0140

pi-innovo.com

Pi Innovo is the expert partner for the design and development of innovative electronics systems to the automotive, transportation, defense, industrial, and aviation industries. Our uniquely adaptable business engagements, based on Pi Team services and OpenECU products, enjoy a strong reputation for delivering results of the highest quality, providing outstanding value for our customers.

Pi Team is the provision and management of multi-skilled teams that can perform any or all of the work required to develop a vehicle electronics system from concept to production. Pi Teams are built from well rounded engineering and commercial staff that have the expertise to work to the highest quality standards and the motivation to innovate by working collaboratively. Key to Pi Team success is the availability of both system engineering specialists and design engineers skills.

OpenECU is a wide range of adaptable, field-ready products and intellectual property designed to accelerate electronics system development. The philosophy behind OpenECU is the creation of modular, reusable technology that is implemented to volume production standards and is fully "open" to custom configuration, adaption and further development.