# Understanding Mobile Apps for Automotive

Kerry Johnson, Senior Automotive Product Manager
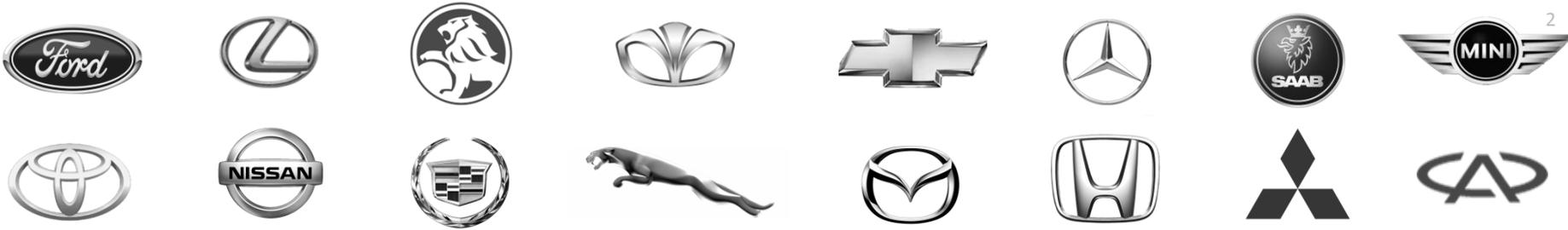
QNX in automotive?

In 2011

- 60+% of infotainment systems shipped
- 9+ million world wide (5+ million in NA)
- 40+% of all cars sold in US

Automaker Primer
What an app author needs to know about the automotive market in 10 minutes

The Lingo

- **OEM** = Automaker
  - GM, BMW, Audi, Toyota, Chrysler, etc

- **Tier1** = Module supplier
  - Delphi, Harman-Becker, Denso, Panasonic, Visteon, etc

- **Tier2** = Software/hardware supplier
  - QNX, Freescale, TI, Pandora, Red Bend, etc

- **RFI, RFQ, RFP**
  - Request for info, quote, proposal

- **Head unit, center stack**
  - Navigation hardware module

- **Infotainment**
  - Information + entertainment + nav + telematics + hands-free

Infotainment and telematics examples

- Audi MMI 3G
- BMW ConnectedDrive
- Chevy MyLink
- GMC/Buick IntelliLink
- OnStar
- Toyota Entune
- Toyota Touch&Go
- Hyundai Blue Link
- (MyFord Touch—not us!)

Consumer vs OEM lifecycle

Applications

Smartphones

Vehicle electronics

SOP   Year 5   Year 10
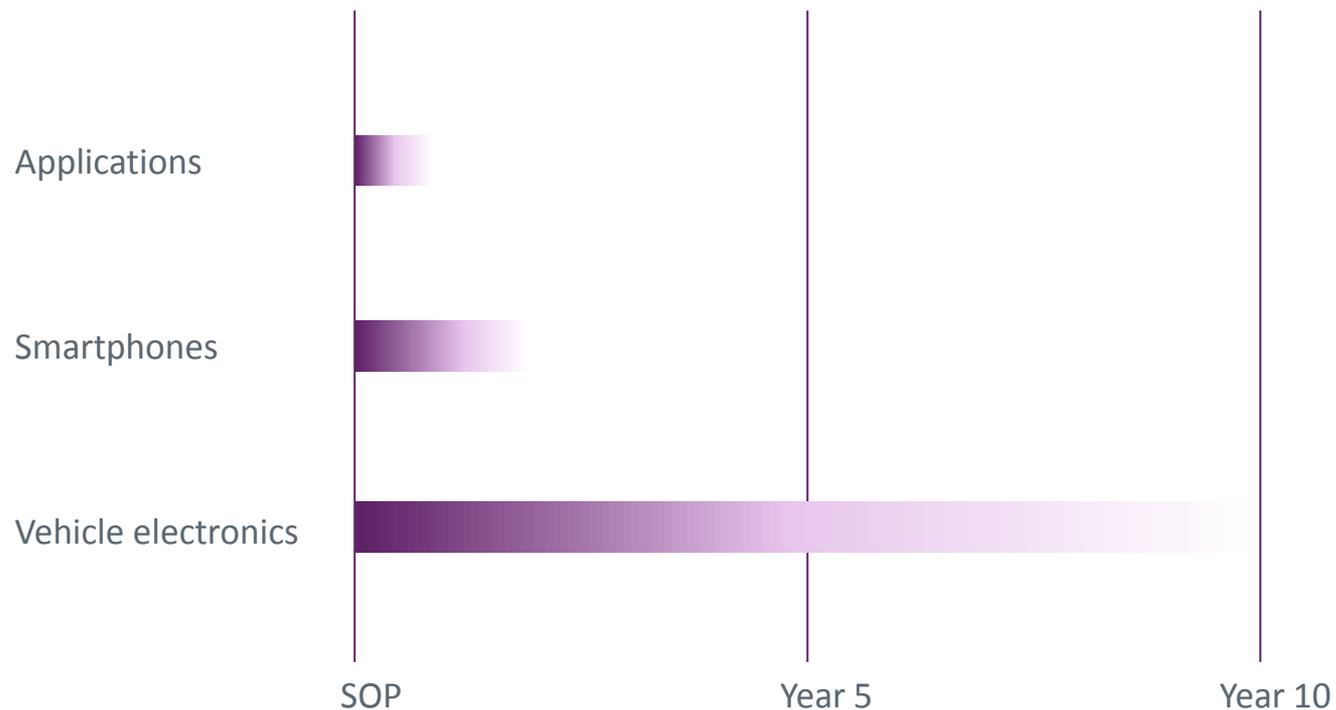
Development takes 3 years!?!

1. Lemons

2. Lawyers

Driver distraction
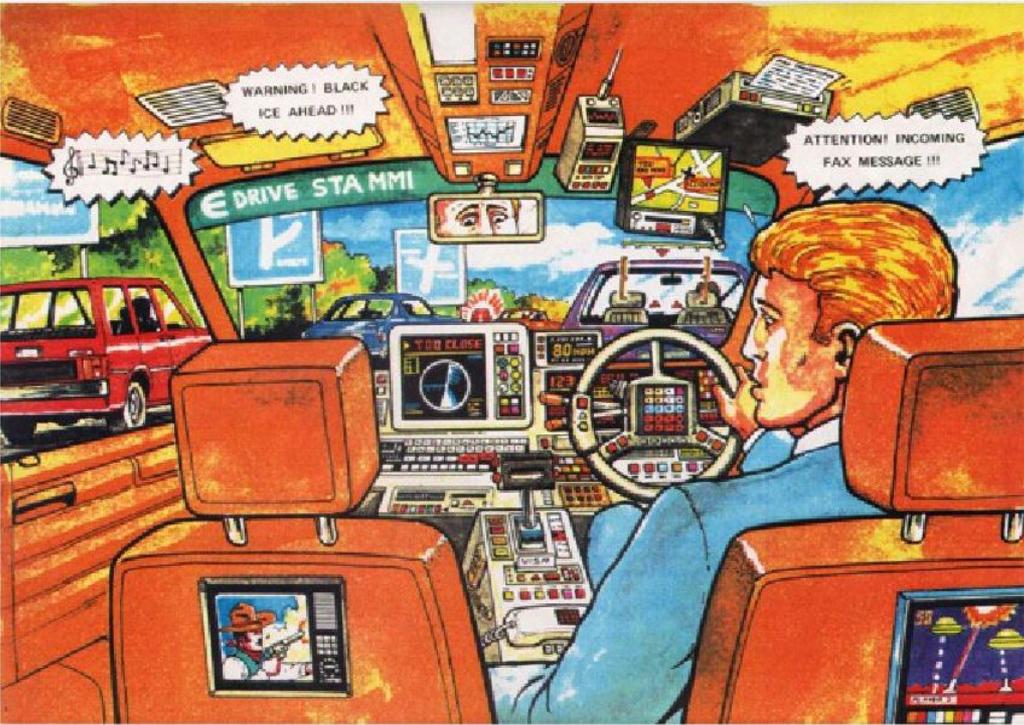
Cars: Where are they? Where are they going?

Head-units yesterday

- 1930: First commercial car radio
- 1952: First FM radio
- 1955: First 'music-on-demand'
- 1963: First transistor radio
- 1965: First eight-track
- 1970: First cassette player
- 1985: First CD player

Head units today

- Typical hardware
  - 32-bit CPU w/MMU: ARM (single, dual and quad core Cortex A8/A9 for mid- to high-end modules)
  - RAM & flash: 16/32MB (hands-free) to 256/512MB (infotainment)

- Infotainment system features
  - Hands-free calling
  - Speech recognition
  - Navigation system
  - Satellite + Internet radio
  - Mobile connectivity

Head units tomorrow

Same features as today, but add:

- Add (car-relevant) social networking

- Extend mobile connectivity
    - Import apps from phones
    - View apps running on phones
    - Share hybrid app between car and phone

- Run apps and app stores

- Connect to cloud-based services

Why HTML5 for auto?

- Cross-platform
  - deploy apps on phones and car
  - extend by using mobile developers, tools, companies
- Flexibility
  - flow deployment and architectures together as needed
  - brand same app on different cars with CSS
- Time-to-market
  - leverage the ease-of-use of high-level language
  - program in rich application environment
- Lifespan
  - get support for the lifecycle of the product
  - avoid vendor lock-in with standards

How can I build my app for the car?

Bringing CE into the car: Simple?

Bringing CE into the car: HARD!

- Nearly unlimited product liability and warranty

- 10+ year part and price availability

- Control of functionality

- Control over look and feel (depends on brand)

- Vehicle bus security

Apps for auto

- 30 not 30,000

- Driver distraction

- OEM validation

MirrorLink

VNC - screen content

cmd & ctrl

A2DP - audio

QNX
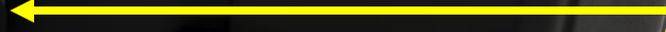
MirrorLink pros and cons

- Pros
  - Exists today
  - Based on standards and well-known technology
  - Has qualified auto industry support

- Cons
  - Interface served up by phone; not necessarily 'snappy'
  - OEM-unique acceptance
  - No centralized 'app store'

Putting my app in the car <u>today</u>

- Working with an OEM?
    - Exposure, exposure, exposure
    - Automotive app store

- Working with the QNX CAR application platform?
    - Making the link between auto OEM and the applications
    - Leverage mobile investment
    - HTML5 SDK

Parent Directory

blackberry.app/

blackberry.connection/

blackberry.event/

blackberry.identity/

blackberry.invoke/

blackberry.system/

blackberry.ui.dialog/

qnx.audiomixer/

qnx.volume/

qnxcar.hvac/

qnxcar.sensors/

WebWorks for QNX CAR platform

- Adding APIs for car-specific features
  - Vehicle bus features: HVAC, body, electrical systems
  - Audio system

- Underlying HTML5 engine provides hardware accelerated graphics
  - Webkit-based transitions optimized to make use of the NVIDIA Tegra GPU

Ripple emulator for QNX CAR platform

Driver distraction for the uninitiated

Distraction avoidance guidelines

High-level extraction from various guidelines

- AAM, USDOT, SAE, ITU-T

1. Minimize off-road glances

- Design app so only need glances of 2 seconds or less at a time
- Minimize visual information in driver's field of view
  - Remove clutter, animations, eye-candy, embellishments, etc.
- Intelligent use of ASR/TTS

Distraction avoidance guidelines

2. Minimize total task duration

– Don't make any one task (from action to achieved) take longer than 15 seconds

– Interruptions not counted

– Does *not* imply you can have eyes off road for 15 seconds...

3. Make applications interruptible

– Drivers should be able to pause, resume interaction

– Assume 2 second 'chunkability'

– If understanding how or where to resume takes >2 seconds, app will be unusable

Distraction avoidance guidelines

4. Make applications predictable
- – Easy to anticipate application behavior
- – Results easily understood
- – Bad example: shifting menus

5. Make applications ignorable
- – Non-immersive user interfaces
- – Limit manual input required
- – Bad example: VR system with timeout on no response

Putting my app in the car <u>tomorrow</u>

Crystal ball

- Don't expect in-car systems to disappear
  - Rear seat category will be phased out
  - In-car systems broader adoption

- OEMs slowly opening up
  - App sandboxes
  - Automotive SDKs

- Within North America, expect to see
  - Streamlined module validation
  - Streamlined app certification

Any questions?

Kerry Johnson

kjohnson@qnx.com

qnx.com/careers