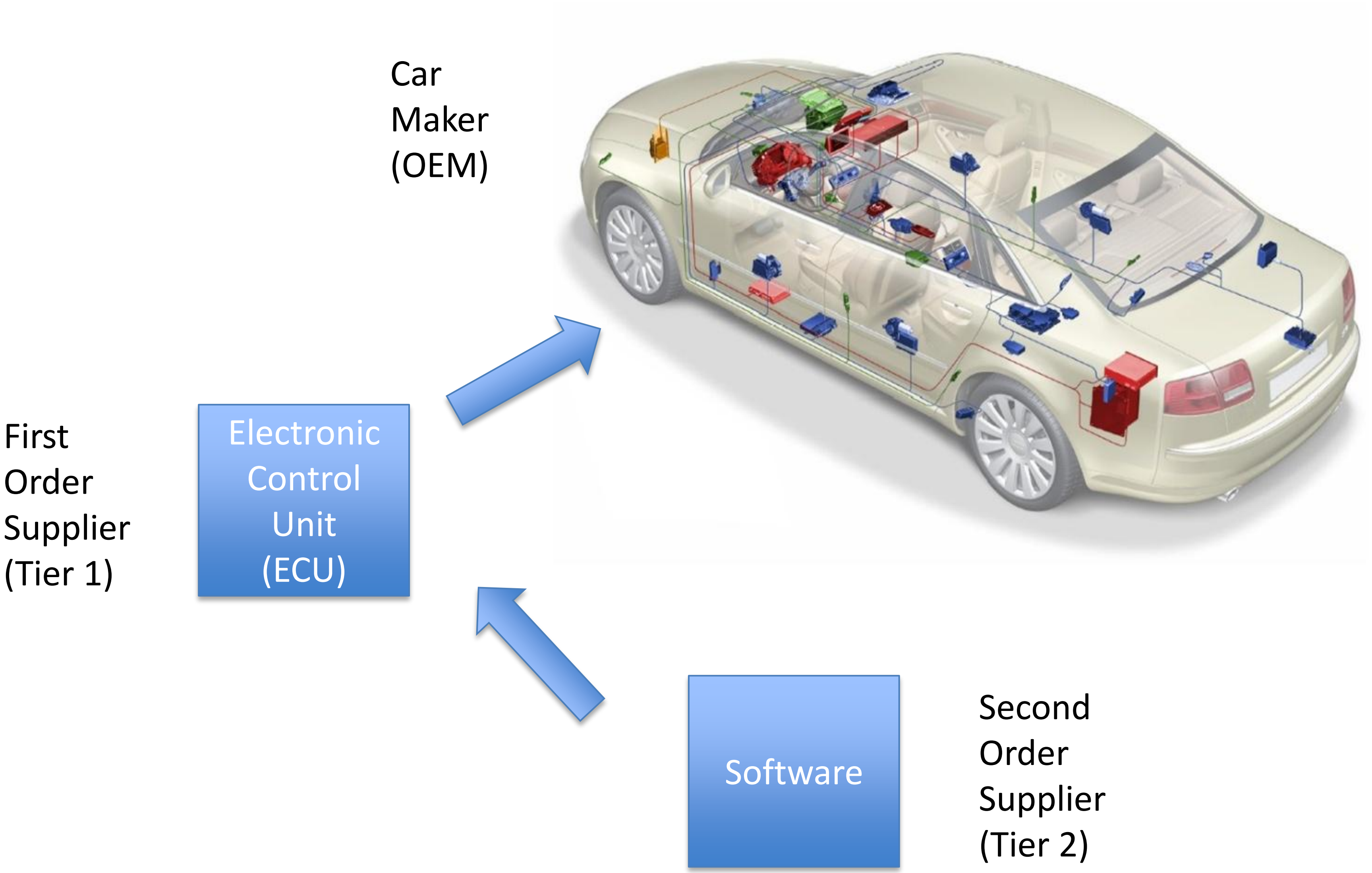
Testing AUTOSAR components with QuickCheck

Thomas Arts

Quviq / Chalmers

Is the software in different ECUs compatible?



Car Maker (OEM)

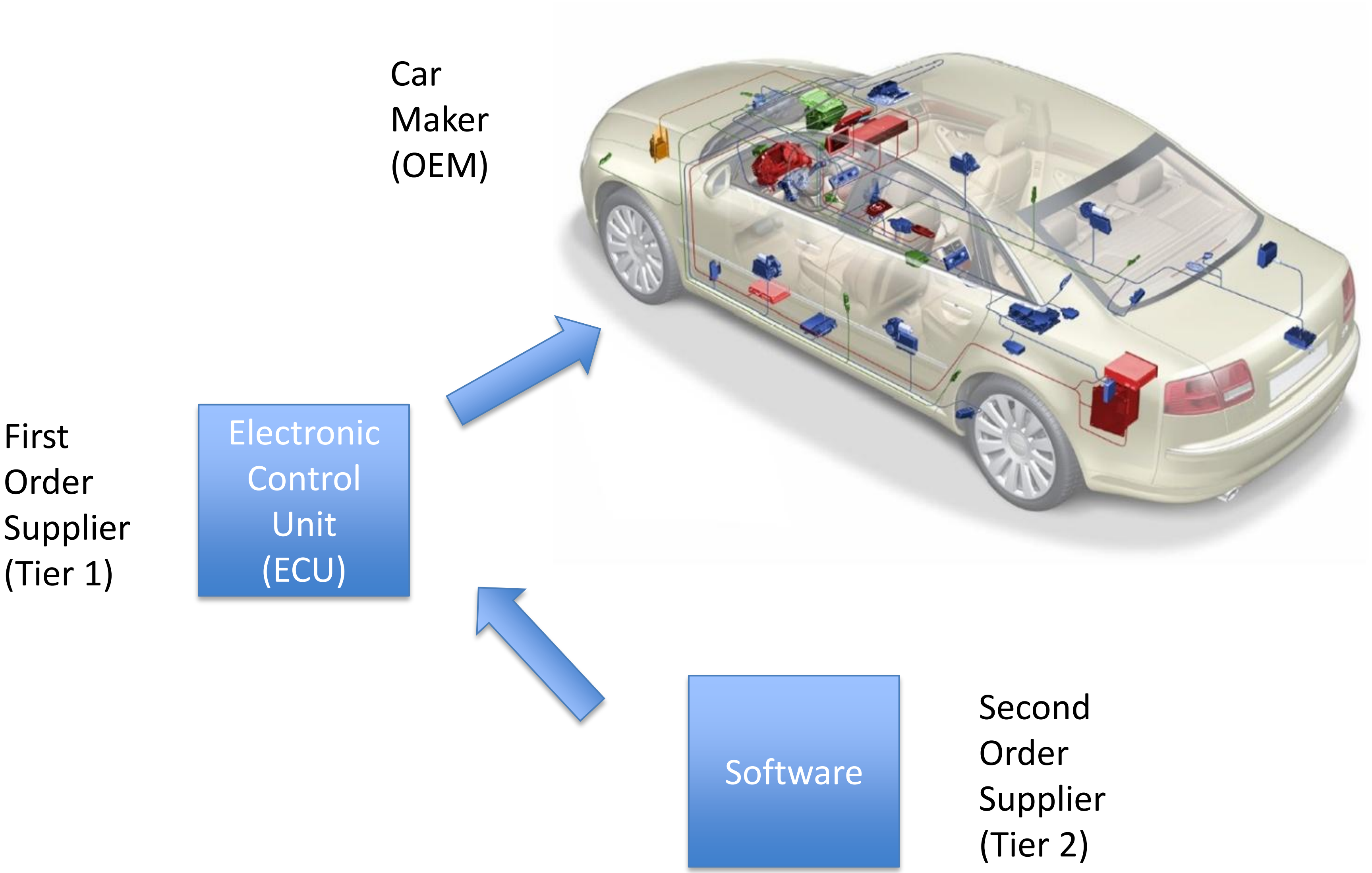
First Order Supplier (Tier 1)

Electronic Control Unit (ECU)

Software

Second Order Supplier (Tier 2)

Is the software implemented conform the specification?



Car
Maker
(OEM)

First
Order
Supplier
(Tier 1)

Electronic
Control
Unit
(ECU)

Software

Second
Order
Supplier
(Tier 2)

The problem

How to test for conformance?

Solution: outsource to India

Put 30+ person years on writing tests

Result: disaster

Why?

Testing AUTOSAR

AUTOSAR is a standard defined by a consortium:

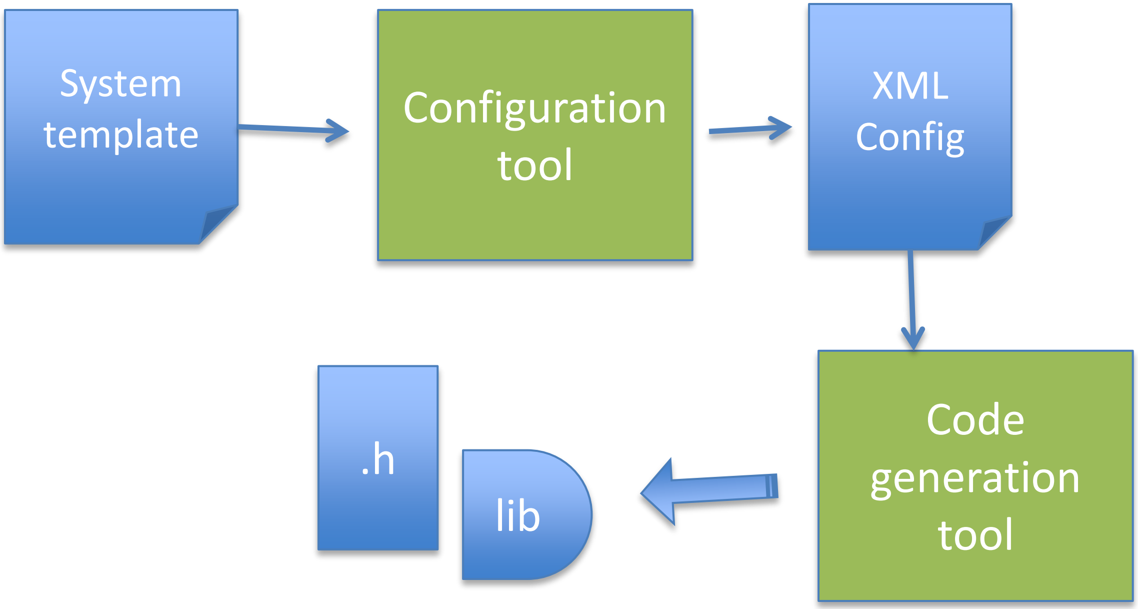
everyone wants their things in there

Result:

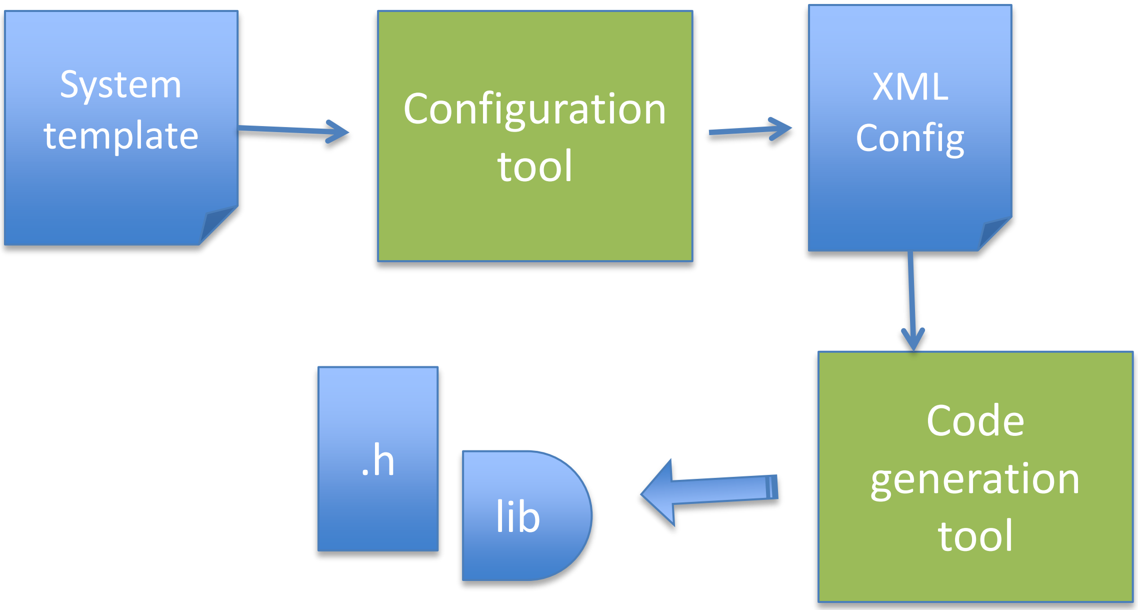
Everything is configurable. Thousands of parameters can be specified

The configuration file(s) and process are standardized ☺

Configurations are vendor specific



Configurations are vendor specific



A test is:

A configuration and a set of API calls with their expected results.

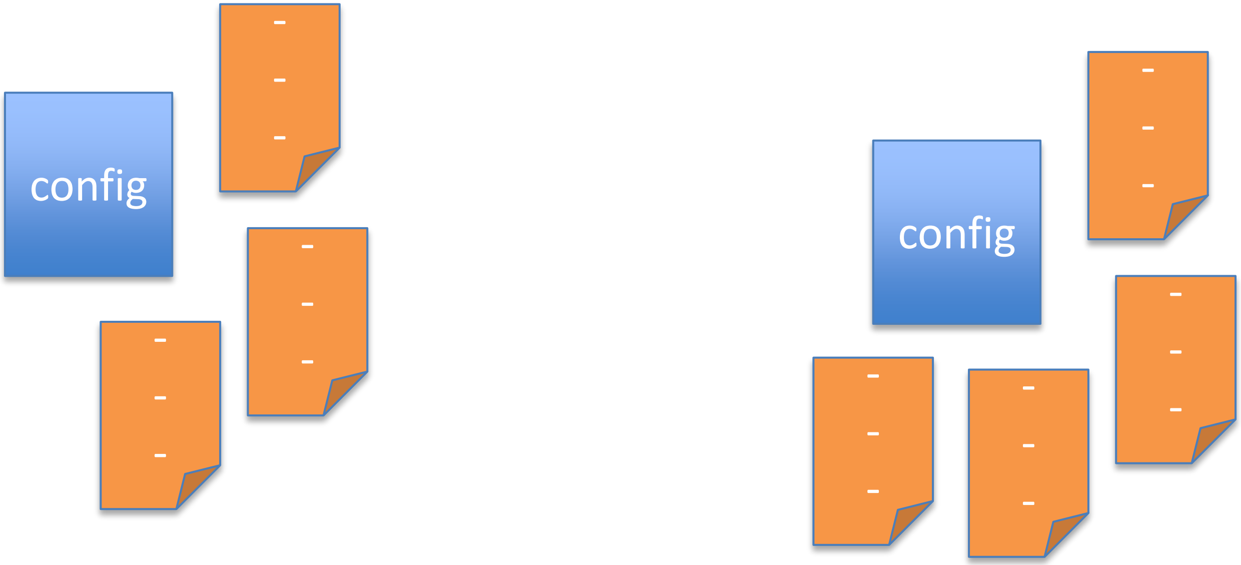
Tests

configurations are small

a number of API sequences per configuration

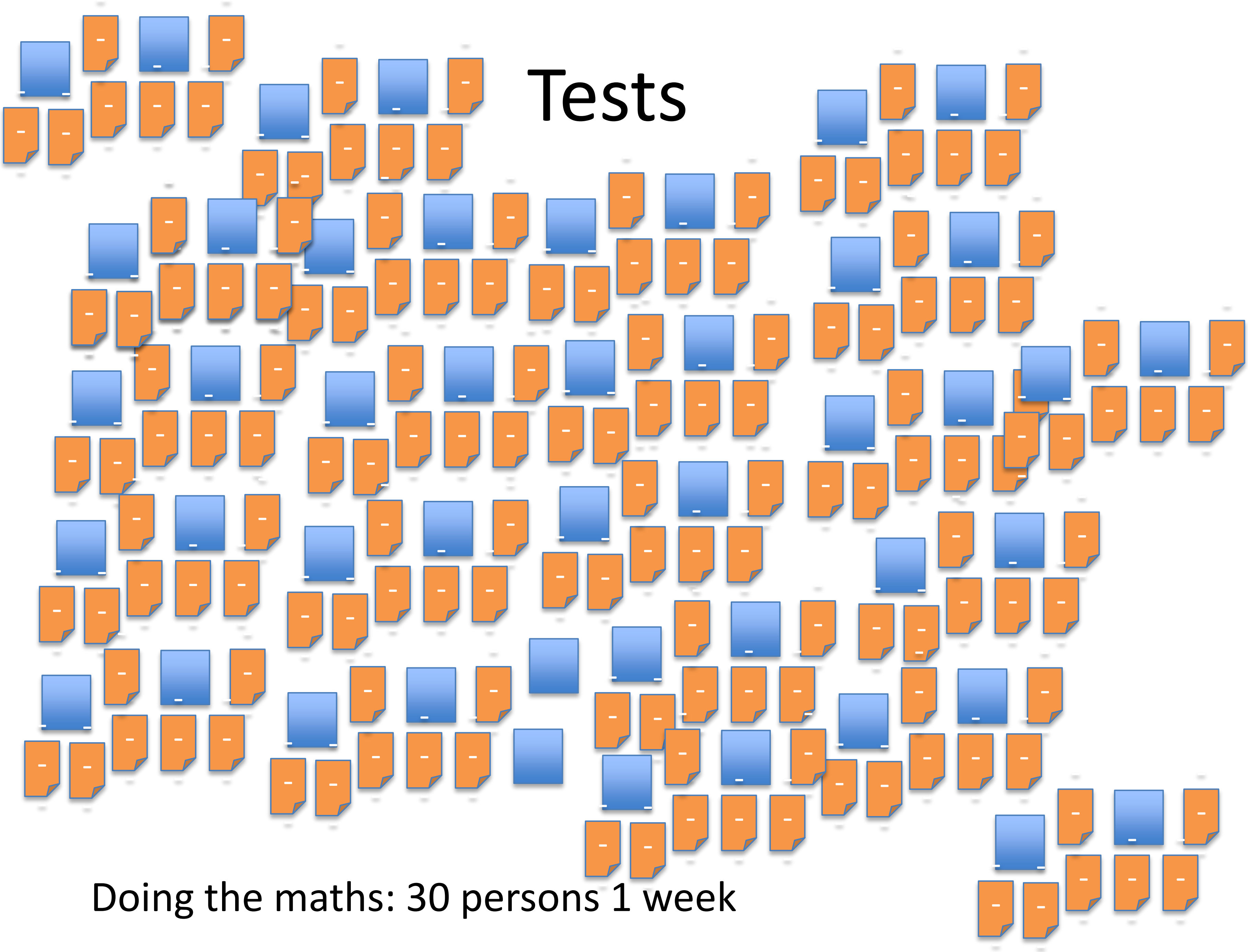
Vendor may need to change configuration a bit before code can be generated and test can be run.

Tests



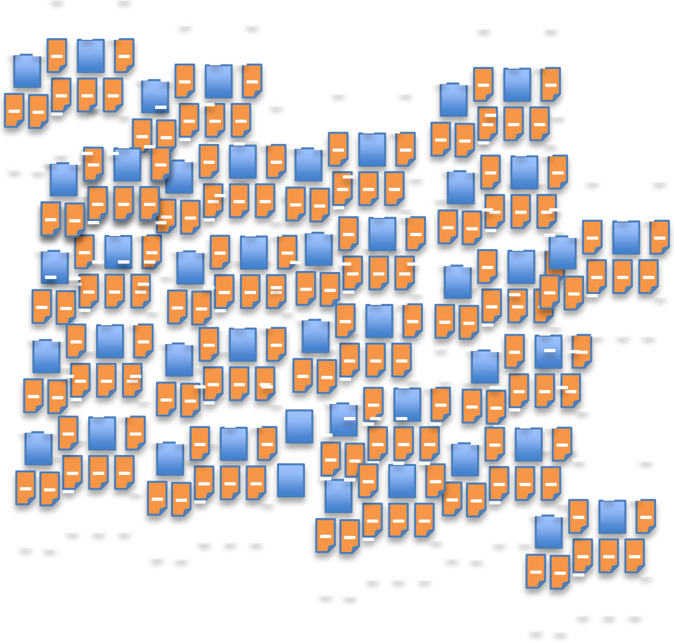
Doing the maths: 1 person 1 week

Tests



Doing the maths: 30 persons 1 week

Tests



X 50

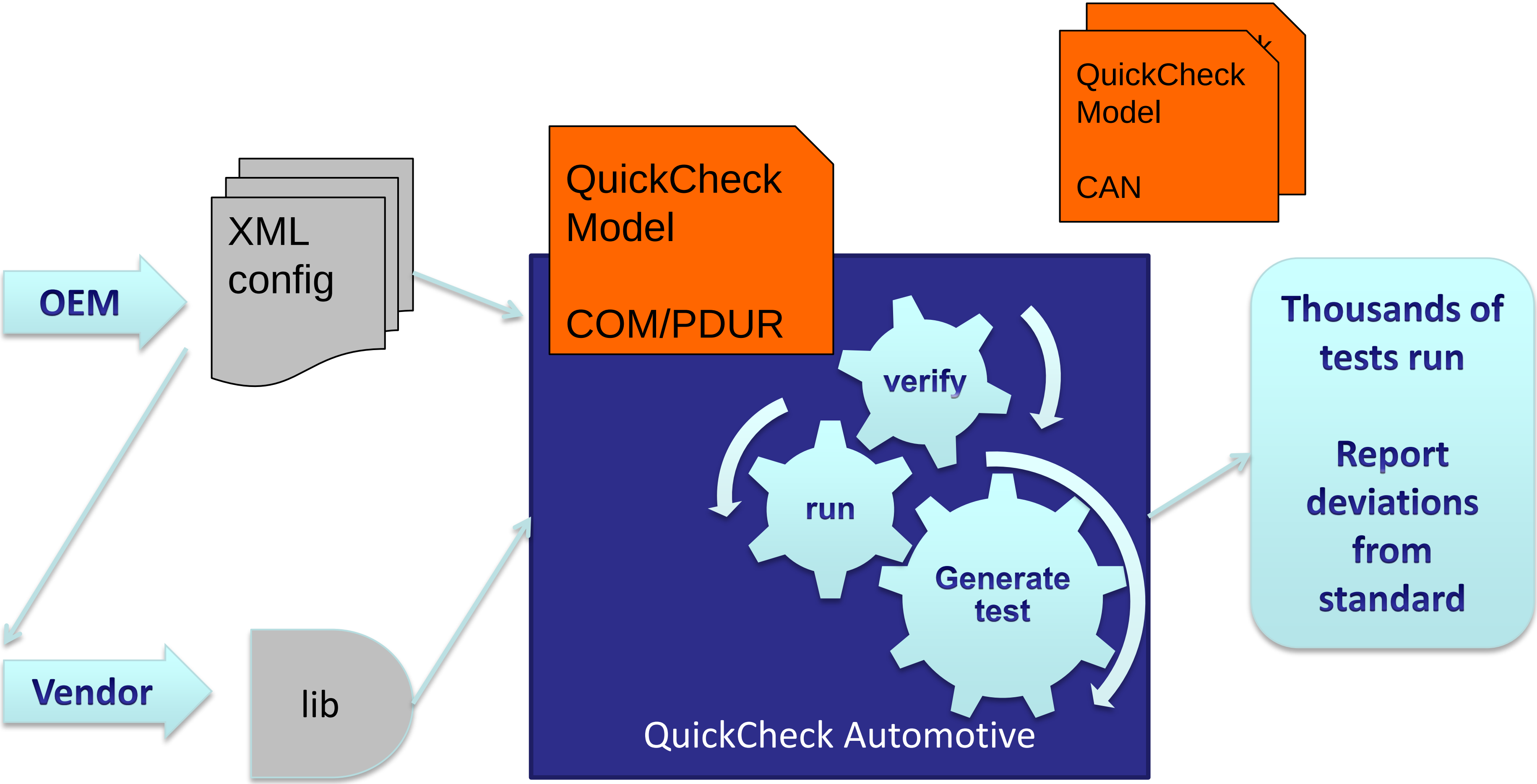
Doing the maths:
30 person years, 2-5 tests per week per developer, 3000-6000 tests

Executing those tests is a nightmare, since one needs to adopt the configurations and generate code

Traditional Testing

- Module testing, each module separately
- Minimal configuration to support a test case
- A few test cases for that configuration
- One feature/requirement tested at the time

Our approach: models



10 times less code
Largely independent of configuration
More scenario’s tested

Model-based testing

- Several modules tested in a cluster
- One large configuration supporting all test cases
- A huge number of test cases automatically generated
- All features/requirements tested at the same time

QuickCheck automatically generates all marshaling code to talk to C

Model based testing

We created models for 3 clusters:

COM/PDUR, CAN and FlexRay

We generated and ran tests against 3 already tested implementations of each cluster: 3 software vendors

We found deviations in each.

- some deviations on purpose,
- each vendor recognized some as bugs.

QuickCheck models

QuickCheck Model

CAN

500 pages of AUTOSAR document

- 1500-2000 lines of model code

  (cf. 15000 lines of C code)

- Building the CAN model

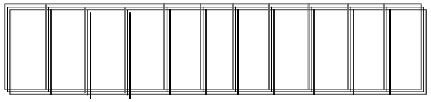
  About 12 weeks

Results

- Erroneous dependencies between features found
  - Mix of many features tested in same tests
- Failures found in “obvious fault-free implementation”
  - Everything is tested, even parts otherwise excluded by manual tests
- General higher coverage
  - Many more tests executed
  - All assertions always considered
- Common human mistakes detected
  - Common human errors for both developer and test designer are found by model
- Ambiguities in specification found more often
  - All assertions always to be considered, even for “absurd” scenarios

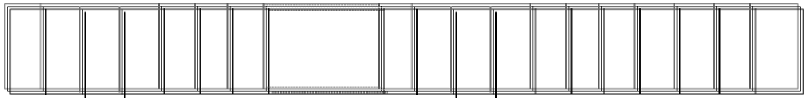
Example: Mixed features

StandardCAN Id



11 bits

ExtendedCAN Id



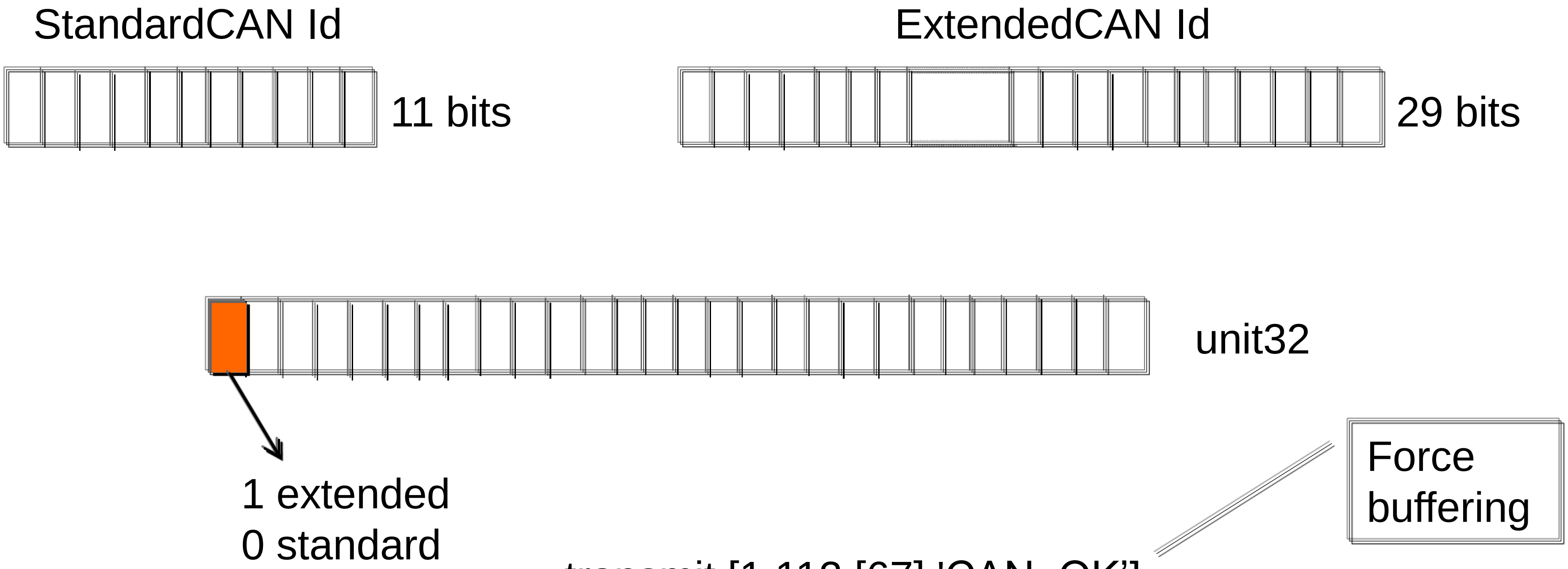
29 bits

Priority: lowest number has highest priority

Example:
Extended Id 113 has higher priority than standard Id 114

Buffered higher priority messages should be sent first

Example: Mixed features



```
transmit,[1,112,[67],'CAN_OK’],
transmit,[2,113,[0],'CAN_BUSY’],
transmit,[3,114,[0],'CAN_BUSY’],
tx_confirmation,[1,112,[67]]
```

Force buffering

Trigger sending

Check callouts: 112, 114 sent, why?

Example: Unrelated events

Stop a group =>

cancel any pending gatewaying for the included IPDUs

```
init,[],
start_group,[10,false],
rx_indication,[canif,1,<<0,0,0,0>>],
rx_main,[],
stop_group,[1],
route_signals,[],
tx_main,[]
```

Unrelated group

Check callouts: Nothing sent, why?

AUTOSAR

Similar when you implement

When we model, we interpret the standard

When we test we discover other possible interpretations

It gets interesting when we detect different interpretations among vendors

Conclusions

We created models for 3 clusters.

We can read configurations and adapt the model to these configurations… Thus, generated test cases make sense in context of that configuration.

Testing C code with Erlang and QuickCheck outperforms throwing man power to the problem.

APPROVED

With QuickCheck Automotive