

Safety and Resilience Issues in Automotive Software Development

Luca Simoncini

University of Pisa, Italy

Example: Modern Cars

- **Embedded control systems in modern car** (brakes, transmission, engine, safety, climate, emissions, ...) Complex interaction increases development costs

Complex interaction increases safety risks



vehicle-to-vehicle network
vehicle-to-environment



- **The resilience of the car you will be driving in the future will depend on the dependability of the software in the other cars around you. Your airbags might inflate, for instance, if the software in a nearby car erroneously warns your car that it is about to crash into it.**
- **In the future car we will have an unimaginable number of powerful computing devices that are all connected, and that can potentially interact.**
- **Each of these powerful devices may be controlled by multi-million line programs. Who is going to write this software, and how can we make sure that the resulting systems are reliable?**
- **Your car may now fail not because of faulty software in your own car but because of faulty software in someone else's car that interacts with yours in an unexpected way.**

Trends in Automotive (on-board embedded systems)

Three groups of trends:

- **Advanced comfort**
 - e.g. Car Body Electronics (adaptive equipment: seats, superposed adjustable steering wheel, (no) pedals,)
 - Noise suppression, adaptive air conditioning, configurable cockpit,
 - Navigation, communication, information, new types of displays

- **Safety enhancement**
 - Vehicle Dynamics (ABS, ASR, ABC, ESP (Electronic Stability Program), AAS (Active Additional Steering), Adaptive Cruise Control, Road Tire friction Control, ...) **Safety – critical controls !**
 - Advanced Warning- and Control Systems (pedestrian protection, crash avoidance, track control, lane support, ...) **Safety – critical controls !**
 - Driver Monitoring, Predictive Driver Assistance, Emergency call system

- **Optimized resource usage**
 - Power Train (Integrated Engine Control, Transmission Control) **Safety – critical controls !**
 - Fully integrated Electrical Energy Management

Trends in Automotive (eSafety on the road)

Extending autonomous on-board functions with interactive and co-operative systems:

- Roadside embedded systems and interaction (intersection, speed control, emergency call systems)
- Local connectivity: vehicle – to –vehicle (long term) – highway throughput optimization, advanced adaptive cruise control
- Global Connectivity: Satellite, traffic navigation and control

Ultimate Goal: Autonomous Driving, “Platooning” of vehicles

- Liability, Legal and Standardization Issues !!

Linking to global infrastructures: Link to “Ambient Intelligence”

Security Issues: Connectivity during Operations & Maintenance !! (Call-back, Upgrades off-line or on-line ?)

Enabling Technology for all of these trends: DES !!

DES Challenges: Major Issues in System & Software Technology

- Requirements Engineering, with respect to Dependability,
- Model Based Development
- Reuse/COTS/Composability/Scalability wrt. Dependability
- Standardisation (cost, maintainability, interoperability)
- (Modular) Certification
- Dynamic Environments (ubiquity/nomadcity - mobility, low power)
- Systematic Testing, Validation and Certification
- Temporal Predictability (TTP) vs. Uncertainty
- Human Factors (HCI)
- SoC: Shrinking feature size, new failure modes
- Diagnosis and Maintenance: Autoconfigurability, Plug & Play, Diagnosis

Dependability:

The 10^{-9} Challenge can only be managed by an architectural approach (components about 10^{-4} to 10^{-5} only – the system is more dependable than each of its components !)

DES - Challenges

Safety Systems Concerns:

- resources shared between functions (encapsulation of task environs)
- stronger interactions among them
- more functionality at less cost (cost explosion in development ?!)

Safety is a system property:

- New hazards arise from fault propagation in composed systems and unintended emergent behaviour

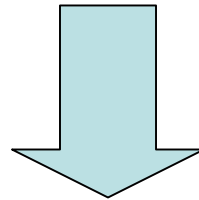
Need for modular Safety Analysis and Certification, depending on

- Partitioning
- Safety Function
- Controlled Failure

- Which problems will have to be solved before we can trust these types of systems?
- The basic methods that are used in software development have not changed much. So how is it that today one can indeed write million line programs, and, at least some of the time, get them to work reliably?
- A good requirements process, a careful system of checks and balances, and perhaps most importantly good tool support is not sufficient even if the quality of the software development tools that programmers use today to develop and test code is vastly better than it was 10-15 years ago.

Today's tools support a style of programming that is at odds with the needs of the types of applications that will soon dominate. They provide good support for the development of sequentially executing code, but have only limited support for multi-threading and concurrency.

Today it is quite difficult to test or to debug a truly distributed application.



SPIN +

The perfect goal will be to build an interactive code checking system

It is necessary that in the next decade programmers will gain access to a new breed of smart source code analyzers for both sequential and distributed systems code. These analyzers can be embedded into standard program development environments, and they can work without the programmer being aware of it. These analyzer demons will be able to warn the programmer virtually instantaneously when subtle bugs get introduced into the code -- the true ideal of a programmer's omniscient helper.