Telematics Architecture
Table of Contents

- Telematics products
  - History, present and trends
- Software Architecture
- 4+1 View applied on telematic products
Early Telemetry in Automotive
Early Telemetry in Automotive
Definition (1)

› “Telecommunication Informatics is the technology of sending, receiving and storing information via telecommunication devices in conjunction with effecting control on remote objects” (wikipedia)

› Telematics can be regarded as an evolution of telemetry (transmission of measurements from the origin to the location of computing and consumption without affecting control on the remote objects)
History – initial features

› In the automotive domain the term evolved to refer to automation in vehicles

› First products targeted emergency warning systems – e.g. automated crash notification, emergency calls, roadside assistance, concierge (where is the closest gas station?)

› Door locking/unlocking and stolen vehicle tracking were added as security-targeted features (Boston bomb attack, stolen car had vehicle tracking)

› Remote diagnostics increased safety by early detection of failures
Recent introduction of 3G/4G mobile communication standards and cheap mobile data enabled permanent Internet connection

- Telematic Control Units were the first ECUs to support **over-the-air software updates**
- Infotainment head units benefitted as well – e.g. navigation data, internet radio streaming, web browsing, etc.

- **Integrated WiFi modules** allowed Internet connection sharing with passengers
Telematics products – external actors
Applicability

- At a very high level telematics systems comprise:
  - A processing unit connected to the local system’s control interface – serial port(s), data bus(es)
  - A communication device (usually a cellular network device)
  - A remote command and control center

- Telematics systems have a large domain of applicability – automotive, utility distribution (gas/electric), road safety, intelligent transportation, military, etc.
Types of telematic services

The telematics services can be categorized as below:

- Safety and Security
- Information and Navigation
- Entertainment
- Diagnostics
Safety and security

- *safety services* include the automatic crash notification, emergency and medical assistance
- *security services* include stolen vehicle tracking, anti-theft alarm notification and remote door services

Automatic crash notification

GPS vehicle tracking system
Information and navigation

using *the information and navigation services* the vehicle occupant can get access to location sensitive information and content

data example of services in this category:

- geofencing
- point of interest download
- turn by turn navigation assistance
Entertainment services

example of services in this category:

• on demand direct music/video downloading

• internet radio

• streaming content and synchronization with home entertainment library
Diagnostics

the *diagnostics services* include

- remote diagnostics
- performance data collection
- remote DTC scanning
- the TCU can perform detailed diagnostic scans when triggered remotely or when certain key thresholds are crossed
Software Overview
Basic Product Features

- Emergency calls – regulated (EU eCall, ERA)
- Remote door control
- Stolen Vehicle Tracking, Theft Alarm, Vehicle Locator
- Software Update over CAN
Software Overview
Advanced Product Features

› Concierge and Roadside Assistance calls, information messages, remote diagnostics

› Geo-fencing (fleet management), navigation assistance (POIs & route download)

› Internet connectivity, WiFi hotspot

› Backend gateway

› OTA Software Updates for TCU and other ECUs
Trends

› Core of the smart/connected car
  › The TCU is the only ECU able to maintain a permanent connection with vehicle manufacturer’s backends for various scenarios
  › Multiple “smart”/”connected” ECUs rather than a single one
    › Non-critical ECUs will be required to implement a backend communication mechanism
› Remote software update of the entire vehicle becomes possible
› Faster vehicle data buses allow better integration with the HU
  › Improved voice calls quality
  › Smart device-like features – navigation/traffic, complete internet access, social media
Initial telematics ECUs comprised a NAD (Network access device), a VuC and a GNSS (global-navigation-satellite-system) receiver:

- The NAD (Network Attached Device) connects to the cellular network for voice calls and SMS.
- The VuC (Vehicle MicroController) handles calls and incoming commands while also interfacing with the vehicle bus (crash notifications, power cycles, diagnostics, software update).
- The GNSS (Global Navigation Satellite System) receiver provides location info to feature node(s).

This design is still used for entry-level Telematics products.

Pros: easy implementation of basic functionality, low cost solution

Cons: reduced set of features, low profitability, 2G nearing EOL
Hardware Overview
Mid and High End Designs

› Expand on the entry level design through a powerful CPU that allows implementation of complex features – Internet connectivity, Over the air software update, remote diagnostics, navigation helpers

› 3G/4G/5G/etc. modems enable faster data speeds

› Extend connectivity options to WiFi, Bluetooth (Low Energy)
  › Internet connection sharing with the Head Unit and vehicle passengers
  › Wireless workshop diagnostics

› BRR (BroadR-Reach) enables high-quality audio connections to the HU for non-safety critical voice calls, internet radio streaming

› High quality and improved usability, but more expensive solution
Architecture definition

The Software Architecture is the organizational structure of the Software application describing the decomposition in its software subsystems (classes) with their connectivity and interaction mechanisms. It includes details related to principles and decisions used to build the Software.
Why Architecture?

› Specify what needs to be created and how.

› Architecture creates a common understanding between all relevant stakeholders about the final product and about the methods. (Bedouin, Citizen, Martian).

**UML** (Unified Modeling Language) is a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system.
Architecture structure (4+1 view)

Software Architectures includes different views

1. **Logical View** – specify how the end-user functionality will be implemented
2. **Physical View** – describes what piece of software will run on which hardware.
4. **Development View** – configuration management and deliveries
5. **Scenarios** – use cases of the final product
4.1

Diagram showing the relationships between different views in a system development process:

- **Logical View**
  - End-user Functionality
  - Integrators
    - Performance
    - Scalability

- **Development View**
  - Programmers
    - Software management
  - Scenarios
  - System engineers
    - Topology
    - Communications

- **Process View**
  - Integrators
  - Performance
  - Scalability

- **Physical View**
  - Programmers
    - Software management
The software architecture role

- **Architectural Drivers**
  - Understanding requirements and constraints

- **Technology Selection**
  - Choosing and evaluating technology

- **Architecting**
  - Designing software

- **Architecture Evaluation**
  - Understanding that the architecture works

- **Coding**
  - Involvement in the hands-on elements of software delivery

- **Architecture Evolution**
  - Ownership of the architecture throughout the delivery

- **Quality Assurance**
  - Introduction and adherence to standards and principles

- **Coaching and Mentoring**
  - Guidance and assistance

---

Confidential

10/19/2018
How to create SW Architecture

1. Get all software requirements and system architecture.
2. Create use cases based on requirements
3. Define or reuse software components that can fulfill those use cases
4. Describe functional use case realization in logical view
5. Think about non functional requirements and create concepts for them in process view
6. If is not defined by System Architecture define where and how your piece of software will be deployed on hardware in physical view
7. If is not defined by an external team of System integration, define how your software will be integrated and delivered in deployment view.
Requirements

› Most of the requirements are coming from **Automotive Domain**.

› We have to use embedded systems (they fit best into the auto vehicles)

› We have to use the buses available in auto vehicles (CAN, MOST). There are gateways from HW to SW. There are interfaces defined for each bus. Also drivers and specific binaries.

› We have to use busses available for embedded devices (USB, BlueTooth, I2C, I2S)
Requirements - examples

FMBeC - XXXX

If an automatic MB eCall is triggered, the system shall send the signal `<S_SOS_LED_Status>` (flashing), which activates the flashing of the LED in the SOS button.

Remark: The 1 Hz flashing frequency is generated by the SOS Button itself and is therefore no requirement for the TCU.

FMBeC - YYYY

The system shall provide the ability to end an automatic eCall by a single press on the MFL end button after the timer `<V_KEYLOCK_TIMER_AUTOMATIC_ECALL>` has expired.
Architecture in Rhapsody - Scenarios

- Scenarios can be created only based on requirements and are depicted in Use case diagrams.

- A use case defines an interactions between a role ("actor") and a system, to achieve a goal. The actor can be a human, an external system.
Allocate requirements to Subsystems is done using Object model diagrams.

Requirements shall be:
- satisfied by subsystems or classes
- traced by use cases
Remote door lock/unlock

Locked Car Keys In Car

Remote Unlock

User

10/19/2018
Logical View – Sequence diagram

Is the most commonly used diagram to show how objects (may be actors, components, classes, etc.) interact with each other to satisfy desired behavior.
Logical view – class diagram in Rhapsody.
Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another.
Logical View – State chart Diagram

› Depicts a class functionality as a state machine.

› A class may have only one associated state or activity diagram.

stm [Use Case] providePowerManagement [Power_States_BLE]

- Standby
  - Start_BLE_Activity_Command_Over_SPI
  - ERA_CallBack_Transition_CMD_Over_SPI

- PowerUp_and_Init

- Active

- PowerDown

- KL_30_or_BuB_Connection

The exact description for the standby is to be checked with the BLE colleagues. This is to prevent potential misunderstandings.

In this context, the standby has the following feature:
- BLE is interruptable via SPI interface
- BLE has no Bluetooth activity, e.g. advertisement, etc.
- BLE consumes as low power as possible
Development View – Model object diagram

- Mapping of SW Blocks to actual files and folders
Physical View - Deployment diagram
Process View – Security (in Diagnostics connection)

- Diagnostic Client
- Diagnostic Service Router
- Internal Diagnostic Server

- HandleUDSMessage()
- HandleGenerateSeedMessage()
- GenerateSeedResponse()
- HandleSendKeyMessage()

[security key sent by client is the same as the computed key]
- PositiveResponse()

[else]
- NegativeResponse()

Generate seed, use security constant and seed to compute the security key

Unlock security access
Questions
the main components of a telematics system are as follows:

› • telematics control unit
› • telematics network operations system
› • wireless communications infrastructure
› • service provider call center
› • service/content provider
Refer to HW/SW Interface Description