Scania
Software development and testing
Agenda

• General info about Scania
• Software development at Scania
• Scania’s electrical system
• Integration testing
• Distribution of real-time data
Main message

• Scania is a software company
• Scania has a high degree of in-house development of ECUs
• A lot of freedom and possibilities to learn new things
Corporate statement

Scania’s goal is to deliver optimized heavy trucks, buses, engines, and services, offer our customers the best total economy and thereby be the leading company in our business segment. The foundation is Scania’s core values, our focus on methods and our motivated coworkers.
Haulage  Construction  Distribution  Special purpose

Premium products and services

Network and services

Intercity and coach  City and intercity  Engines
Modular product system
Scania Technical Center
SESAMM – Scanias electrical system
Modular Product System

- Well balanced performance steps
- Standardised interfaces
- Same need - identical solution

Scania Electrical System is a part of the Modular Product System
Scania Electrical System - Principles

- One common electrical system for all vehicle types
- Function allocation independent of vehicle specification
- Backward compatible
- Rebuildability
- High level of functionality in degraded mode
- Segments
- In-house development of SW in strategic nodes
- CEPPSS (Continuous Evolution of Properties Planned in Small Steps)
User Functions

– A User Function describes a vehicle function from which the user has a direct benefit

– The complete set of User Functions describes Scania’s electrical system

Opticruise –
UF 493 ”Transmission automatic”

More examples:
UF 352 ”Bus Stop Brake”
UF 415 ”Hill Hold”
UF 511 ”Rear Wheel Steering”
Allocation Elements

- An Allocation Element describes a logical component of a User Function as implemented in an ECU
Benefits

- Scalable
  - Few ECUs on low cost vehicles
  - Possibility to add systems and segments for increased content
- Modularised
  - Encapsulation and modularisation reduces communication need and complexity
  - Possible to chose degree of centralisation
  - Clear organisational responsibility for components and functions
- Evolution
  - CEPPSS
  - Balancing complexity and backwards compatibility
- Testing
  - ECU system level testing possible locally before delivery to integration test
  - Stepwise integration possible
- Isolation between ECU systems
  - Easier to prove freedom of interference and avoid unnecessary mixed criticality
- Flexibility in subsegments
  - Possibility to adapt interfaces quickly to new systems without affecting main segments
  - Often in-house SW in main nodes
- Builds on proven concept
Where are we in the organisation?

Mainly RE and NE developing SW, but also RB, RC and NB.
• Our E/E system is ONE system
• We have to see it as a whole system and not only separate parts of it
• "Small changes" can have/lead to unexpected dependencies
• We have to analyse each change to evaluate its consequence(s)
• Development of the E/E-system is performed parallel in many areas
• It is important to have a process for synchronisation

The release process is Scania’s process for packaging the electric/electronic system in our vehicles
What is the release process?

1. Release planning
2. Analyse
3. Packaging
A complete delivery of the electrical system per SOP

ECU-system development

- SW
- Electric
- Product documentation (System och function)

Release process

- Safety analysis
- Production tools (PSM/PSE/SPCT)
- Integration test report
- Field test report
- Communications specification
- Support for after market (Scomm, MSC)

Aftermarket process

- Tools for aftermarket (SDP3)
- Drivers manual
Development of SW

Code-Development

Function- and module tests

Dependency analysis

 Analyse/ Modularisation

 Safety analysis

PD Phase 1: D "Impact on electrical system defined"

Production tools

Integration tests

Field tests

Support for after market

Release process

SW-projects (working method e.g., RUP, SCRUM)

SOP

ECU-system

Complete E/E-System
Software Development

Embedded System Development Process

- Dependency analysis
- Analysis / Modularisation
- Test planning

SW development

Test

Electrical system, HW & SW
Production tools
Integration tests
Field tests
Support for after market

CD
PD
SOP

UTGÅVA 1, 2009-03-31
Info class Internal
RESA/Staffan Persson
Releaseprocessen
Release process

- Release process is a flow with a pulse
- Each planned change is flagged with a **CR**

Abbreviations:
CR = Change Request, P1 = Integr.test 1, P2 = Integr.test 2, P3 = Integr.test 3
SCRUM
Methodology for SW development
SCRUM is determined by

- Iterative
- Increments
- Focused work in short cycles
- Priorization
- Self-organized team

- Everything is timeboxed
- Transparent
- Face to face
- Periodic deliveries
Req. from, customers, teams etc.

Product owner

SCRUM master

Team

Ceremonies

Planning

Daily SCRUM

Demo

Retrospective

Artefacts

Backlog

Sprintlog

Deliverable product increment
Test environments for integration testing
I-lab: Hardware-In-the-Loop

- Electrical signals
- Fault injection
- Signal handling of sensors
- Dynamic vehicle model
- System under test: HW+ SW
- Load simulation
- Fault injection
- Signal handling of control signals
Haulage

City traffic

Construction

Long range

Test report

~250 TC per night

SOPS + ~180 TC

~250 TC per night
Distribution of real-time data

• Our new integration test lab has a CAN bus solution where maximum CAN length is reached
• To reduce CAN length, buses are not accessible in all cabinets
• However, we still want to get one real-time view of all CAN buses
• This requires a distributed solution
Distribution of real-time data

- We allocate one CAN bus as a synch bus
  - A synch message is sent periodically
- The synch bus is accessible from each computer node
- Each computer node receives CAN frames on CAN buses, including synch, and sends them to a merger
- How should the merger be implemented such that it
  - Can cope with the expected number of messages
  - Can present a merged data stream without too much delay
Distribution of real-time data

computer

computer

computer

Synch bus

Merged stream

merger
Distribution of real-time data

- One program for sending synch messages
- One program per computer for receiving and forwarding CAN frames
- One program for sorting frames in correct order
- One program for visualizing CAN frames
Distribution of real-time data

- Merging
  - Robust
  - Handle all kinds of edge cases
  - Maintainable
- Development
  - A C++ version has been implemented. Single-threaded.
  - A prototype using Actor pattern has been implemented. We consider this one to be more robust and maintainable.
  - Programming language with thread local heaps
Distribution of real-time data

TCP/IP → Receiver → Filter n → synch → merger → distr → TCP/IP

filter1