Das Beste aus zwei Welten

Automotive-HMI mit webbasierten und proprietären Technologien

HTML5 ist mehr als nur eine Beschreibungssprache zum Erstellen von Internetseiten, denn es bietet eine Anwendungsumgebung, die alle Fähigkeiten eines üblichen HMI-Toolkits aufweist – inklusive Renderer, Content-Authoring-Tool, Programmiersprache (Javascript) und Layoutvorgaben (beispielsweise CSS3). HTML5 bringt außerdem Vorteile, die traditionelle HMI-Toolkits nicht bieten, zu denen auch der Zugang zu einem großen Pool an Web-Anwendungsentwicklern und die Integration der neuesten Smartphones gehören.

Autoren: Andy Gryc und Thomas Fleischmann

Unternehmen der Autobranche haben großes Interesse daran gezeigt, HTML5 als Basis für die Entwicklung von grafischen Benutzeroberflächen (HMI, Mensch-Maschine-Schnittstellen) zu verwenden. Auch alle großen Mobilfunkplattformen unterstützen HTML5. Allein dies macht HTML5 für den Automobilbereich schon interessant, und die neusten Entwicklungen machen den Einsatz noch reizvoller. Das World Wide Web Consortium (W3C) beispielsweise hat die Automotive Business Group ins Leben gerufen, die den Auftrag hat, den Einsatz von Webtechnologien einschließlich HTML5, CSS3 und Javascript im Fahrzeug zu beschleunigen. Die Gruppe arbeitet aktuell an der Spezifikation einer Programmierschnittstelle (API) für den Zugriff auf Daten der Fahrzeugsensoren, die ausschließlich auf der Verwendung von Web-Technologien basiert. In der Zwischenzeit entwickelt das Open-Source-Projekt Apache Cordova eine Reihe standardisierter Javascript-APIs für den Zugriff auf native Gerätefunktionen. Die APIs bleiben über Software-Plattformen hinweg konsistent (Android, Blackberry 10, iOS und Windows 8 gehören unter anderem zu den unterstützten Plattformen), wodurch die Entwickler mehrere Plattformen gleichzeitig

Bild 1: HTML5-Apps laufen auf einer mit einem nativen Toolkit erstellten Infotainment-HMI.

bedienen können. Gemeinsam werden diese Initiativen sicherstellen, dass die HTML5-App-Umgebung für Fahrzeuge mit der Umgebung auf Mobiltelefonen harmoniert.

Trotz dieser vielen Vorteile ist HTML5 nicht immer die beste Option für ein im Fahrzeug integriertes HMI. HTML5 hilft dem Automobilhersteller zwar, die vernetzte Außenwelt in das Fahrzeug zu holen, aber andererseits will niemand, dass diese Welt mit ihrem unvorhersehbaren Webcontent und möglichen Sicherheitsbedrohungen das HMI-Verhalten im Fahrzeug negativ beeinflusst. Tatsächlich benötigen die Automobilhersteller HTML5 nicht so sehr als Kern-HMI sondern für flexible Erweiterungs-Apps, die in einer Sandbox angeboten werden. Auch wenn sich die Leistung von HTML5 ständig verbessert, bieten die meisten nativen Toolkits immer noch schnellere Startzeiten und eine schnellere Ausführung bei gleichzeitig weniger Speicherbedarf.

Natives HMI-Toolkit

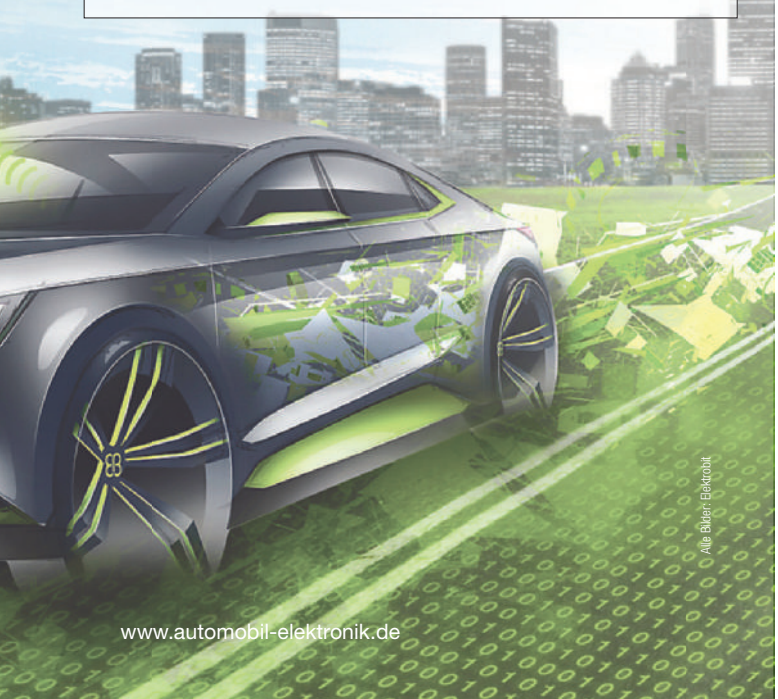
Ein natives HMI-Toolkit kann auch während des Systemdesigns und der Entwicklung Vorteile bieten. Einige Toolkits unterstützen beispielsweise State-Machines, wodurch Entwickler ein komplettes HMI ohne Schreiben eines Codes erstellen können. State-Machines vereinfachen auch die Herstellung eines prüfbaren HMI, das eng mit der HMI-Spezifikation des Automobilherstellers übereinstimmt. Einige native Toolkits bieten auch Funktionen an, die speziell auf die Bedürfnisse der Autoentwickler zugeschnitten sind. EB Guide von Elektrobit beispielsweise, ein gängiges und umfangreiches Toolkit für die Entwicklung, Code-Generierung und Produktion von Infotainment- und Cluster-Systemen, ist das einzige HMI-Tool mit voller Sprachdialog-Integration im HMI.

Auf einen Blick

Zwei Umgebungen vermischen

Ein Konzept, das HTML5- und native Umgebungen miteinander vermischt und das jeweils Beste aus beiden Welten nutzt, ermöglicht das direkte Hinzufügen von heruntergeladenen Apps und Inhalten von mobilen Geräten zum bestehenden Infotainment-Design. Tatsächlich ist es mit der richtigen Plattformarchitektur auch möglich, Android-Apps Seite an Seite mit HTML5-Apps und dem nativen HMI einzubinden, wobei die Apps von Android und HTML5 in der sicheren Umgebung getrennter Anwendungsbereiche laufen.

infoDIREKT www.all-electronics.de 312AEL0613



Der Sprachdialog nutzt die gleiche State-Machine-Infrastruktur, die auch zur Entwicklung der grafischen Benutzeroberfläche verwendet wird. EB Guide und andere Toolkits erlauben den Entwicklern sogar, in Adobe Photoshop erstellte HMI-Komponenten direkt in die Live-Systemgestaltung zu importieren. Auf diese Weise können die Ingenieure des Teams sofort das nutzen, was der HMI-Designer bereitstellt, anstatt Tage oder Wochen mit der Neuerstellung des HMI in Code zu verbringen.

Einige native Toolkits unterstützen auch Frameworks, die simulieren, wie Software-Komponenten in der Produktionsumgebung mit anderen Komponenten interagieren. Außerdem besteht die Möglichkeit, die Interaktion einer HMI mit Fahrzeugsystemen zu simulieren, die sich noch in der Entwicklung befinden. Meistens stehen die HMI-Entwickler außen vor, da sie fertige Hardware, bevor sie ihre Arbeit austesten können. Dank der Simulation sind sie jedoch unabhängig und arbeiten parallel mit dem Integrationsteam. Indessen erlauben die von nativen Toolkits unterstützten Test-Frameworks dem Autobauer, zu überprüfen, ob ein Entwurf entsprechend der Spezifikation umgesetzt wurde.

HTML5 und natives HMI-Toolkit gleichzeitig

Müssen sich Automobilunternehmen also zwischen den Vorteilen von HTML5 und denen nativer HMI-Toolkits entscheiden? Nicht im Geringsten. Beispiel hierfür ist Bild 1, das die CAR-Plattform für Infotainment von QNX zeigt, in der HTML5-Anwendungen auf einer HMI laufen, die mit einem nativen Anwendungs-Toolkit erstellt wurde: Komponenten, die in HTML5 erstellt wurden und Komponenten, die mit dem nativen Toolkit erstellt wurden, erscheinen gleichzeitig auf demselben Display.

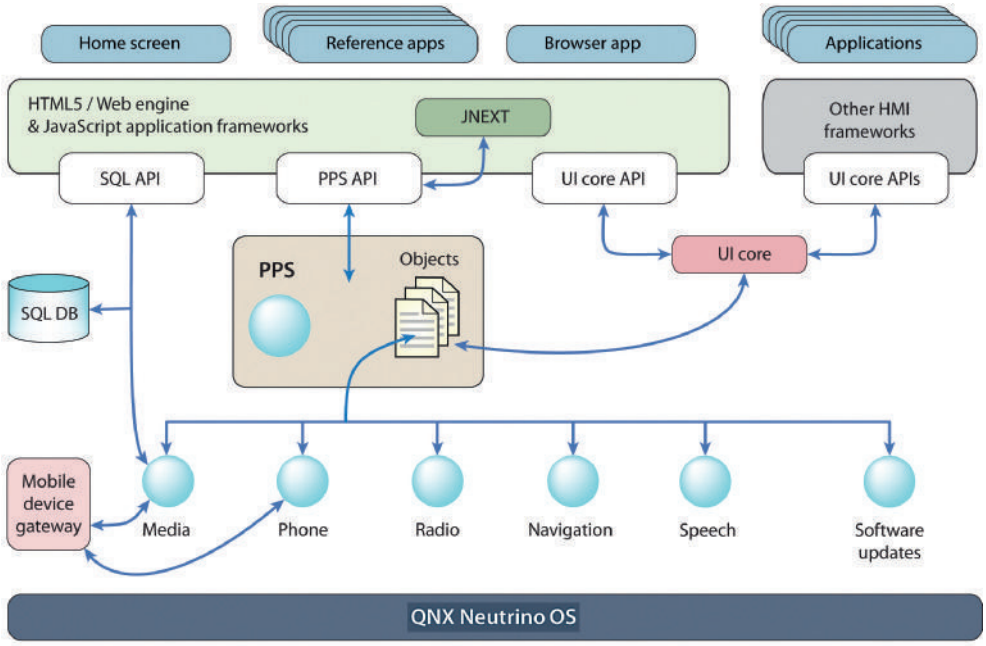


Bild 2: Eine Abstraktionsschicht für Dienste, die persistente Publish-Subscribe-Objekte (PPS) zur Kommunikation zwischen HMI-Komponenten und Diensten auf darunterliegenden Ebenen verwendet.

Technische Voraussetzungen

Um diese nahtlose Einblendung zu erreichen, sind einige technische Voraussetzungen notwendig. Zunächst wird eine grafische Gestaltung benötigt, also die Fähigkeit, die Ausgaben verschiedener Anwendungen in einem einzelnen Fenster zu integrieren. Die Fenster müssen sich als Kacheln, überlappend, integriert oder einer Kombination hieraus darstellen lassen. Für eine gute Leistung muss das System hierbei die Hardwarebeschleunigung im Grafikprozessor ausnutzen.

Als nächstes ist das Management von Ereignissen wichtig. Alle Input-Ereignisse – Berührung (Touch), Druckknopf/Schalter, Drehregler, Timer oder intern generierte – müssen an den entsprechenden Prozess innerhalb jedes Subsystems geleitet werden. Auch eine Interprozesskommunikation (IPC) ist erforderlich. Mehrere zusammenwirkende Prozesse müssen auf eine generische Weise Informationen teilen, Aktionen auslösen und Statusberichte erstellen, damit mehrere Toolkits und in verschiedenen Programmiersprachen geschriebene Komponenten darauf zugreifen können.

Um HTML5 und native Strukturen erfolgreich zu mischen, ist auch eine Abstraktionsschicht nötig, die koordiniert, wie die verschiedenen Technologien mit den Systemdiensten interagieren. Bei einer auf Publish-Subscribe-Messaging basierenden Abstraktionsschicht beispielsweise bieten Datenobjekte Zugriff auf nahezu alle zugrundeliegenden von der Infotainment-Plattform angebotenen Dienste wie Multimedia-Engine, Datenbank-Engine, Fahrzeugbusse, angeschlossene Geräte, Bluetooth-Profile, Stimmerkennung, Freisprechfunktion, Telefonbuch und Kontaktdatenbanken. Jedes Objekt besteht aus mehreren Merkmalen, die Zugriff auf bestimmte Funktionen bieten, wie etwa die Frequenz des aktuellen Radiosenders. Bei korrekter Umsetzung erlaubt die Abstraktionsschicht den in verschiedenen Programmiersprachen geschriebenen Prozessen, neue Attributsänderungen einfach zu veröffentlichen und Informationen über Attributsänderungen zu erhalten.

Abstraktionsschicht

Eine Abstraktionsschicht für Dienste muss mehrere Aufrufer eines Dienstes koordinieren und Synchronisationsprobleme vermeiden. Diese kommen zwar selten vor, können aber dennoch zu Systemversagen führen. Die Schicht sollte auch von der Umsetzung der zugrundeliegenden Dienste getrennt sein. Auf diese Weise hat eine Auslagerung der zugrundeliegenden Engine für zum Beispiel Navigation oder Spracherkennung keine Auswirkungen auf andere Anwendungen, die auf die Engine zurückgreifen.

Zuletzt wird ein Launcher-Prozess benötigt, der die Anwendungen aus der Perspektive des Nutzers visuell miteinander integriert. Dieser Prozess muss Apps starten, sie mit den zugrundeliegenden Diensten koordinieren und alle Anwendungsänderungen abstimmen, also als Dirigent des Systems fungieren. *(av)* ■

Infokasten

Vorteile von HTML5

Native Toolkits können für die meisten HMIs im Fahrzeug die beste Wahl sein, aber in einigen Fällen ist HTML5 sinnvoll. HTML5 ist in folgenden Fällen für ein System empfehlenswert:

- Reskinning, Personalisierung und Kundenanpassung: HTML5 unterstützt Cascading Style Sheets (CSS), die die Anwendungslogik klar von der Benutzerschnittstelle trennen. HTML5 macht es somit einfacher, eine HMI an Kunden anzupassen oder zu „re-skinnen". Reskinning ist nur bei wenigen anderen Toolkits integraler Bestandteil.
- Ausführbare HMI-Spezifikation: Manchmal möchte der Automobilhersteller dem Tier-1 eine direkte Spezifikation des HMI zur Verfügung stellen, anstatt das HMI anhand von Bildschirmausdrucken nachbilden zu lassen. In diesem Fall stellt der Automobilhersteller den in HTML5 codierten HMI-Prototyp bereit, und der Tier-1 kann sich um die Details bei der Umsetzung kümmern und die eingebettete HMI mit weniger Rückfragen effizienter erstellen.
- Wiederverwendung des Codes über Produktlinien hinweg: HTML5 kann auf der Head Unit im Fahrzeug und auf Mobiltelefonen laufen. Daher können Entwickler eine einzige HMI-Code-Basis erstellen, die unabhängig davon arbeitet, ob das Auto eine Head Unit besitzt (bei der das HMI im Auto läuft) oder ein „kopfloses", telefonbasiertes System (bei dem das HMI auf dem Telefon läuft).

Die Autoren:
Andy Gryc ist Senior Automotive Product Marketing Manager bei QNX Software Systems.

Thomas Fleischmann ist Senior Product Manager HMI bei Elektrobit Automotive.