Testing automotive software

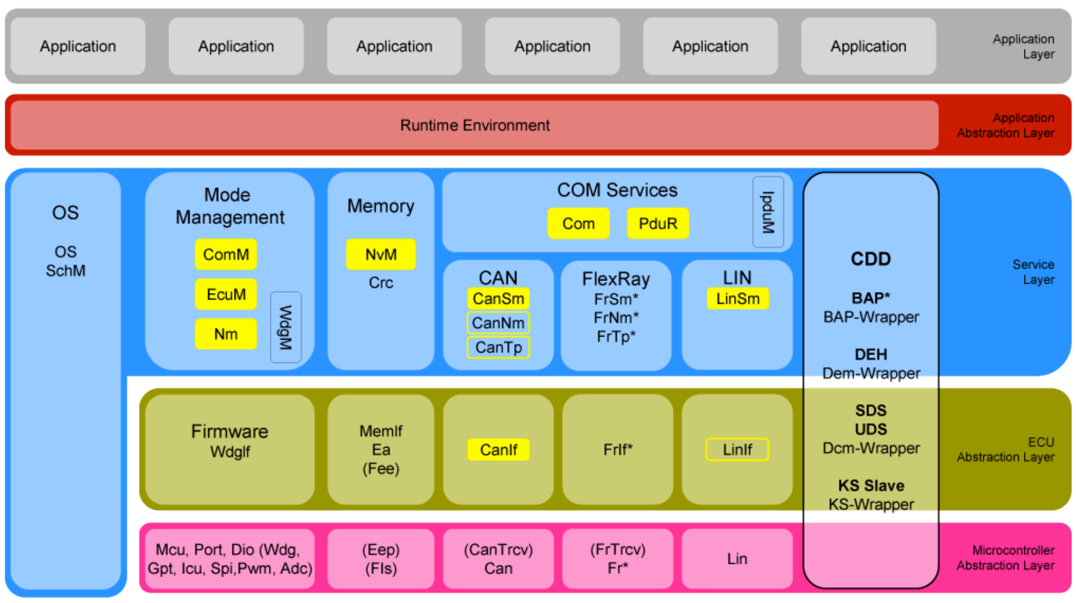
Ulf Norell

RawFP Meeting May 20, 2011

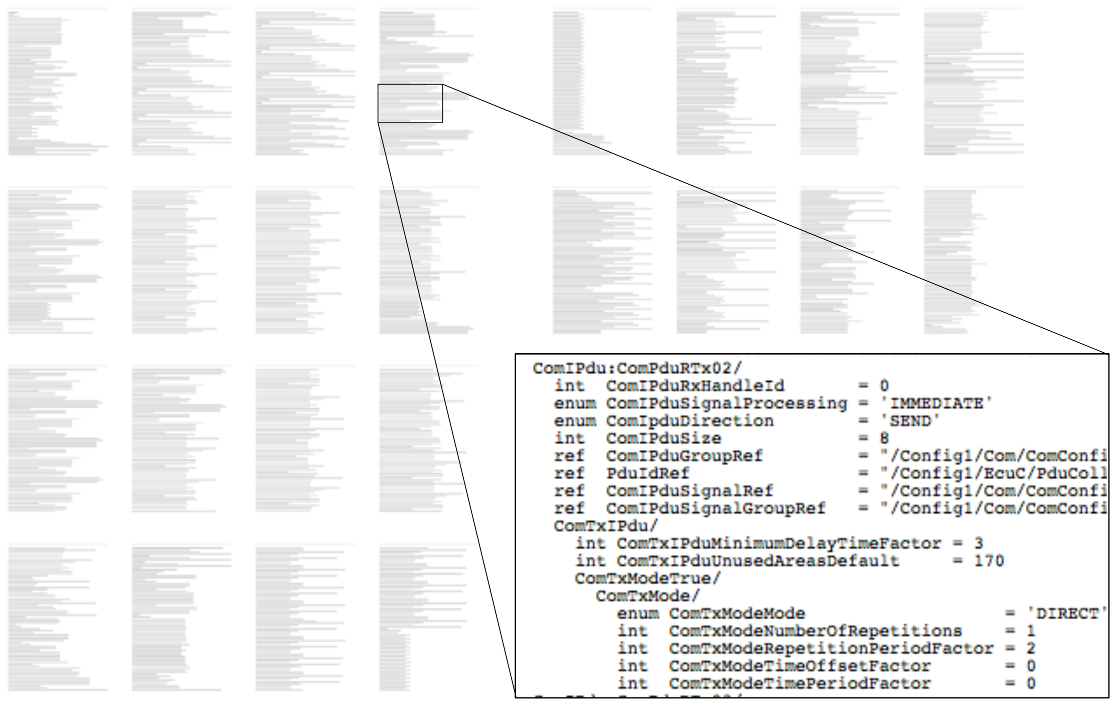
What is AUTOSAR?

- Standard for the operating system running in your car.
- A modern car contains lots of computers that need to communicate with each other.
- Much of the standard defines the numerous network protocols used.

(Some) AUTOSAR components



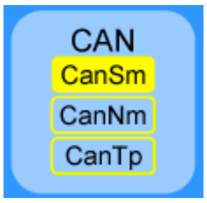
Highly configurable



```
ComIPdu:ComPduRTx02/
  int  ComIPduRxHandleId         = 0
  enum ComIPduSignalProcessing = 'IMMEDIATE'
  enum ComIpduDirection          = 'SEND'
  int  ComIPduSize               = 8
  ref  ComIPduGroupRef           = "/Config1/Com/ComConfi
  ref  PduIdRef                  = "/Config1/EcuC/PduColl
  ref  ComIPduSignalRef          = "/Config1/Com/ComConfi
  ref  ComIPduSignalGroupRef     = "/Config1/Com/ComConfi
  ComTxIPdu/
    int ComTxIPduMinimumDelayTimeFactor = 3
    int ComTxIPduUnusedAreasDefault     = 170
    ComTxModeTrue/
      ComTxMode/
        enum ComTxModeMode                = 'DIRECT'
        int  ComTxModeNumberOfRepetitions = 1
        int  ComTxModeRepetitionPeriodFactor = 2
        int  ComTxModeTimeOffsetFactor     = 0
        int  ComTxModeTimePeriodFactor     = 0
```

Clusters

- If a developer is implementing several components they may choose to violate internal interfaces
- Sometimes this might even be necessary to get acceptable performance
- Testing nightmare!



Clusters

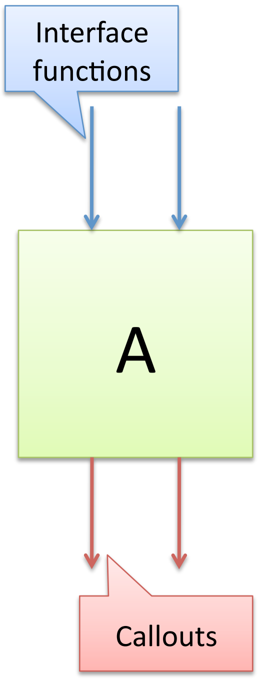
- Solution 1: Write a test model for the cluster
- Problems:
  - A different developer might not have implemented all components in the cluster
  - Hard figure out what the specification is
  - Models get big and clunky

Clusters

- Solution 2: Write composable models for the components
- Problems:
  – QuickCheck state machine models aren't composable

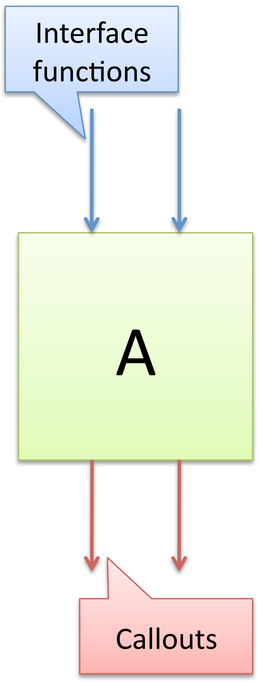
Testing a single component

- Test case: sequence of commands
  – a call to an interface function
  – return values for the callouts
- The model
  – keeps track of the state of the system
  – predicts which callouts will be called



Testing a single component

- Running a test: for each command
  - tell the callout functions what to return
  - call the interface function
  - check that the callouts were called with the right arguments
  - check that the function returned the right result



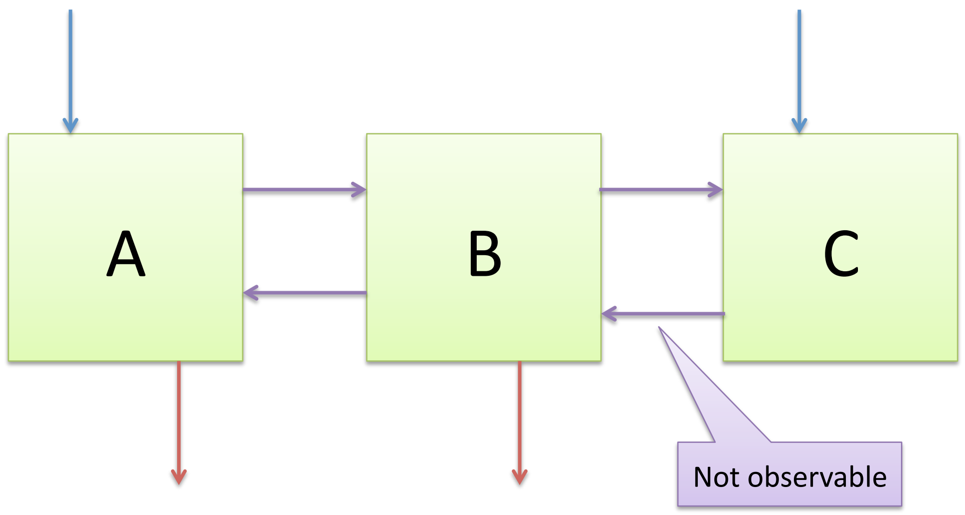
Component model

- callouts(Call) -> Callouts
- precondition(Call, Callouts) -> Bool
- next_state(S, Res, Call, Callouts) -> S
- postcondition(S, Call, Res, Callouts) -> Bool

Generators for results

Actual results

Testing a cluster



Return value callback

- We need return values for internal function calls
  - return_value(S, Call, Callouts) -> Result

Specifying a cluster

- components() -> list(Component)
- classify_callout(Call) ->
  external | {internal, Component}
- interface_functions() -> list(Call)

Current state

- I've got this running on a toy example
- I'm currently adapting one of our AUTOSAR cluster models with encouraging results