Dr. Jan Becker
CEO von Apex.AI
in Palo Alto (USA)

© Apex.AI

Standards für Automobil-Betriebssysteme

Die deutsche Automobilindustrie diskutiert zwar ein gemeinsames Betriebssystem. Es scheint jedoch jeder seine eigene Lösung durchsetzen zu wollen. Damit steuern wir auf eine Situation zu, die wir schon einmal erlebt haben: beim Smartphone. Vor zwölf Jahren war der Smartphone-Betriebssystemmarkt fragmentiert und bestand aus Apple iOS und einem halben Dutzend weiterer Systeme. Heute sind nur Android mit 72 % Marktanteil und iOS mit 27 % übriggeblieben. Der Grund dafür ist simpel: Betriebssystemmärkte sind Winner-takes-all-Märkte.

Natürlich will nun jeder große Player in der automobilen Welt so wie Apple sein und „ein (i)OS besitzen“. Diese Gelegenheit gab es tatsächlich vor acht Jahren. Genutzt hat sie Tesla. Für alle anderen ist die Zeit dagegen abgelaufen. Außerdem wird verkannt, dass sich eine Software-Entwicklungskultur nicht über Nacht etablieren lässt und die Transformation ein enormes Budget erfordert. Ein Budget, das zwar Apple und auch Tesla, nicht aber unbedingt den klassischen Herstellern zur Verfügung steht. Dabei stellen Kultur und Geld nur zwei Hürden dar, die dritte ist das Fehlen an herausragenden Softwarearchitekten. Sie sind eine rare Spezies, die nur selten ihren Weg in die Auto-industrie findet. Und wenn die Auto-Unternehmen dennoch Softwareentwickler bekommen, dann müssen diese zunächst aufwendig auf das eigene proprietäre System trainiert werden.

Bleibt also das Android-Modell. Hier sind die Schnittstellen offen, und es gibt frei verfügbaren Quellcode. Diese gemeinsame, offene Basis kann jedoch nicht von einem der großen Player kommen, denn dann würde dieser Basislayer keine breite Akzeptanz finden. Im Idealfall hätte diese Basis ohne-

hin bereits eine weite Verbreitung, um ein Zuspätkommen zu verhindern. Und wenn dieses System dann auch noch international an den Universitäten Verwendung fände, dann wären die zukünftigen Entwickler bereits mit ihm vertraut, bevor sie in ihren ersten Job einsteigen. Zu schön, um wahr zu sein?

Vielleicht nicht. Es hilft ein Blick in die Robotik, die genau diese Herausforderung mit dem quell-offenen Robot Operating System (ROS) gelöst hat. Seit vier Jahren gibt es bereits die robustere zweite Version (ROS 2), die durch eine seit einigen Monaten verfügbare sicherheitszertifizierte Variante (ISO 26262 ASIL D) ergänzt wird.

Eine weitere Herausforderung für die OEMs und Tier-1-Zulieferer ist die Software-Entwicklungsproduktivität: Automobilprojekte beschäftigen oft Hunderte von Softwareentwicklern. Der enorme Anpassungsaufwand lässt sich aber vermeiden, wenn man eine wirklich saubere Softwarearchitektur schafft, anstatt umständlich Legacy-Komponenten mit neuer Software zu kombinieren. Um erneut die Smartphone-Analogie zu bemühen: Niemand würde versuchen, auf seinem aktuellen iPhone 13 Software zu installieren und zu kombinieren, die für ein Nokia-Telefon von 2002 und einen Blackberry von 2006 entwickelt wurde.

Vielleicht sollte die Industrie den Android-Weg gehen, einen gemeinsamen Basislayer verwenden und sich auf das Wesentliche, nämlich auf die Richtung Kunde und Wettbewerb differenzierenden Merkmale, konzentrieren. Hier kommen so viele Features auf uns zu, dass es mehr als genug zu tun gibt für die Softwareentwickler der OEMs und Tier-1.