Curs nr. 6, 22.03.2015

Human Machine Interface

Ady Popa

Summary

- **HMI in general**
  - Definition
  - HMI history
  - HMI usability

- **Automotive HMIs of today**
  - Introduction
  - Architectural aspects
  - GUI tool chains
  - GUI testing

- **Automotive HMIs of the future**
  - Future HMIs
  - Future technologies

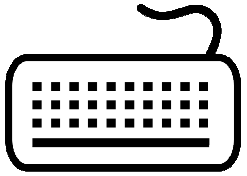
Summary

- **HMI**
  - **Definition**
  - HMI history
  - HMI usability
- Automotive HMIs of today
- Automotive HMIs of the future

Summary

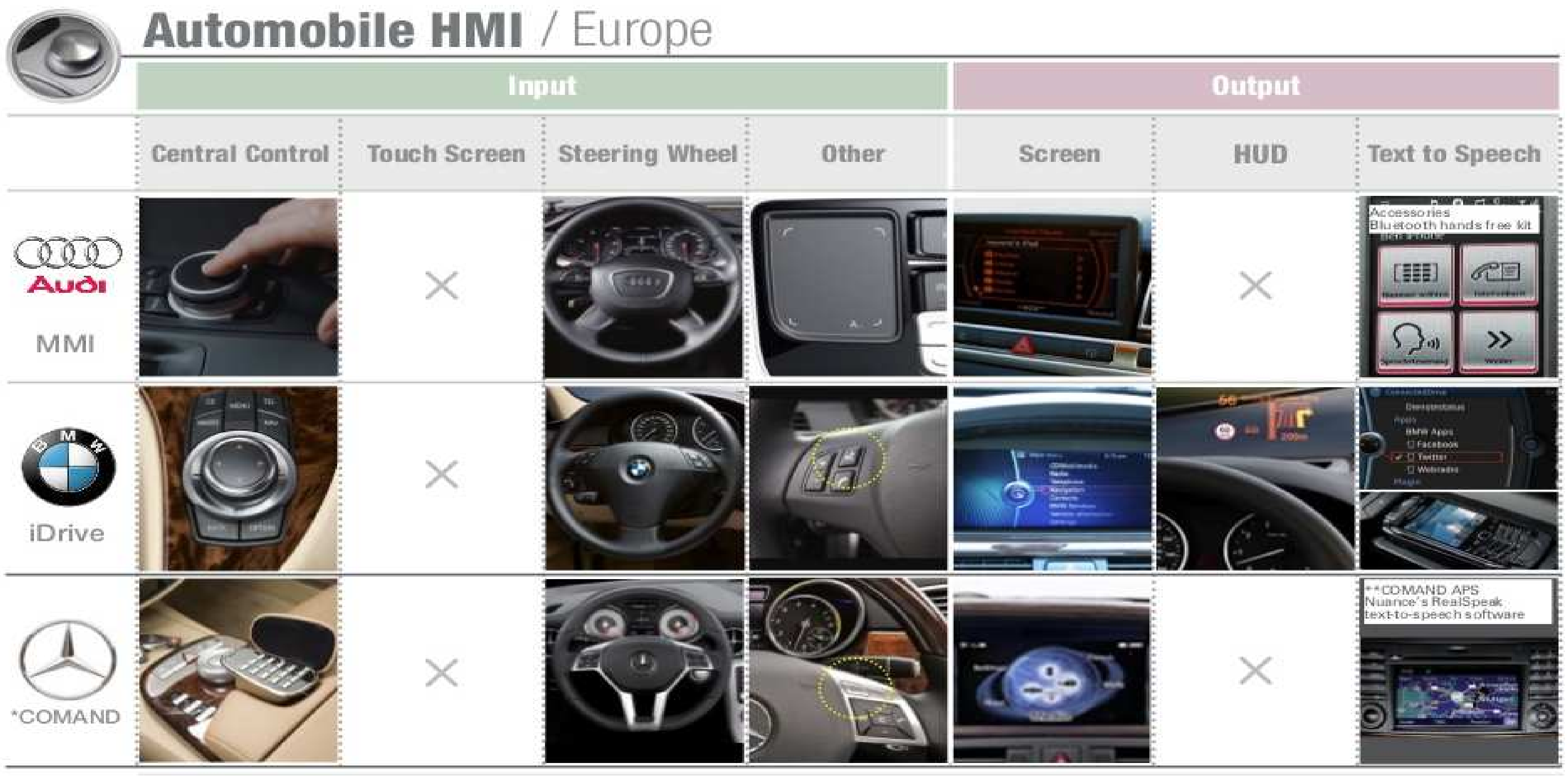
- **HMI**
  - **Definition**
    - The Human Machine Interface is where people and technology meet. We all use HMIs to monitor and control a wide range of electronic systems, such as our home thermostats, building elevators, bank kiosks, gas pumps, manufacturing operator panels, and data access terminals.

HMI

- **Human Machine Interface(HMI, HCI or MMI)**
  - Input 
  - Output
  - Consistency ("*As far as the customer is concerned, the interface is the product*." Jef Raskin)
    - The control for different features should be present in a consistent manner.
    - Principle of least astonishment.
    - Consistency counsels against user interface changes version-to-version.

HMI

Human Machine Interface(HMI, HCI or MMI)



Automobile HMI / Europe

	Input				Output		
	Central Control	Touch Screen	Steering Wheel	Other	Screen	HUD	Text to Speech
MMI		✗				✗	Accessories Bluetooth hands free kit
iDrive		✗					
*COMAND		✗				✗	**COMAND APS Nuance's RealSpeak text-to-speech software

HMI

Human Machine Interface(HMI, HCI or MMI)

Automobile HMI / Asia

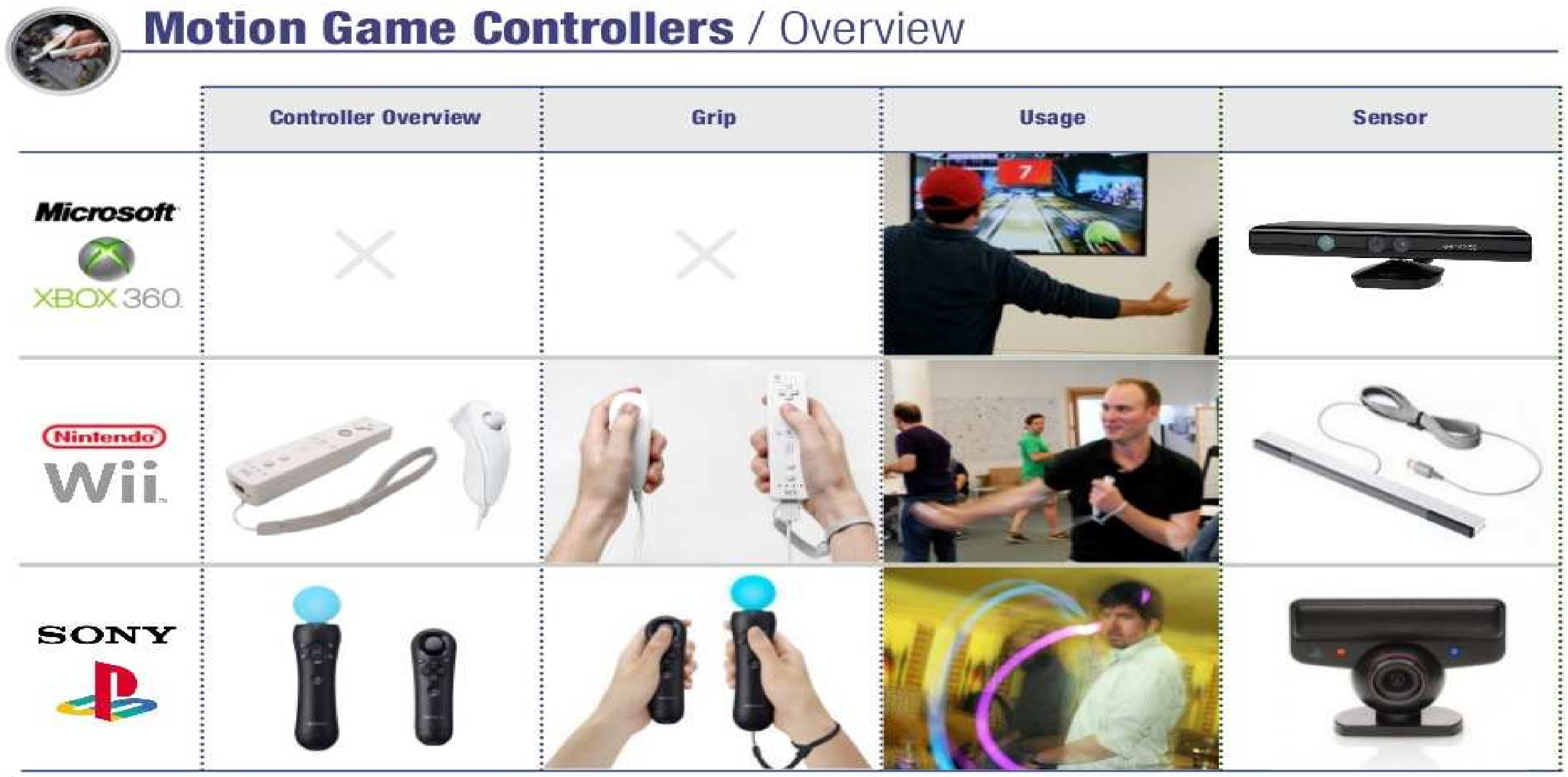
	Input				Output		
	Central Control	Touch Screen	Steering Wheel	Other	Screen	HUD	Text to Speech
LEXUS Remote Touch		×					* Expected 2013 Lexus GS 350 ×
INFINITI				×		×	×
HYUNDAI						×	×

HMI

- **Human Machine Interface(HMI, HCI or MMI)**

HMI

Human Machine Interface(HMI, HCI or MMI)

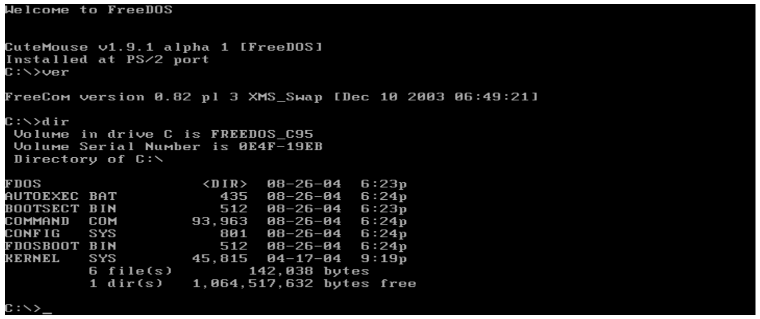


Summary

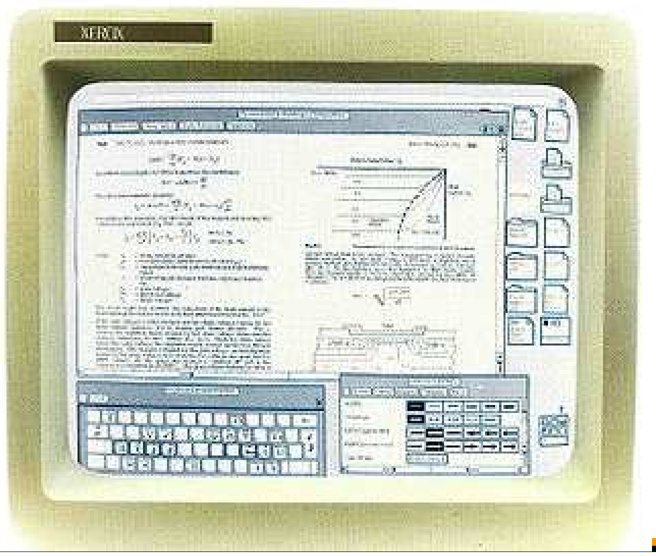
- **HMI**
  - Definition
  - **HMI history**
  - HMI usability
- Automotive HMIs of today
- Automotive HMIs of the future

HMI History

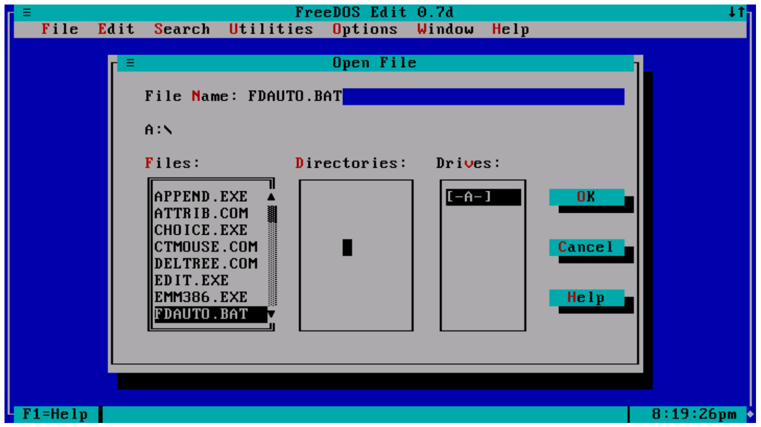
- Command line interface(DOS)



```
Welcome to FreeDOS


CuteMouse v1.9.1 alpha 1 [FreeDOS]
Installed at PS/2 port
C:\>ver

FreeCom version 0.82 pl 3 XMS_Swap [Dec 10 2003 06:49:21]

C:\>dir
 Volume in drive C is FREEDOS_C95
 Volume Serial Number is 0E4F-19EB
 Directory of C:\

FDOS                 <DIR>  08-26-04  6:23p
AUTOEXEC BAT           435  08-26-04  6:24p
BOOTSECT BIN           512  08-26-04  6:23p
COMMAND  COM        93,963  08-26-04  6:24p
CONFIG   SYS           801  08-26-04  6:24p
FDOSBOOT BIN           512  08-26-04  6:24p
KERNEL   SYS        45,815  04-17-04  9:19p
         6 file(s)        142,038 bytes
         1 dir(s)   1,064,517,632 bytes free

C:\>_
```



- Text user interface(Free DOS)



```
File Name: FDAUTO.BAT

A:\

APPEND.EXE
ATTRIB.COM
CHOICE.EXE
CTMOUSE.COM
DELTREE.COM
EDIT.EXE
EMM386.EXE
FDAUTO.BAT
```

[-A-]

Xerox 8010(1981): user interface consisted of graphical elements such as windows, menus, radio buttons, check boxes and icons. The PARC user interface employs a pointing device in addition to a keyboard

HMI History

HMI Phone evolution

Summary

- **HMI**
  - Definition
  - HMI history
  - **HMI usability**
- Automotive HMIs of today
- Automotive HMIs of the future

HMI Usability – What do we expect from an HMI?

- **HMI Usability** (In many cases, the user interface to a program is the most important part for a commercial company: whether the programs works correctly or not seems to be secondary. Linus Torvalds)

- The standard of ISO 9241 states three quality components of the use applicable to the HMIs design:
  - ✓ **effectiveness** - Does the product do what the users require/expect? Does it do the right thing?
  - ✓ **efficiency -** Can the users learn the HMI quickly? Can they carry out their tasks with minimum expended effort, including a minimum of errors? Does it improve the productivity/effort ratio? Does it do things right?
  - ✓ **satisfaction** - Do users express satisfaction with the product? Does the new product reduce stress? Do the end users now have a more satisfying job?

Else....

HMI Usability – Why is it important?

- Crucial in safety-critical applications (nuclear power-station management, air-traffic control ...)
- In the automotive field, it is critical to define **clear interfaces that are easy to understand and operate**, in order to keep distraction of the driver at a minimum.
- Delivering the increasing applications complexity while maintaining ease of use is a challenge

How the mentioned targets can be achived?

Usability Engineering Principles:

•#1: Know Your Users
•#2: Involve Users Early and Continuously
•#3: Rapid and Frequent Iteration Toward Measurable Usability Targets

Summary

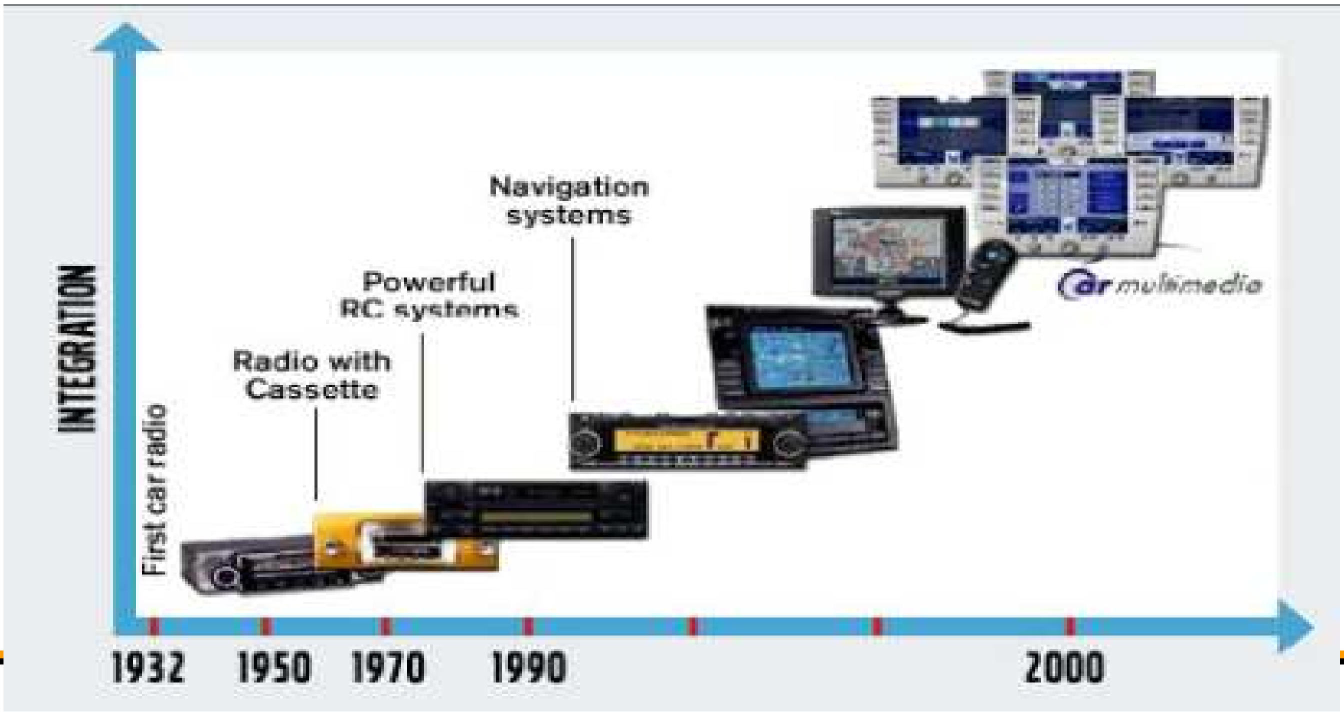
- HMI
- **Automotive HMIs of today**
  - **Introduction**
  - Architectural aspects
  - GUI tool chains
  - GUI testing
- Automotive HMIs of the future

Automotive HMIs of today - Introduction

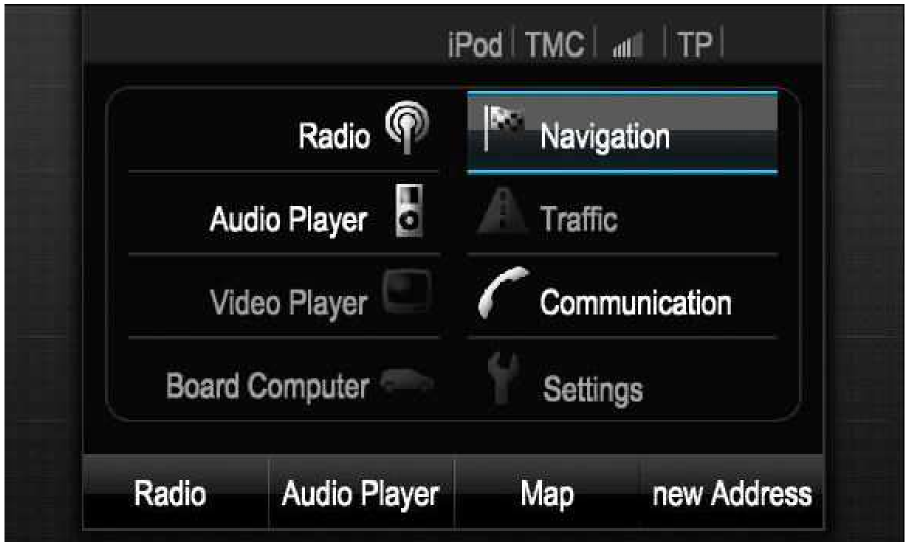
"... today's cars and trucks are: highly sophisticated consumer electronics..." (Rick Wagoner, ex-General Motors CEO)

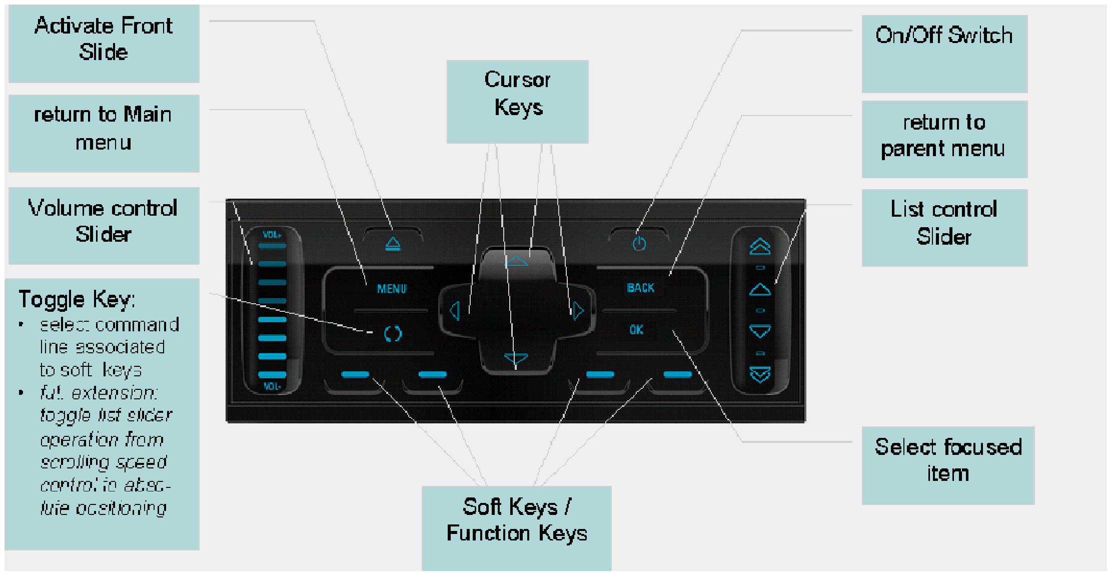
The car becomes an extension of the home, office, wireless and portable consumer environments.

The HMIs are essential in networking vehicles with the environment, to other vehicles and the drivers themselves

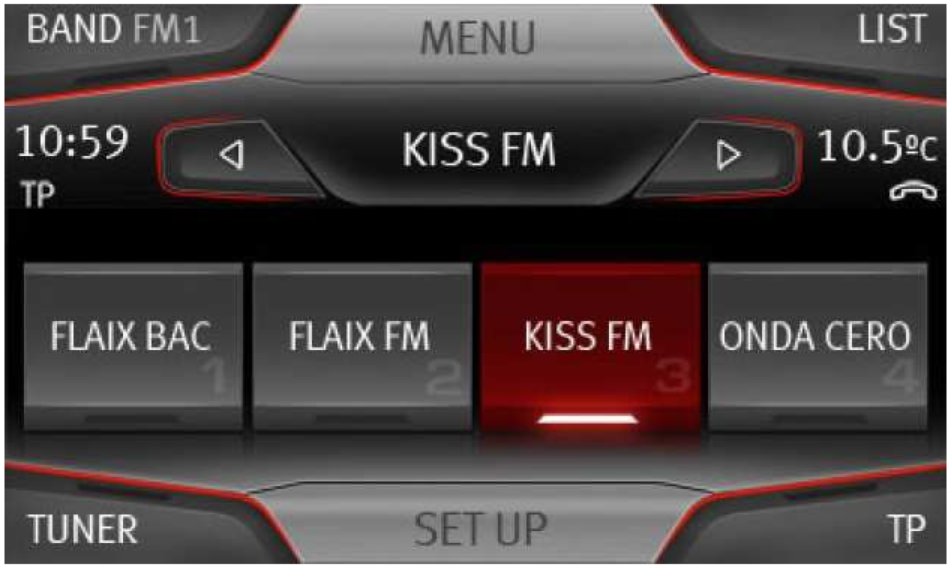


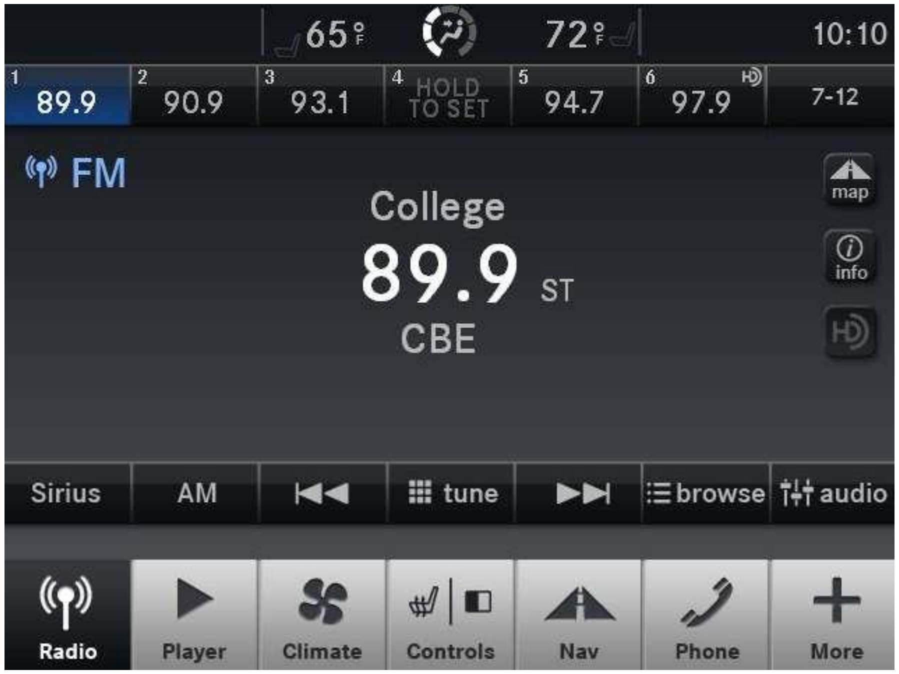
Integrated HMI Concept – Example #1





Integrated HMI Concept – Example #2





New concepts

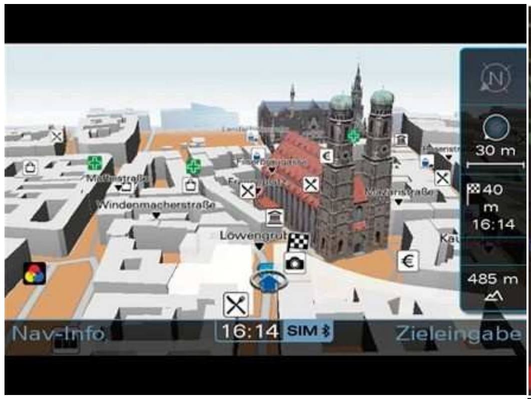
- New developments in driver orientation and vehicle control have emerged, enabling control of all kinds of different functions through just a few instruments
- The control elements crucial to driving, plus the most frequently used functions, are positioned around and on the steering wheel (indicators, windscreen wipers, headlight controls and basic sound system and communication controls)
- Most other functions – A/C, navigation settings, suspension settings - are activated through the Controller: a single turn-and-push dial in the centre console
- The Controller can be used with just one hand, and offers haptic feedback so it can be used without even looking at it. iDrive helps drivers use climate control, communication, navigation and entertainment systems safely whilst driving by utilizing an intuitive display system designed to cause as little disruption to concentration as possible

Input HMI technologies inside vehicles

- Key-panel
- Touchscreen
- Steering wheel controls
- Voice recognition
- Gesture recognition / Head-gesture recognition

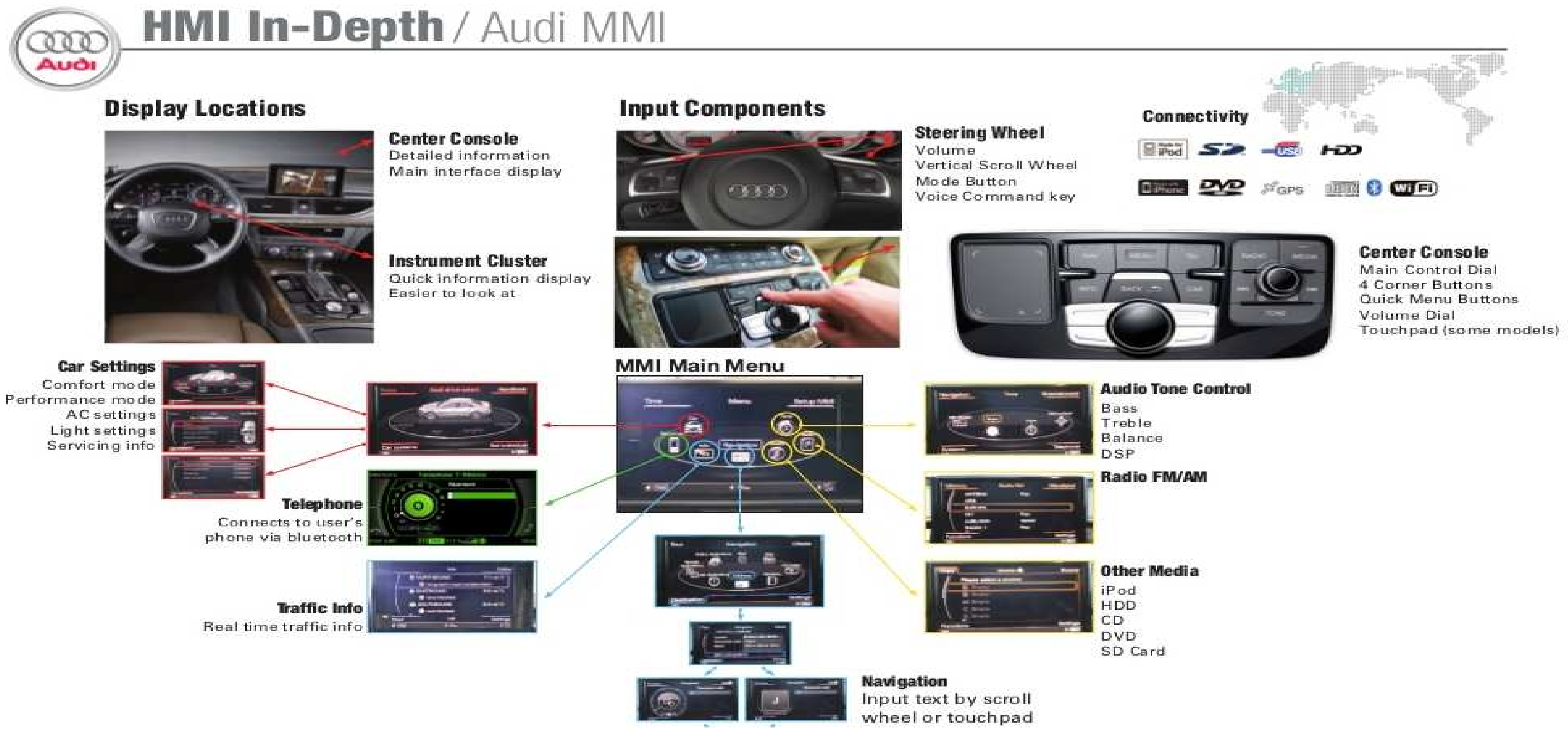
Output HMI technologies inside vehicles

- Color display
- Instrument cluster display
- Text-To-Speech
- Head-up display



HMI

Human Machine Interface(HMI, HCI or MMI)



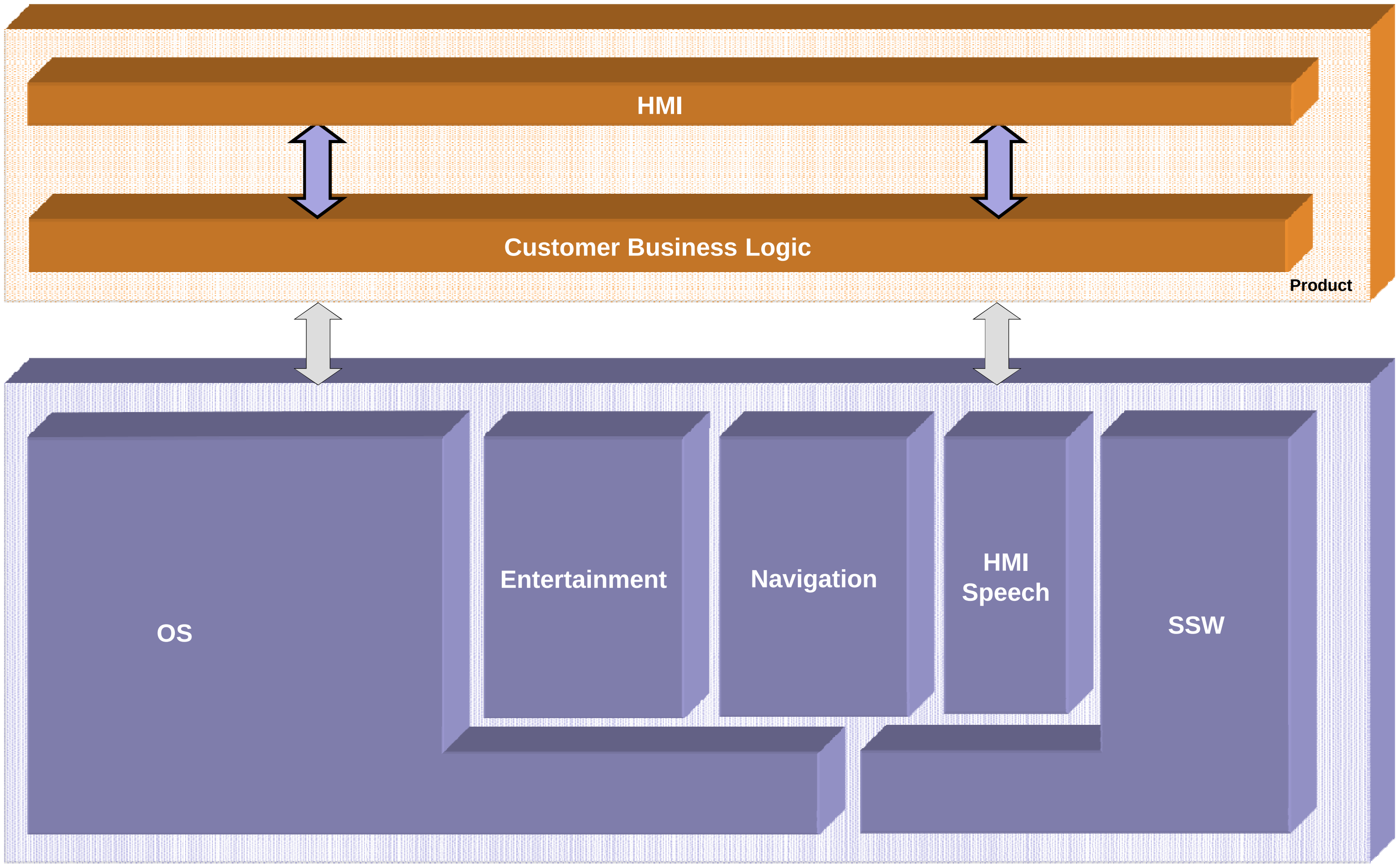
HMI design challenges

- Increased pressure for market differentiation
- Increasingly sophisticated customer expectation (multi-modal, dynamic HMI; consistency, ease of use and familiarity, location aware interface/content)
- Integration with external CE and services (cell phones, MP3 players, iPods...); updates
- Driver distraction
- Reduced development costs (HMI effort could be higher than 50% of a project's resources)
- Shortened design cycles
- Manage latest CE & desktop technologies (faster startup times, lower CPU prices, predictable response times, smoother animations)
- Personalization
- Platform development

Summary

- HMI
- **Automotive HMIs of today**
  - Introduction
  - **Architectural aspects**
  - GUI tool chains
  - GUI testing
- Automotive HMIs of the future

Multimedia Systems - Architecture Overview



Used Architectural Patterns

MVC (Model-View-Controller) – introduces a decoupling mechanism of an application three main components:

- **model** – the domain-specific representation of the data that the application stores, manages and presents
- **view** – the (visual) presentation of the application data
- **controller** – business-logic, the functional algorithms handling the information exchange between the model and the view

The pattern isolates the application data from business logic (BL), input and presentation (HMI), allowing for independent developing, testing and maintenance of each. Within the HMI borders, we may assume the following associations:

- The ***model*** is represented by the data coming from the actual business-logic (application specific)
- The ***view*** is **menu-oriented**, designed using the VAPS-XT tool-chain; it's usual for automotive HMIs to be organized as a tightly coupled menu-net (i.e. menu = the content displayed to the user at a certain point in time, composed by widgets).
- The ***controller*** describes the **bidirectional interaction** (commands and data-flow) necessary to handle the user input and to keep the HMI updated with the latest state of the application model. In VAPS-XT, the controller functionality is embedded in Calculator objects.

Used Design Patterns

1. Observer (publisher/subscriber) – an object storing a value or performing an action (the *subject* or *publisher*) keeps a list of *observers* (*subscribers*) and notifies them automatically of any state changes (i.e. value change, operation progress). The MMP HMI implements this pattern within the **Data Access Components (DAC)** – a set of reusable components designed to ensure:

- consistent communication between any HMI (GUI, SUI) and the underlying business-logic (BL)
- decoupling between BL and HMI threads (i.e. MMP uses a single-threaded HMI)
- HMI is immediately informed of any BL change via the notification mechanism (the HMI is observer for BL subjects)

2. Factory method – The *factory* creational pattern defines an interface for creating an object, but let the subclasses decide which class to instantiate. The purpose of factory method is to create objects without specifying (i.e. through its signature) their exact class. The MMP HMI implements this pattern for the Data Access. The factories are organized in 2 “levels”:

- **Application Factories** – one for each application (e.g. Media Player, Radio, Navigation); they provide concrete Control Factories to the HMI upper layers
- **Control Factories** – provide actual controls (references) to the upper HMI layers that need to register to them as observers. The type of the controls is not known by the “client", as the factory methods signatures are based on IxxxControl interfaces.

3. Lazy initialization – the tactic of delaying the creation of an object, the calculation of a value, or some other expensive process until the first time it is needed.

Model Based Engineering - Definition

- **1. Find a language which describes your requirements or your solution in very precise way.**
  - The language should be easy to learn by the people who have to use it.

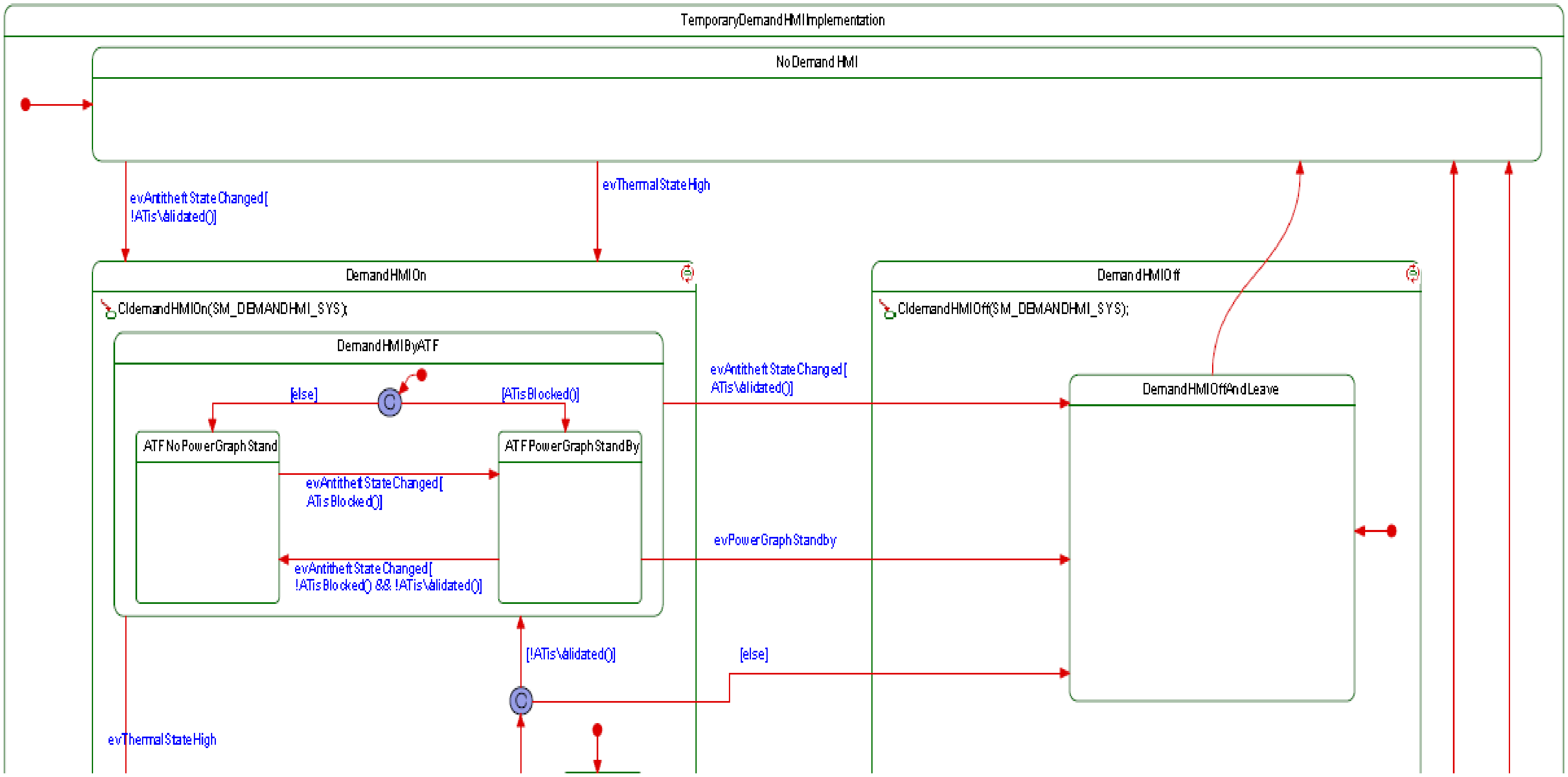
- **2. The language itself is based on some well known conventions**
  - Such conventions are also used for other languages.

- **3. Such a language is called "meta model"**
  - Based on such a meta model so called model instances are created.
    Examples from other areas:
    - Sewing pattern for the fashion industry. (Here a model is something totally different ;)
    - Circuit diagram for electronics
    - Blueprint for constructions

Useful applications of MBE in Automotive

- Defining behavior of a component using state machines
- Harel state charts are a standard for defining behaviors
- Customer specification is written as state machine
    - Comparing the diagrams is easier than comparing diagram to code.

Useful applications of MBE in Automotive


TemporaryDemandHMIImplementation
NoDemandHMI
evAntitheftStateChanged[!ATisValidated()]
evThermalStateHigh
DemandHMIOn
CIdemandHMIOn(SM_DEMANDHMI_SYS);
DemandHMIByATF
[else]
[ATisBlocked()]
ATFNoPowerGraphStand
ATFPowerGraphStandBy
evAntitheftStateChanged[ATisBlocked()]
evAntitheftStateChanged[!ATisBlocked() && !ATisValidated()]
[!ATisValidated()]
evThermalStateHigh
evAntitheftStateChanged[ATisValidated()]
evPowerGraphStandby
[else]
DemandHMIOff
CIdemandHMIOff(SM_DEMANDHMI_SYS);
DemandHMIOffAndLeave

MBE - Example for Model based Design - Statemachine

- Summary:
  - HMI requirements specifications vs. implementation approaches
  - manual code implementation vs. modeling – generated code
  - Specific test cases where Rhapsody state machine comes in handy
  - Benefits of Rhapsody features in state machine approach
  - Code generation
  - Simulations

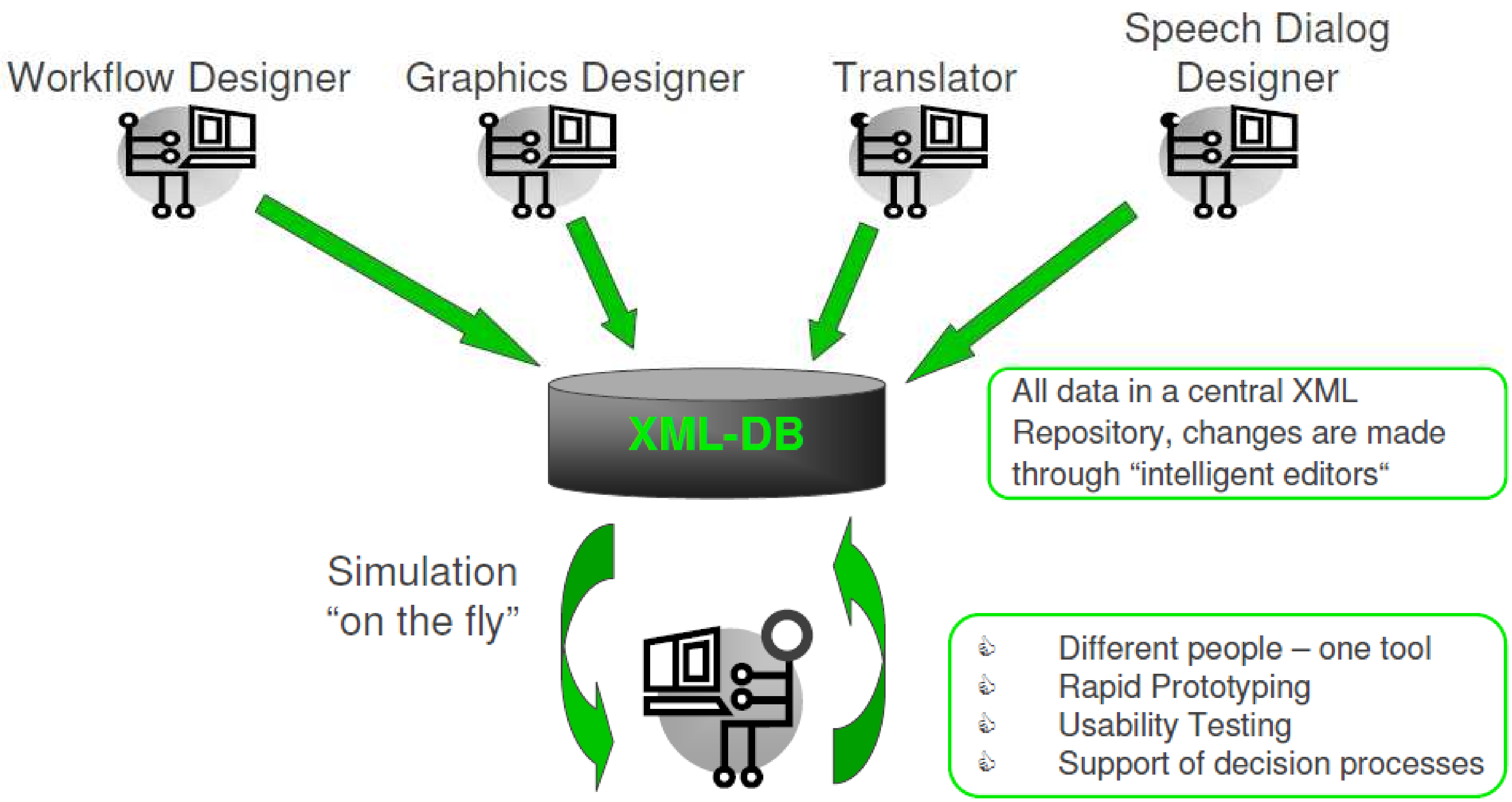
Summary

- HMI
- **Automotive HMIs of today**
  - Introduction
  - Architectural aspects
  - **GUI tool chains**
  - GUI testing
- Automotive HMIs of the future

Graphical User Interface – Tool-chains – Why?

- **Cost and time saving** HMI development process
- Rapid prototyping, simulation, specification and code generation in **one tool chain**
- Short cycles for **change requests** within iterative software engineering
- **Early** detection of problems / early testing / early stable decisions
- **Tool plugins** for functional extensibility
- Automatically **generated code**
- Adaptation to **different** operating systems, embedded graphic libraries and hardware platforms

Integrated HMI Development with EB GUIDE



All data in a central XML Repository, changes are made through "intelligent editors"

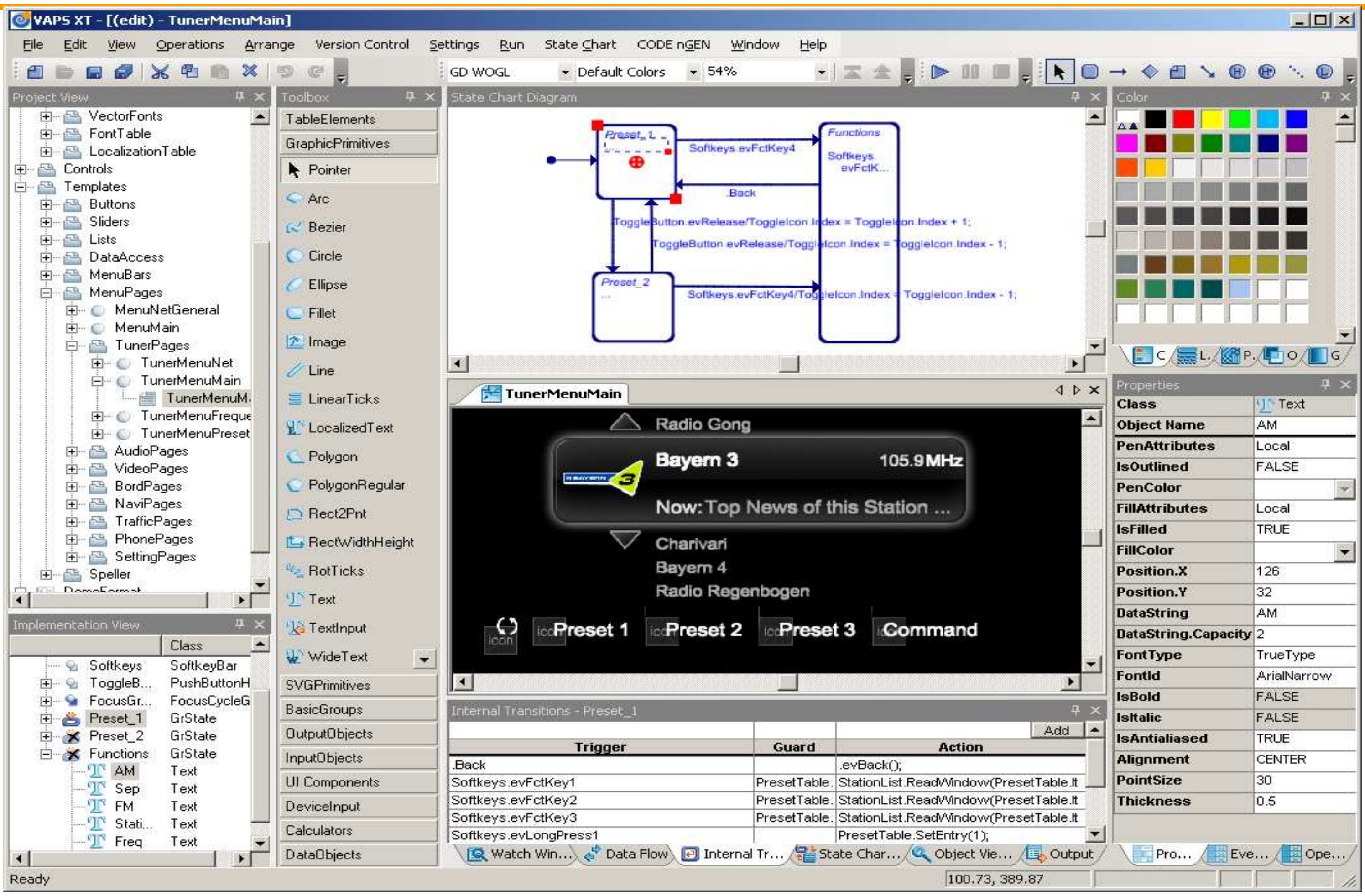
- Different people – one tool
- Rapid Prototyping
- Usability Testing
- Support of decision processes

Continental

Microsoft Silverlight for Microsoft Windows Embedded

- Silverlight for Windows Embedded (SWE) is a C++-based framework used for the creation of user interfaces of Windows Embedded CE applications.
- allows the separation of design and application logic.
- the designer creates the user interface including animations using Microsoft Expression BlendTM.
- Silverlight for Windows Embedded has been available since October 2009, with the release of Windows Embedded CE 6.0 R3.

GUI Tool-chains – VAPS-XT IDE



Implementation View

	Class
Softkeys	SoftkeyBar
ToggleB...	PushButtonH
FocusGr...	FocusCycleG
Preset_1	GrState
Preset_2	GrState
Functions	GrState
AM	Text
Sep	Text
FM	Text
Stati...	Text
Freq	Text

Internal Transitions - Preset_1

Trigger	Guard	Action
.Back		.evBack();
Softkeys.evFctKey1	PresetTable.	StationList.ReadWindow(PresetTable.It
Softkeys.evFctKey2	PresetTable.	StationList.ReadWindow(PresetTable.It
Softkeys.evFctKey3	PresetTable.	StationList.ReadWindow(PresetTable.It
Softkeys.evLongPress1		PresetTable.SetEntry(1);

Properties

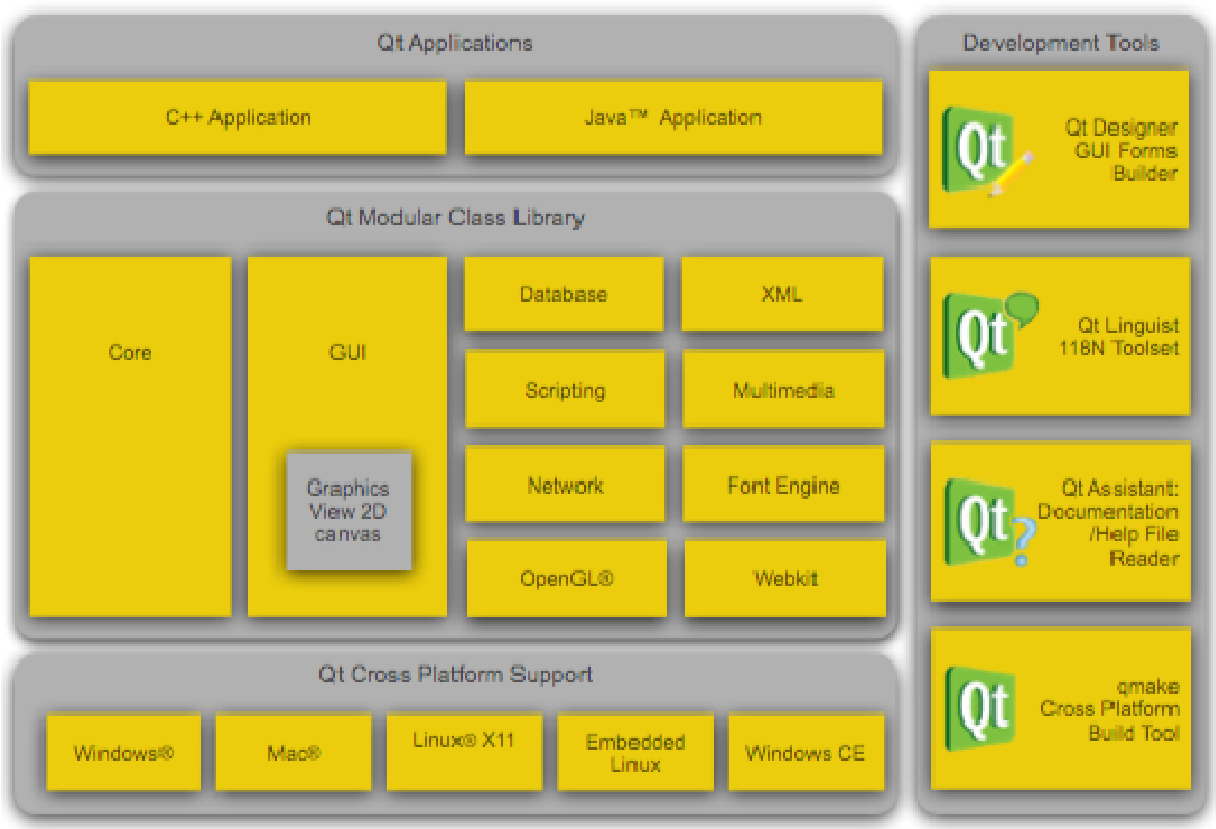
Class	Text
Object Name	AM
PenAttributes	Local
IsOutlined	FALSE
PenColor	
FillAttributes	Local
IsFilled	TRUE
FillColor	
Position.X	126
Position.Y	32
DataString	AM
DataString.Capacity	2
FontType	TrueType
FontId	ArialNarrow
IsBold	FALSE
IsItalic	FALSE
IsAntialiased	TRUE
Alignment	CENTER
PointSize	30
Thickness	0.5

Qt – open source SDK

- Qt is a cross-platform application and UI framework for developers using C++ or QML, a CSS & JavaScript like language. Qt Creator is the supporting Qt IDE
- Started in 1994 for Linux
- 10 different platforms (Windows | Mac | Linux/X11 | Solaris | Embedded Linux | Windows Embedded | Green Hills Software INTEGRITY | QNX | VxWorks)
  - used in desktop environments (Skype, KDE & KOffice suite, etc) as well as embedded devices (Nokia, MeeGo, etc)
  - currently, working on support for iOS and Android
- Large supporting community (http://qt-project.org/)
  - Increased bug fix yield
  - New features with every release (~3 months)
  - Covering a multitude of disciplines (GUI, Media, Navigation, Connectivity modules)

Qt is a cross-platform application and UIframework. It includes a cross-platform class library, integrated development tools and a cross-platform IDE.

Qt – open source SDK



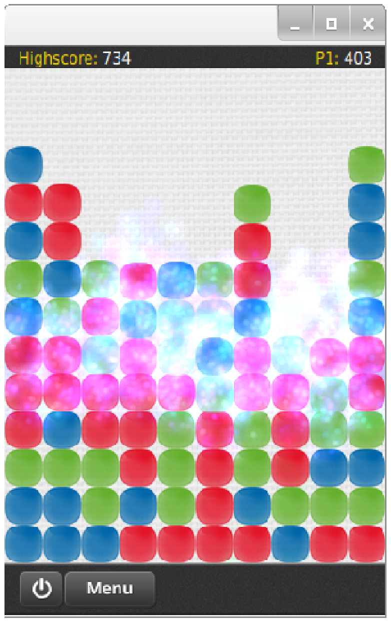
- Qt public APIs
  - Cross-platform
  - Modular
- Platform support classes
  - Optimization per platform
- Qt support tools
  - QtCreator IDE
    - General Qt development
  - Qt Designer
    - Design desktop applications
  - Qt Linguist
    - Manages internationalization issues

Qt – open source SDK

- QML
  - Declarative language used for widgets design
  - Supports Javascript
  - Powerful built-in features – image management, text management, animations, graphical effects, etc
  - Easy binding mechanism to C++ application code

```
Item {
     Rectangle {
          id: myRect
          width: 100
          height: 100
     }
     Rectangle {
          width: myRect.width
          height: 200
     }
}
```

QML engine



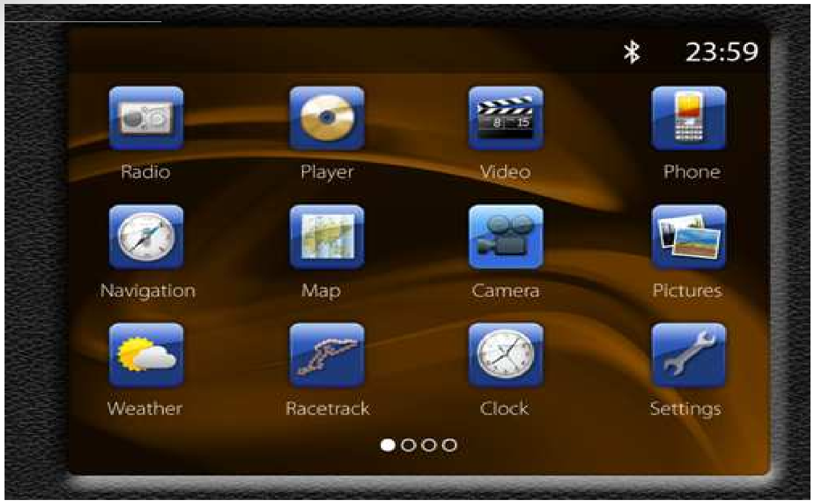
CES 2012 -2013

Las Vegas Demo Presentation – Open Infotainment platform

First Version of Linux / Genivi based platform

Key Messages/What's new

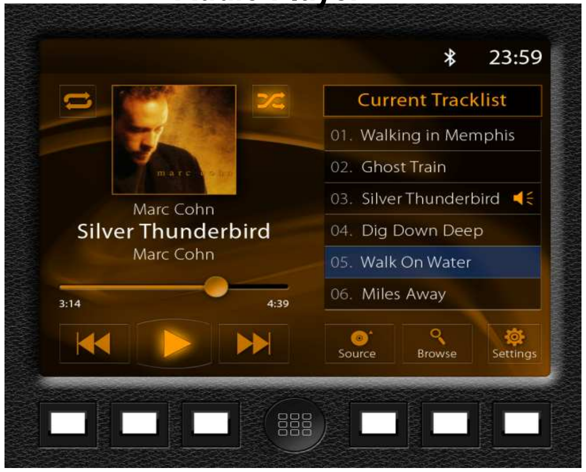
- Linux-based platform / GENIVI compliant
- HMI framework based on Qt
- HTML 5 web browser, HTML5 based applications
- Content in the Cloud – feature technology
- Qt support for wayland
- Touch Screen support



CES 2012

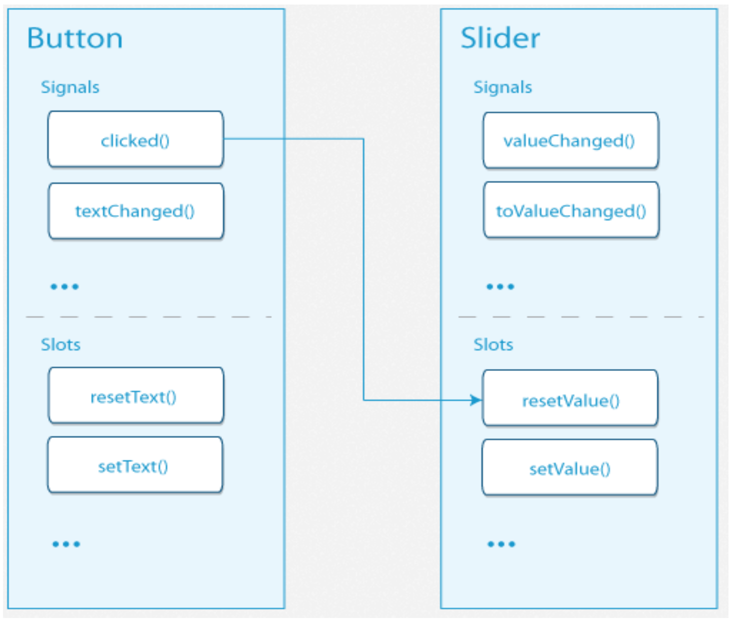
Las Vegas Demo Presentation – Open Infotainment platform

Audio Player



- Screen Design defined in QML
- Widget definition in QML(ex: SliderWidget, ListWidget, DropDownListWidget, ScrollBarWidget, ButtonWidget, …)
- Use QT signals/slots mechanism
- Application logic defined in C++
- Expose C++ classes to QML
- C++/QML binding mechanism

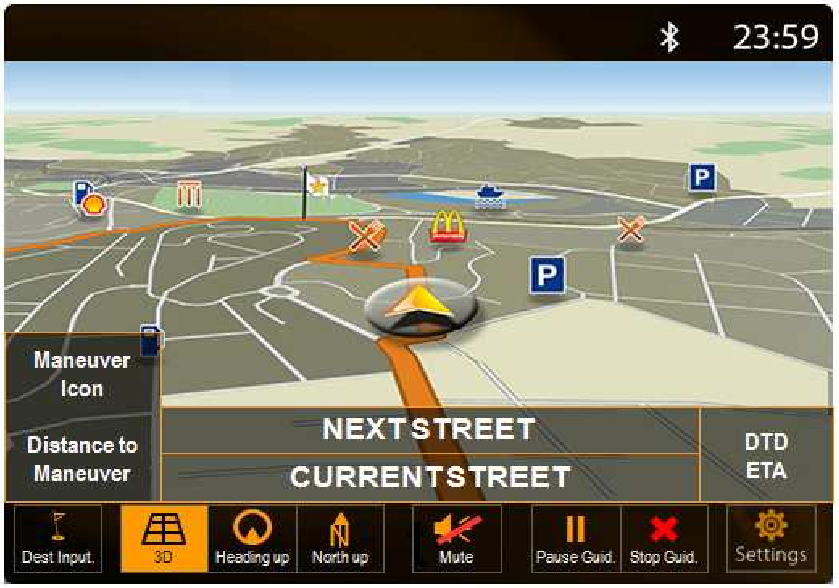
Video Mode



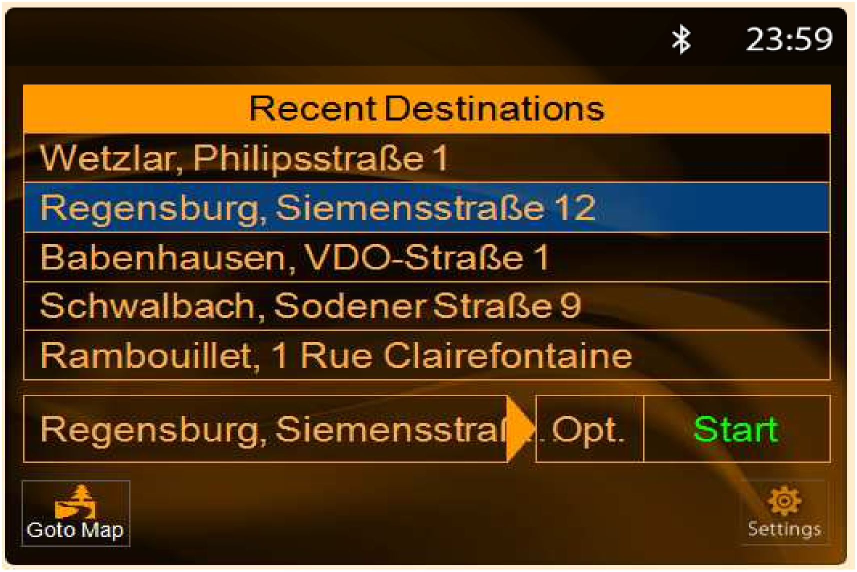
CES 2012

Las Vegas Demo Presentation – Open Infotainment platform

Navigation – Map View



Navigation – Destination List



Recent Destinations

Wetzlar, Philipsstraße 1
Regensburg, Siemensstraße 12
Babenhausen, VDO-Straße 1
Schwalbach, Sodener Straße 9
Rambouillet, 1 Rue Clairefontaine

Regensburg, Siemensstra...

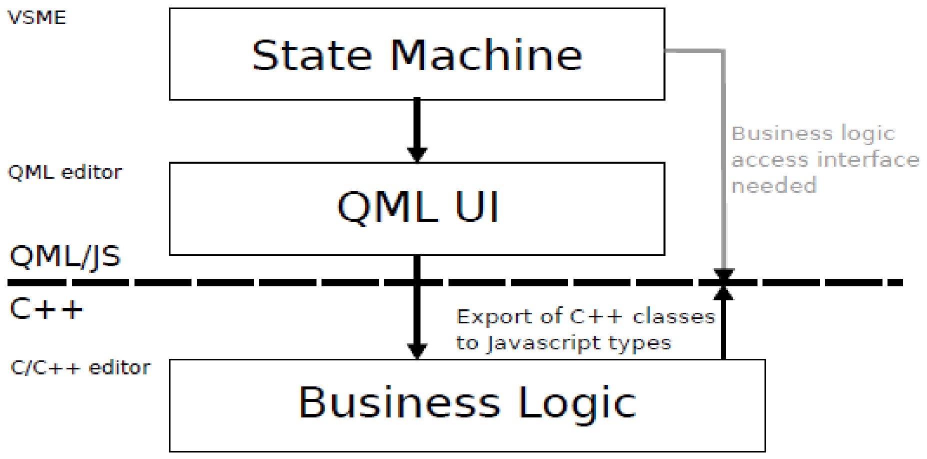
GUI Tool-chains – Qt Creator IDE & Automotive Modeler

QT support for Automotive HMIs

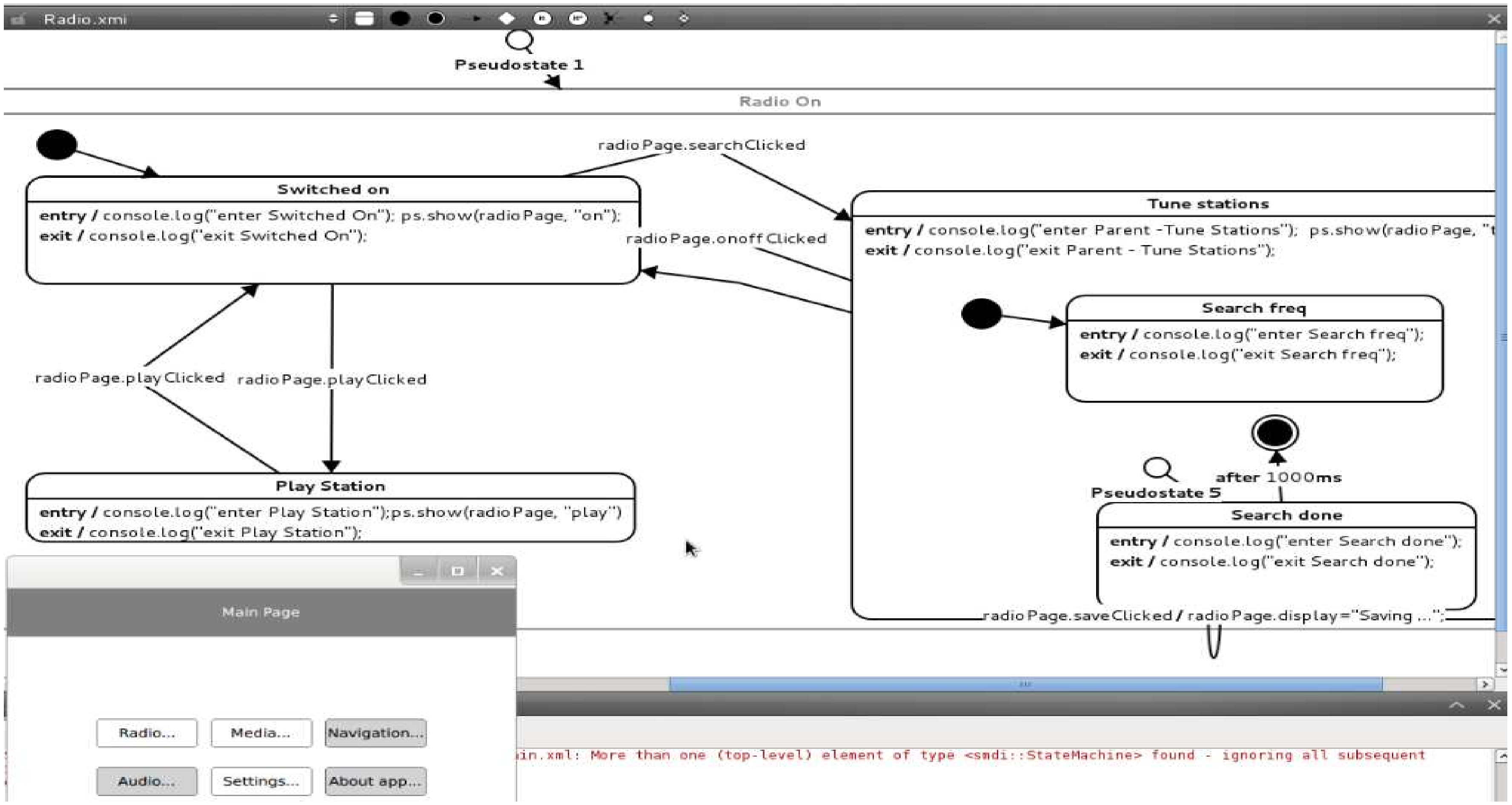
- Qt Creator - IDE for QT HMI development.
- The Automotive Modeler is a QtCreator Plugin

Automotive Modeler Plugin

- Visual IDE used to model the dynamic behavior of an HMI.
- The dynamic behavior of an HMI is modeled in a state machine
- Based on UML 2.0
- Connection to QML Screen Design & Function calls
- The Statemachine is interpreted during runtime
- Debugging & Simulation of Statemachine


VSME
State Machine
QML editor
QML UI
Business logic access interface needed
QML/JS
C++
Export of C++ classes to Javascript types
C/C++ editor
Business Logic

GUI Tool-chains – Qt Creator IDE & Automotive Modeler



```
Switched on
entry / console.log("enter Switched On"); ps.show(radioPage, "on");
exit / console.log("exit Switched On");

Tune stations
entry / console.log("enter Parent -Tune Stations"); ps.show(radioPage, "t
exit / console.log("exit Parent - Tune Stations");

Search freq
entry / console.log("enter Search freq");
exit / console.log("exit Search freq");

Play Station
entry / console.log("enter Play Station");ps.show(radioPage, "play")
exit / console.log("exit Play Station");

Search done
entry / console.log("enter Search done");
exit / console.log("exit Search done");

radioPage.saveClicked / radioPage.display="Saving ...";
```

```
in.xml: More than one (top-level) element of type <smdi::StateMachine> found - ignoring all subsequent
```

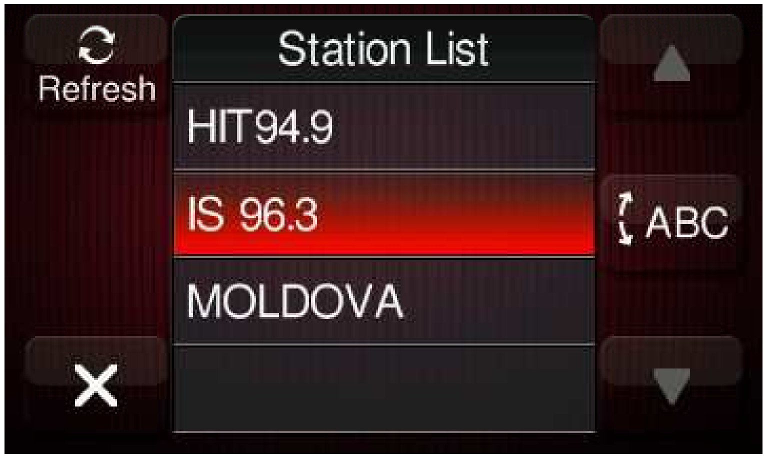
Summary

- HMI
- **Automotive HMIs of today**
  - Introduction
  - Architectural aspects
  - GUI tool chains
  - **GUI testing**
- Automotive HMIs of the future

HMI TestingTest Aspects

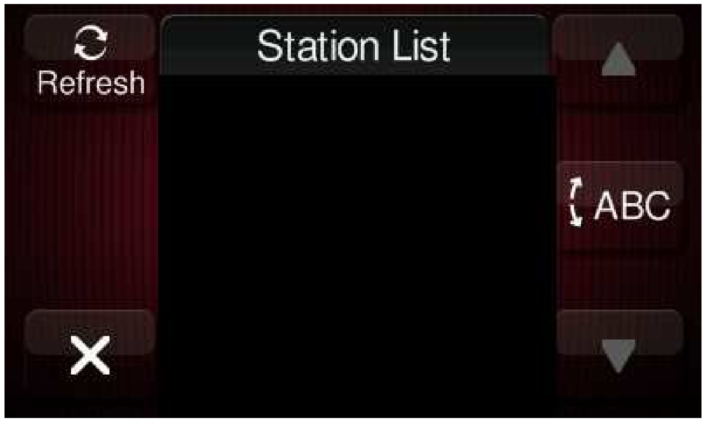
Each Requirements has different aspects which will be tested in different areas

- Example: *Radio list which is controlled via touch screen*
  - **Module Test**: Test the screen layout and possible transitions.
  - Integration Test: Test the application binding that list of receivable stations is shown
  - System Test: Test the touch screen that it is possible to select a certain station and audio is   hearable



HMI Testing

- Visualisation
  - Test framework which supports the following functionality:
    - Execute via module test a functionality to compare the current screen via a reference screen
      - Define the content of the application MOCKs
      - Request the screen to be shown in the runtime
      - Compare the created image with the corresponding reference screen
- Internationalization test
  - Select a certain language or certain dynamic text
  - Repeat „Visualization Test“



HMI Testing

- Behavior
  - Test framework which supports the following functionality:
    - Execute via module test a functionality to trigger certain input events and compare the transition via expected transition
      - Request the source screen to be shown
      - Stimulate a given event
      - Compare the target screen with the expected value

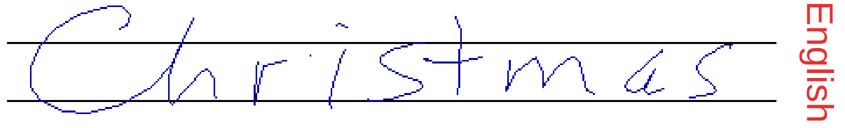
- Results of these tests will be counted to create screen test coverage

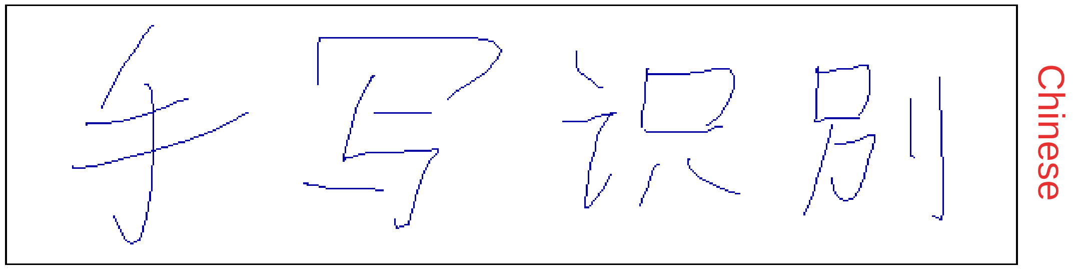
Summary

- HMI
- Automotive HMIs of today
- **Automotive HMIs of the future**
  - **Future HMIs**
  - Future technologies

Handwriting

- Why
  - For many languages is more appropriate than using virtual keyboard:
    - Simplified Chinese ≈6000 symbols
    - Traditional Chinese ≈13000 symbols.
- Where:
  - Phone dialing - digits or name selection
  - Navigation – POIs or addresses.
  - List selection.
  - Emails, SMSs.

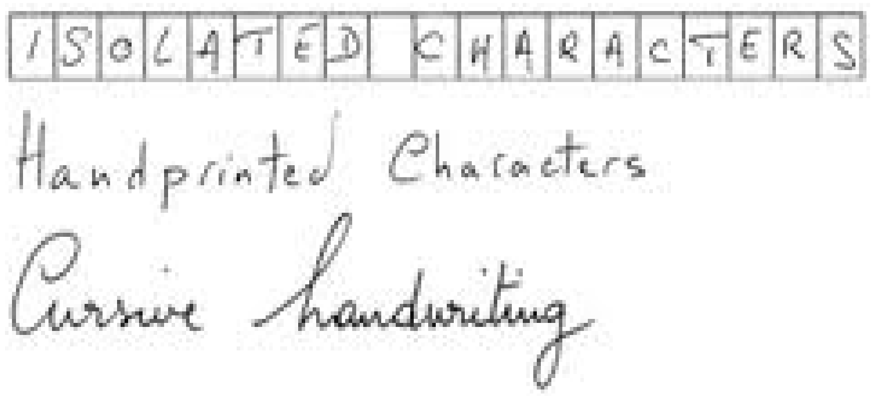




Handwriting

- How it works:
  - Character by character – isolated writing.
  - Word recognition with dictionary support.
  - Sentence recognition:
    - Hand-printed writing
    - Cursive writing
- Challenges:
  - Driver distraction.
  - Asynchronous recognition.
  - Performance.

Isolated vs Handprinted vs Cursive

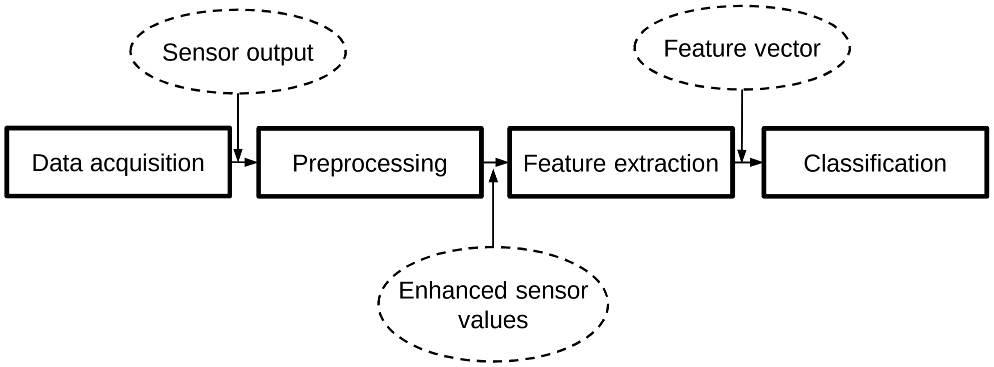


Gesture Recognition

- Gesture is a natural way of communication beneath speech and haptics.
- Hand gestures are non distracting – increase both comfort and driving safety.
- Operate in-car devices:
  - Accept or reject phone calls
  - Skip between radio stations, media files, switch between menus;
  - Scroll large texts (SMSs, Emails).
- Simple hand gestures: Left, Right, Up, Down, Stop.
- Based on HMM (Hidden Markov Models).
- Challenges:
  - Training own gestures.
  - Performance.

“Right” gesture

Pattern recognition pipeline



Other HMIs

Attention Assist

- Sensor-based system that monitors the degree of tiredness of the drivers
- If fatigue signs detected, the driver is visually and acoustically alerted: "Attention Assist. Break!".

Head-Up Displays

- First designed for airplanes, currently being developed in order to provide driver information without distracting the field of vision (transparent display)
- Information displayed: speed, distance traveled, infotainment settings and navigation guidance, visual alerts from ADAS
- Disadvantages:
  - dangerous evolution that might lead eventually again to driver distraction
  - light efficiency

Summary

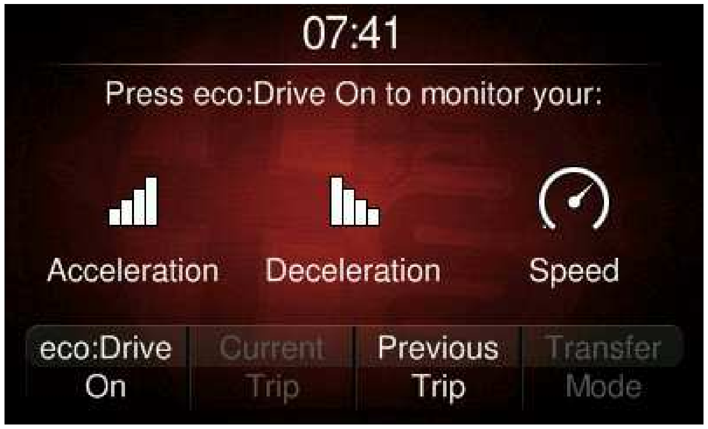
- HMI
- Automotive HMIs of today
- **Automotive HMIs of the future**
  - Future HMIs
  - **Future technologies**

HTML 5 in Automotive

- Why HTML5?
  - Cross platform, cross device (mobile, desktop, TV).
  - Widely known by developers.
  - Supports Multimedia (audio / video), 2D / 3D graphics, Vectorial graphics (SVG) and animations, Web GL based on Open GL ES 2.0.
  - Offline applications.
  - Runs IN or OUT-of-browser (standalone apps).

HTML 5 in Automotive

- Reliable for 3rd party applications:
  - Eco-drive.
  - In-vehicle networking, TV, 3D games, publishing
  - More secure.
- Challenges:
  - Standardization of Web API to access in-car subsystems.
  - Driver distraction.
  - Needs optimizations to match performance.



Questions?

Referinte

- BMW L7 project
- VP2 Project
- MMP Project
- “Valid Useful User Experience Measurement” – 2008 Nigel Bevan