



# Creating outstanding digital cockpits with Qt Automotive Suite

Get your digital cockpit first to the finish line with Qt.

Embedded World 2017

# Trends in cockpit digitalization require a new approach to user experience design and software development



1972



2017



**Speech Input**

**Head-up display**

**Digital instrument cluster**

**Sensor fusion**

**Touch Screen**

**Smartphone connectivity**

**Car-to-X communication**

**Augmented reality**

**Object detection**

**Internet connectivity**

**Gesture control**

**Autonomous Driving**

**Big Data & Analytics**

**Driver Assistance**

# OEMs are seeking ways to harmonize and accelerate the development of digital cockpits

## Harmonize IVI tool chain

- › Establish a common tool chain for IVI development across the company
- › Enable quick development cycles from early design through prototyping to target hardware
- › Use the same tools and assets for development of head units, instrument clusters, head-up displays, passenger screens, digital control panels, etc.

## Common development platform

- › Build a prototyping platform that can be used by user experience design & development teams across all bands and model series
- › Build and deliver an SDK as part of the platform
- › Build a library of UI assets reusable across projects and brands

## Customize environment

- › Get source code access to development tools
- › Customize and extend the development environment
- › Brand and customize the SDK



# Introducing the Qt Automotive Suite (QtAS)

## Pre-built assets



Neptune

Automotive reference UI enabling 3rd party app integration

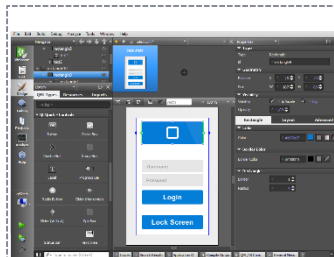
Input Fwk

Virtual keyboard; Easily integrate speech, gesture, handwriting, etc.

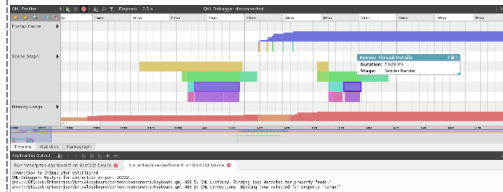
Qt IVI

Cross-platform automotive APIs (e.g. Vehicle Info+Config)

## Extensive tooling



Qt Creator is an integrated development environment (IDE) and central to building custom SDKs



QML Profiler for identifying rendering bottlenecks



GammaRay for deep UI introspection

## Strong Ecosystem

The Qt Automotive Suite is built and maintained together with strong community partners.



# Built on the leading C++ Cross-Platform Framework...



One Technology  
for All Platforms

Cross-Platform Class  
Library



Shorter  
Time-to-Market

Integrated  
Development Tools

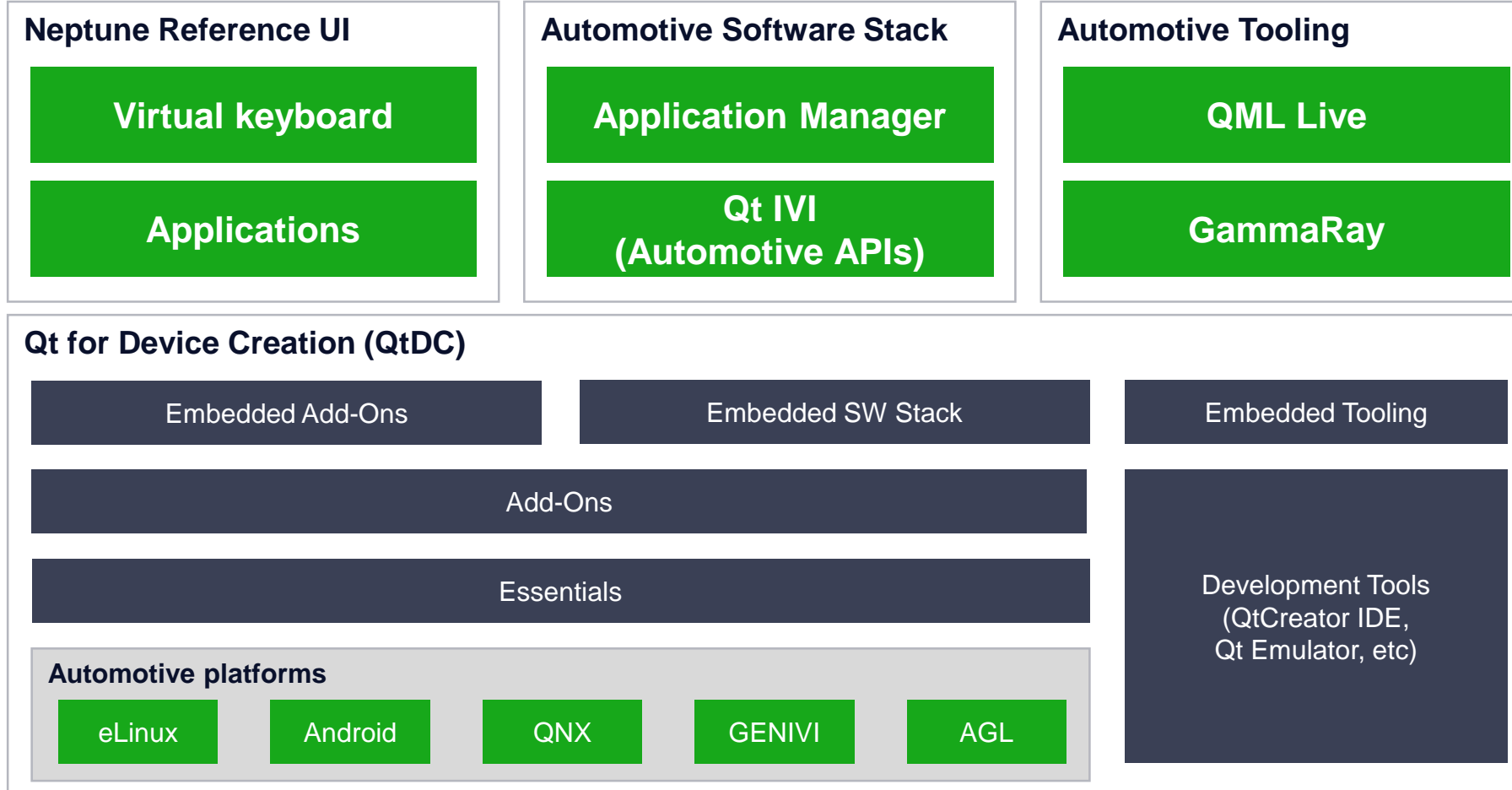


Productive  
development  
environment

Cross-Platform IDE,  
Qt Creator

Used by ~1 Million developers in 70+ industries

# ...the Qt Automotive Suite includes a set of pre-built components and specific tools for IVI development



## Value

- › Qt for device creation offering specifically enhanced for automotive customers
- › Pre-built components for rapid project start
  - › Reference UI
  - › Application Manager
  - › QtIVI
- › Embedded tooling
  - › Remotely deploy
  - › Remotely debug
- › Additional tooling pre-integrated with Qt Creator
  - › QMLlive
  - › GammaRay
  - › Emulator

# Application Manager & Compositor

## Create your unique software platform

- › Create a multi-process architecture
- › Improve robustness by splitting your screen area into different functional units / applications
- › Reduce risks and dependencies by eliminating monolithic HMI applications and enable independent teams & 3<sup>rd</sup> party app development

## The developer experience with productivity at its core

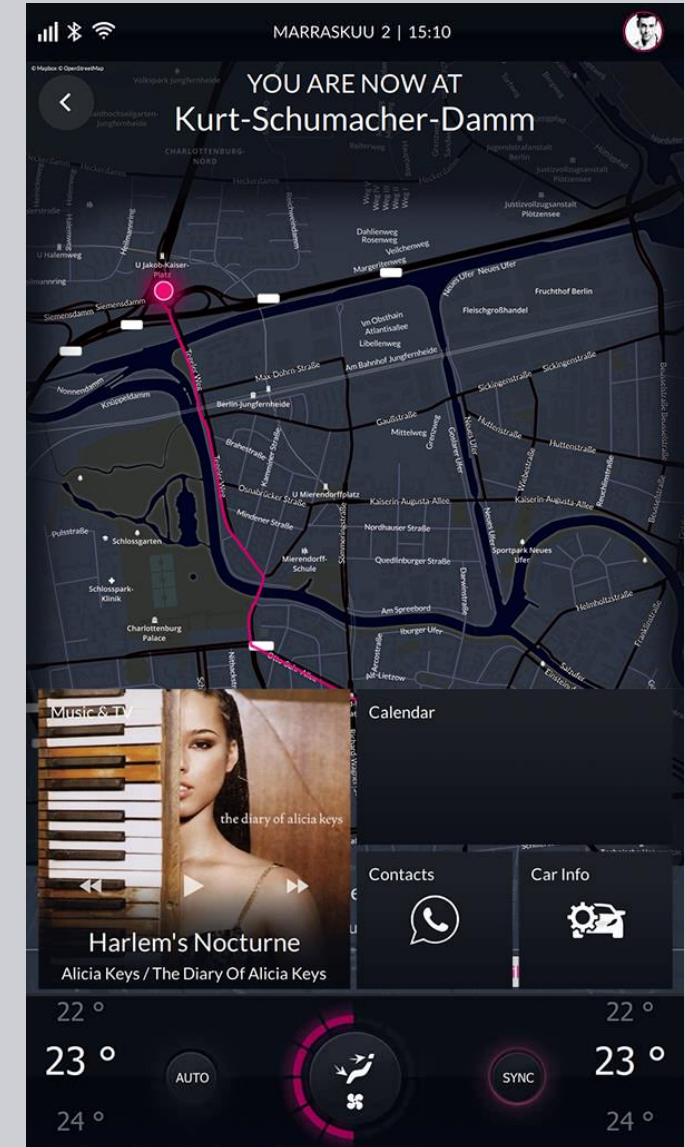
- › Less lines of code to test, debug and maintain - creating window servers on high abstraction levels
- › Enable simultaneous development & testing by separating your UI into different functional units
- › Minimize risk by only updating one unit at a time
- › Security model to protect the integrity of Wayland hardware accelerated compositing
- › Supports both OpenGL and HTML applications

Status

Application layer

Launcher

Static controls



# Qt IVI - Extensible Cross Platform APIs

- › Industry wide APIs
- › No OS/Middleware platform lock-in
- › Enables industry wide 3rd party app ecosystem
- › Security model to control app access to APIs
- › Emulator backend for desktop simulation



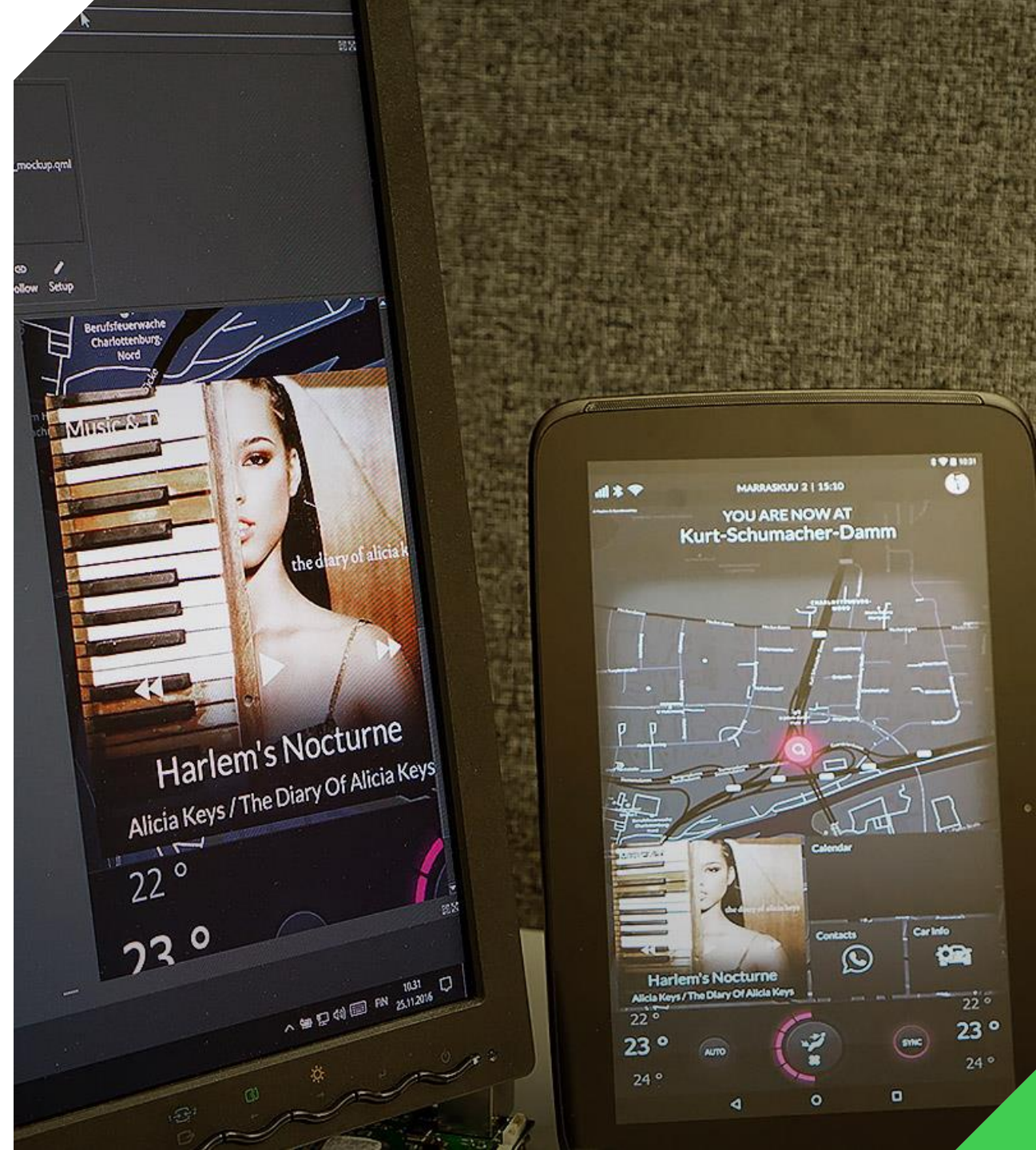


# Single click compile, download, run, debug to Target. No Target Hardware? No Problem with device emulation



# QML Live – Create a Better HMI by Iterating Designs Faster

- › Live reloading of QML code
- › Instantly see UI changes on target
- › Develop locally and simultaneously display and modify the UI on one or more local and/or remote clients
- › Facilitates rapid iterations of design and development
- › Faster iterations -> better HMI

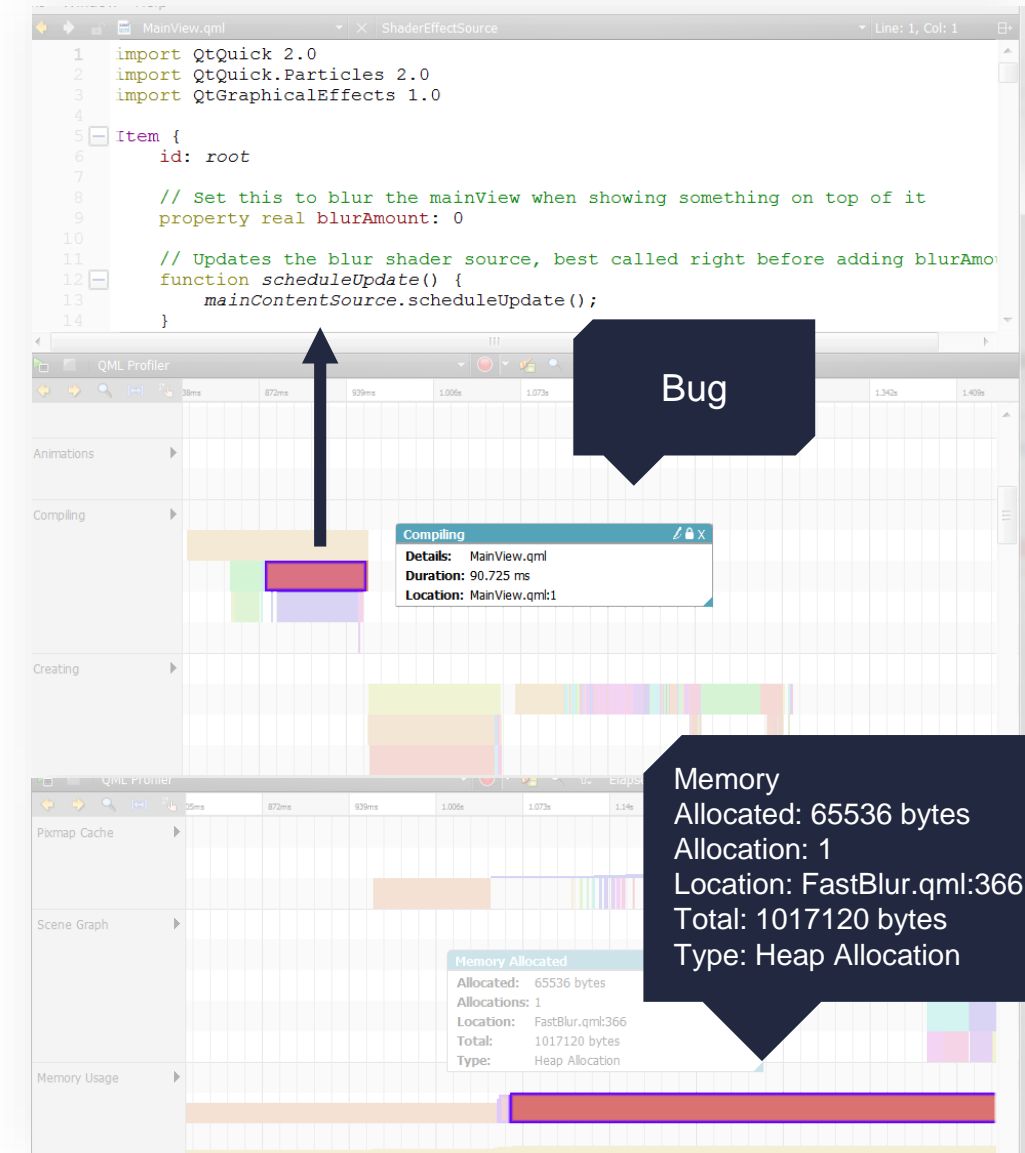




# QML Profiler

Debug performance issues in Real-Time

- › Typical causes
  - › Too much JavaScript
  - › Unnecessary workload in GUI thread for invisible items
  - › Heavy QML items to compile and create
  - › Long-running C++ functions
- › Easy to use
  - › Collecting data (2 options)
    - › Start a profiler from launching the application
    - › Attach to the running application
  - › Analyze Data
- › Analyze different aspects of performance e.g.
  - › Pixmap cache - Size of a memory allocation info per image
  - › Scene graph - Graphic rendering status information
  - › Memory usage - Memory usage profile of QML engine
  - › Input events - User input data profile
  - › CPU Usage Analyzer - CPU usage profile on embedded devices



# Maximize performance & uptime with GammaRay

- › Maximize uptime by finding & fixing bugs faster with high level debugger
- › Optimize your code with performance profiling
- › Visualize, Access and Modify properties of Qt objects at runtime
- › Over 20 Qt Objects Supported (Qt Quick, state machines, text layout, model/view, timers, etc)
- › No runtime agent

The screenshot displays the GammaRay debugger interface. On the left, a 3D scene is visible with a callout box labeled 'Bug B002' pointing to a specific object. On the right, the 'Raw Vertex Data' table is shown, which contains the following data:

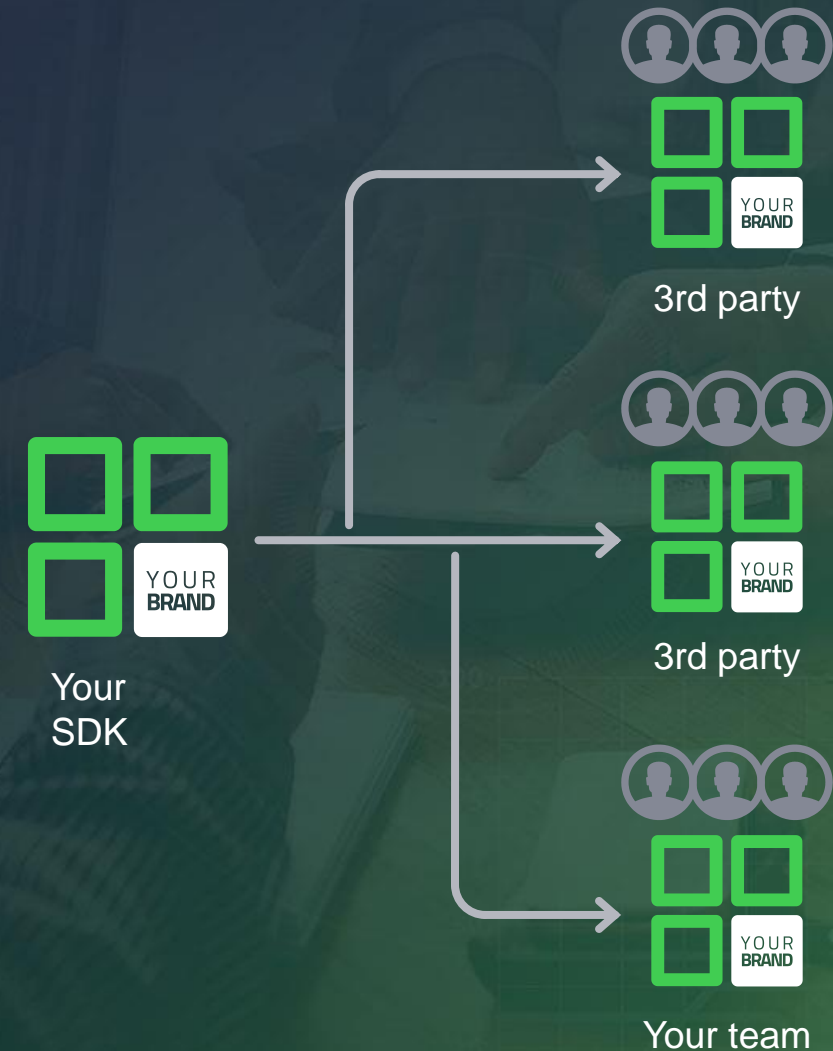
	vertex	vertexColor	vertexOffset
79	52.1899, 14.3656	0, 0, 0, 0	14.3656, 0
80	0.81007, 14.3656	0, 0, 0, 0	17.2963, nan
81	51.8066, 17.2963	255, 255, 255, 255	17.2963, nan
82	1.19335, 17.2963	255, 255, 255, 255	17.2963, nan
83	51.8066, 17.2963	170, 170, 170, 255	17.2963, nan
84	1.19335, 17.2963	170, 170, 170, 255	17.154, nan
85	52.7965, 17.154	170, 170, 170, 255	17.154, nan
86	0.20353	170, 255	17.154, 0
87	52.7965		17.154, 0
88	0.20353		20.0003, nan
89	52, 20.0003	255, 255, 255, 255	20.0003, nan
90	0.999998, 20.0003	255, 255, 255, 255	20.0003, nan
91	52, 20.0003	170, 170, 170, 255	20.0003, nan
92	0.999998, 20.0003	170, 170, 170, 255	20.0003, nan
93	53, 20.0003	170, 170, 170, 255	20.0003, nan
94	-1.90735e-06, 20.0003	170, 170, 170, 255	20.0003, 0
95	53, 20.0003	0, 0, 0, 0	20.0003, 0
96	-1.90735e-06, 20.0003	0, 0, 0, 0	46, nan
97	52, 46	255, 255, 255, 255	46, nan
98	1, 46	255, 255, 255, 255	46, nan
99	52, 46	170, 170, 170, 255	46, nan
100	1, 46	170, 170, 170, 255	46, nan
101	53, 46	170, 170, 170, 255	46, nan
102	0, 46	170, 170, 170, 255	46, 0
103	53, 46	0, 0, 0, 0	46, 0
104	0, 46	0, 0, 0, 0	48.704, nan
105	51.8066, 48.704	255, 255, 255, 255	48.704, nan
106	1.1934, 48.704	255, 255, 255, 255	48.704, nan
107	51.8066, 48.704	170, 170, 170, 255	48.704, nan

Callout boxes indicate the following bugs:

- Bug A001**: Points to row 86 in the 'Raw Vertex Data' table.
- Bug B002**: Points to a specific object in the 3D scene.
- Bugs C003 C004**: Points to a specific object in the 3D scene.

# SDK Creation

- › Create a branded SDK with project UI and software assets
- › Enable 3rd party app development and services
- › Ensure consistent look and feel of HMIs/apps with ready made components and style guide
- › Reduce risk and save time with a single preconfigured install with all necessary tools
- › Qt SDK + Program's HMI components + Program's software components → Program Branded SDK





# Qt SCXML

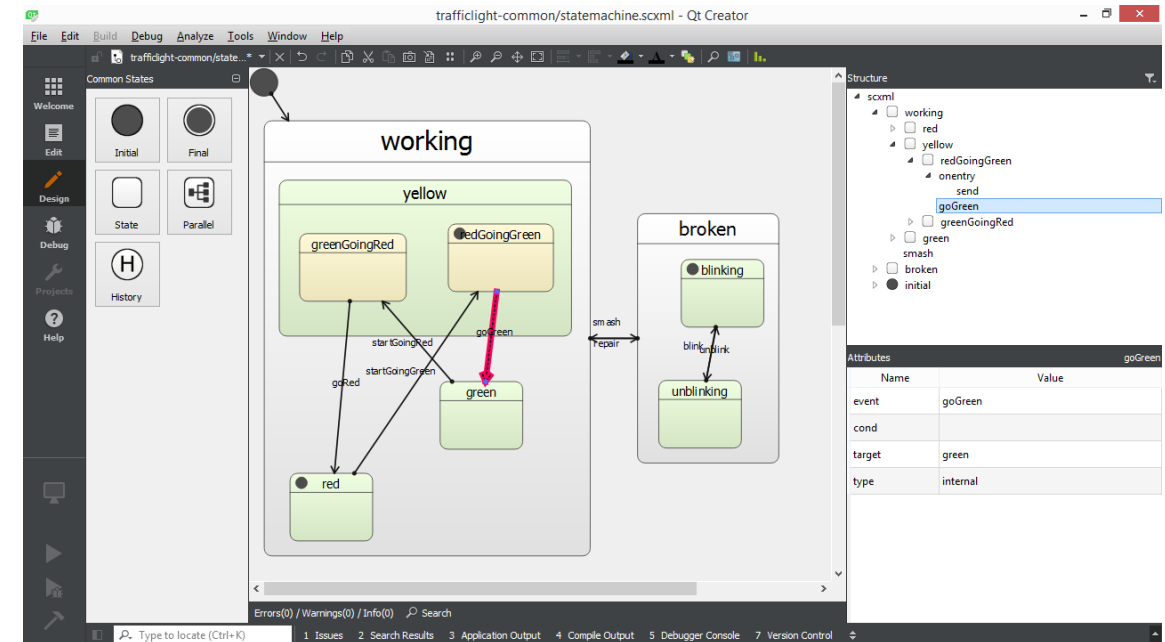
Reduce risk of unexpected system behaviour with fast & easy state chart integration

## Ultimate Performance, Reliability and Stability

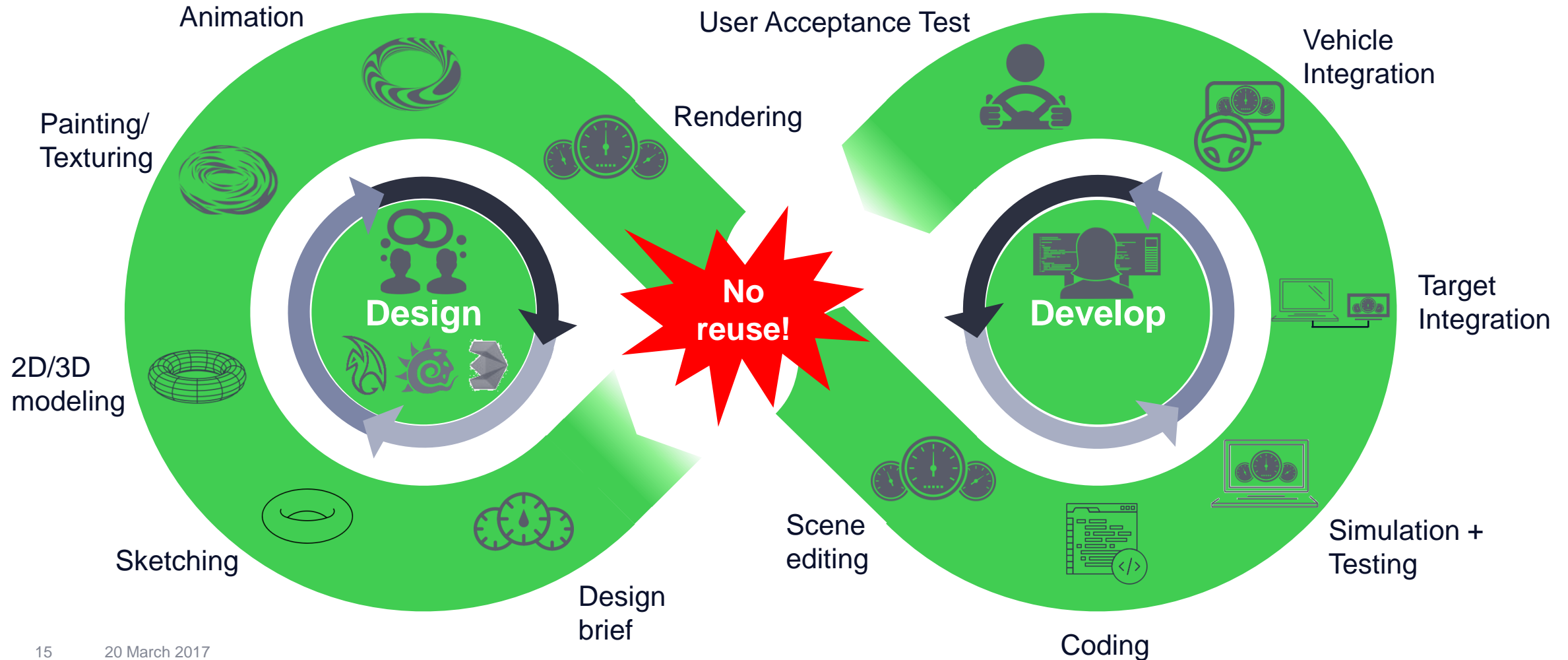
- › Reduce risk of unexpected behavior
- › Ensure maximum uptime
- › Fast and easy integration
- › Concise representation of business logic

## The developer experience with productivity at its core

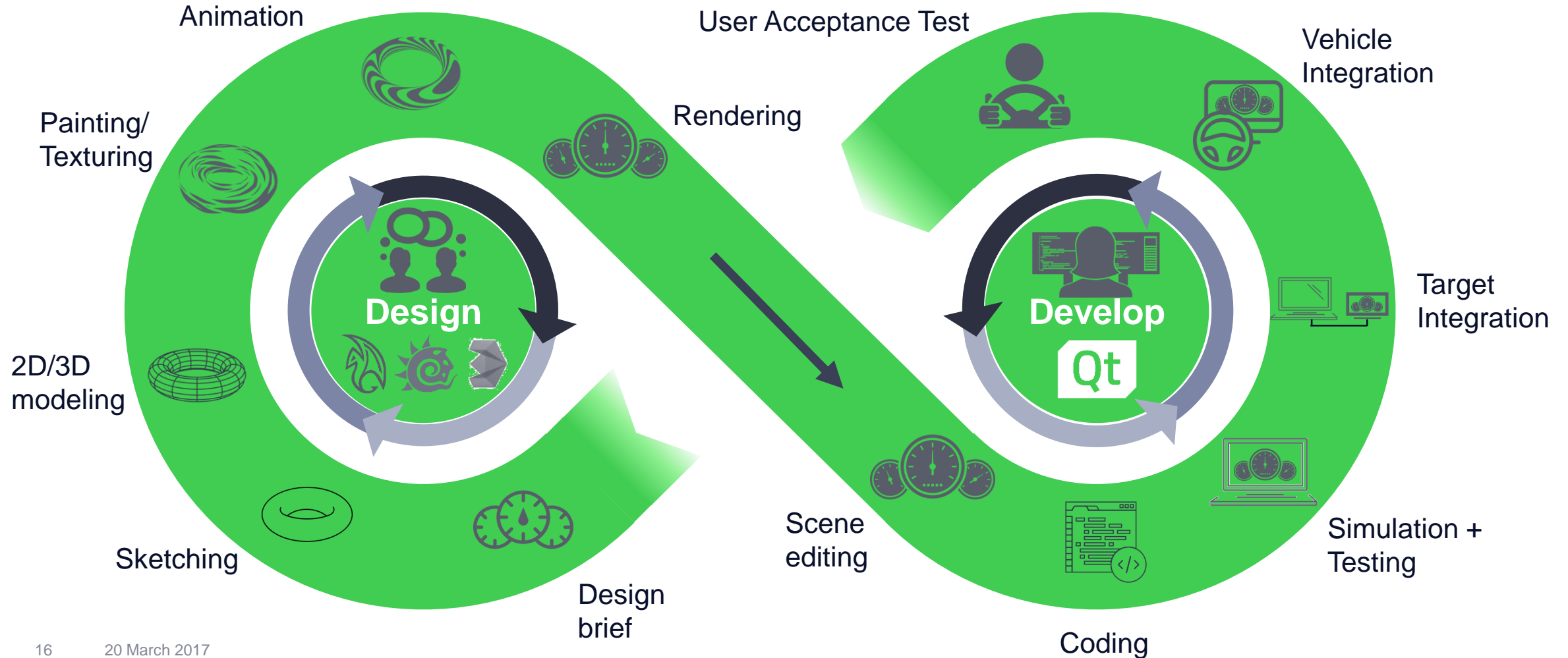
- › Formalize applications & validate workflows with SCXML to define state machines
- › Easily drive your state machine from a state chart rather than business logic expressed in C++ or QML
- › SCXML docs can be compiled into C++ and readily integrated into any Qt application
- › SCXML can also be dynamically loaded and interpreted at runtime



# The design and development cycles for digital cockpit systems are often disconnected



# Qt Automotive Suite closes the gap to allow for rapid prototyping and quick iteration of design and development





# Thank You!

[www.qt.io](http://www.qt.io)

**Dr. Roger Hampel**

Director,

Automotive Business Development

[Roger.Hampel@qt.io](mailto:Roger.Hampel@qt.io)

+49 160 9317 1194

Qt is what you make of it!

Check out  
our Blog at  
[blog.qt.io](http://blog.qt.io)

The screenshot shows the Qt Blog homepage with a dark green header. The header includes the 'Qt Blog' title, '24468 Comments', and '1848 Posts'. Below the header, there are two main columns: 'Dev Loop' and 'Biz Circuit'. The 'Dev Loop' column lists articles such as 'Qt 5.8 Beta Released', 'Qt World Summit 2016 Webinar Series Kicks Off', 'Qt Creator 4.2 Beta released', 'Qt 5.6.2 Released', 'Qt Quick Controls 2.1 and Beyond', 'Customizable vector maps with the Mapbox Qt SDK', and 'Qt Champion nominations for 2016 now open'. The 'Biz Circuit' column lists articles such as 'Qt World Summit 2016 Webinar Series Kicks Off', 'Make It Magic – User Experience and the Internet of Things', 'Customizable vector maps with the Mapbox Qt SDK', 'Your data, your code, your cloud... your choice!', 'Internet of Things: Why Tools Matter?', and 'Creating Certified Medical Devices with Qt'. On the right side, there is a yellow Snapchat QR code with the Qt logo, a 'Categories' dropdown menu, an 'Archives' dropdown menu, a 'Subscribe (RSS)' link, and a 'Recent Comments' section.