소프트웨어-정의 자동차와 미래 자동차 산업

Software-Defined Vehicle and Future of Automotive Industry

홍성수
서울대학교
Seongsoo Hong
Seoul National University

소프트웨어-정의 자동차가 미래 자동차 산업의 핵심 기술로 급격히 부상하고 있다. 그 이름이 지니는 모호성과는 달리, 소프트웨어-정의 자동차는 명확한 산업적, 기술적 필요성을 가지고 있다. 또한 타산업에서 이미 완성된 기술과 검증된 성공을 전제로 하고 있기에 미래 자동차 산업에서 핵심적 요소가 될 것이다. 따라서 소프트웨어-정의 자동차에 대한 명확한 이해를 바탕으로 그 한계와 가능성을 평가해 볼 필요가 있다. 본 고에서는 소프트웨어-정의 자동차의 등장배경과 필요성을 살펴보고, 이를 명확하게 정의하며, 우리가 해결해야 할 소프트웨어 공학적 이슈들을 살펴본다.

소프트웨어-정의 자동차의 등장 배경

벤츠[1], 폭스바겐[2], 현대기아자동차[3] 등의 글로벌 기업들이 대외적으로 천명하듯이, 소프트웨어-정의 자동차(Software-Defined Vehicle, SDV)는 최근 자동차 산업에서 가장 핵심적인 키워드 중의 하나로 부상하고 있다.[4] 그러나 한편으로 SDV는 많은 사람들에게 큰 모호성을 던지는 용어로 평가되기도 한다.[5]

급격히 변화하는 미래 자동차 산업을 고려했을 때, SDV는 지극히 논리적인, 최선의 기술적 선택이라고 할 수 있다. 그 이유는 SDV가 미래 자동차 산업에 반드시 이식되어야 하는 실체적 당위성과 필요성을 가지고 있기 때문이다. 이와 같은 주장을 좀더 잘 이해하기 위해 SDV의 등장 배경을 (1) 제조업 패러다임, (2) 제조 효율화, (3) 소프트웨어 공학이라는 세가지 측면에서 분석해 보자.[6]

첫째, 제조업 패러다임의 관점에서 볼 때, 지난 20년간 큰 변화가 발생하였는데, 그것이 바로 "서비스화"이다.[7] "제조업의 서비스화"란 제품에 스마트 칩을 내장하여 새로운 서비스를 창출하는 "제품 서비스화(Product Servitization)"와 새로운 서비스를 제품화하는 "서비스 제품화(Service Productization)"를 의미한다. 자동차 산업도 예외가 아니어서 서비스화의 물결을 피해갈 수 없다. 오히려 글로벌하게 이를 선도해야 한다. 그러기 위해서는, 반드시 하드웨어와 소프트웨어를 융합하여, 제품과 서비스, 그리고 고객을 결합시키는 능동적인 제품 기획과 엔지니어링이 필요하다. 이를 지원하는 제품 형태가 바로 SDV인 것이다.

둘째, 전기차와 같은 미래차의 등장으로 자동차 산업에서 제품 원가 경쟁은 더욱 첨예해지고 있다. 주요 완성차 기업들은 가격경쟁력 제고를 위해 PC와 스마트폰과 같이 표준화된 컴퓨팅 하드웨어 플랫폼을 채용하고 있다. 이는 ECU 단위의 결합을 전제로한 기존의 공급망에 큰 변화를 초래하게 되며, 소프트웨어가 하드웨어로부터 분리되는 디커플링(Decoupling)을 발생시킨다. 이것이 SDV로 진화하는 차량 E/E 아키텍처의 기반이 된다.

셋째, 차량에 탑재되는 방대한 양의 소프트웨어를 효과적이고 효율적으로 다룰 수 있는 소프트웨어 공학적 체계가 필요한데, 이것이 SDV의 핵심 개념인 Service-Oriented Architecture(SOA)이다.[8] 단일 소프트웨어의 개발과 배포에 치중하는 기존의 개발 방식과는 다르게 SOA에서는 소프트웨어의 점진적 개발을 선호한다. 또한 공유가능한 Micro-Service를 결합하여 소프트웨어를 제작하고, 제품의 배포 후에 개선과 업데이트를 수행하는 점진적 개선(Evolutionary Refinement)을 추구한다.

전통적인 자동차 기술의 관점에 몰입되면, SDV는 모호한 언어적 관념으로만 보여질 수 있다. 그러나 제품의 서비스화를 이미 성공적으로 경험한 타산업의 관점에서 보면, 위의 세가지 현상은 새로운 것이 아니며, 오히려 성공의 요인으로 작용한다는 것을 알 수 있다. 그 대표적인 예가 스마트폰 산업이다. 2007년에 처음 등장한 아이폰은 산업에는 물론, 사회와 문화에까지 지대한 영향을 미쳤다. 이제 우리가 주목할 점은, 스마트폰이 보여준 성공을 SDV에서도 거두기 위해 필요한 기술과 전략을 모색하는 것이다. 특히 자동차 산업의 특성을 고려하여, 타산업에서 수정 없이 채용할 수 있는 기술과 차량에 특화시켜 적용해야 할 기술을 통찰력 있게 구분하고, 취사선택하여야 한다.

SDV란 무엇인가?

SDV는 그 기능이 소프트웨어에 의해서 정의되는 또는 결정되는 차량을 의미한다. 이때 SDV의 "기능"이 가지는 양면적 특징을 이해하는 것이 매우 중요하다. SDV의 기능은 소비자의 관점인 "Feature"와 개발자의 관점인 "Function"을 모두 포함한다. 이것은 제품을 기획하고 제조하는 과정에서 소비자의 관점과 개발자의 구현수단을 일체화시킨다는 것을 의미한다. 이때 그 일체화의 수단이 바로 "서비스"이다.

SDV에서 "서비스"는 고객이 자신의 욕구를 충족시키기 위해 기꺼이 대가를 지불하도록 하는 이윤 창출의 수단이면서,

동시에 개발자가 제품을 기획하고 설계할 때 사용하는 구현 수단이기도 하다. SDV에서 서비스의 특성은 〈그림 1〉과 같이 규정지을 수 있다.[4] 즉, 서비스는 제품 기획의 단위이고, 개발의 단위이며, 공유와 배포의 단위가 된다. SDV와 같이 서비스를 중심으로 소프트웨어 시스템을 개발하고 운영하는 구조를 앞에서 언급한 SOA[8]라고 한다.

SOA는 원래 엔터프라이즈 컴퓨팅(Enterprise Computing)에서 파생된 개념으로 방대한 소프트웨어를 기획하고, 개발하고, 배포하고, 운영하는 총체적인 워크프로세스와 소프트웨어 개발 방법론을 포함하고 있다. SDV에서 SOA를 채용했다는 것은 이런 워크프로세스와 개발 방법론을 사용할 수 있음을 의미한다.

그림 1. SDV의 서비스의 특성

그림 2. SDV의 핵심 요소들

SDV 실현을 위한 핵심요소들

SDV를 제품으로 기획하여 구현하고, 고객에게 판매하여 사용되도록 하기 위해서 SDV 생태계는 〈그림 2〉와 같은 네 가지 구성요소를 갖추고 있어야 한다.[5] 첫째가 "서비스 모델"이다. 서비스는 SDV의 소프트웨어 시스템을 구성하는 빌딩 블록인 동시에 고객에게 제공되는 기능을 표현한다. 따라서 SDV의 서비스 모델은 〈그림 1〉의 특성을 잘 만족하여야 한다.

둘째는 서비스의 통합을 지원하는 미들웨어로서 "소프트웨어 플랫폼"이다. 이 플랫폼은 다양한 서비스들을 정적으로, 동적으로 통합하는 것을 가장 핵심적인 기능으로 지원한다. 동시에, 기능안전을 보장해야 하는 차량 제어 플랫폼이 요구하는 실시간성, 신뢰성, 보안성 등 다양한 비기능적 요구사항을 만족시켜야 한다.

셋째는 "E/E 아키텍처"이다. 이는 SDV의 다양한 소프트웨어를 수행시키는 하드웨어 플랫폼을 의미한다. 이 E/E 아키텍처는 기존의 도메인 기반 차량 아키텍처와는 달리 중앙집중화와 표준화를 지원한다. 이를 통해 말단 ECU에서 수행되던 제어 펌웨어 조차도 차량 컴퓨터(Vehicle Computer)로 불리우는 고성능 AP(Application Processor) 위에서 중앙집중적으로 수행된다. 즉, 거의 모든 종류의 소프트웨어가 차량 컴퓨터의 표준화된 하드웨어 자원을 공유하면서 수행된다. 이와 같은 런타임 플랫폼에 대한 탈의존성이 서비스의 동적 배포와 수행을 지원하는 토대인 것이다.

넷째가 "소프트웨어 공급망"이다. SDV 생태계 안에서는 제3의 벤더들이 소프트웨어 컴포넌트로 구현된 서비스를 공급하게 된다. 결국 제어 소프트웨어가 ECU라는 하드웨어에 내장되어 공급되던 기존의 하드웨어 중심적인 공급망과 차별화되는 새로운 공급망 체계가 되는 것이다.

미래 자동차 산업에서 승패는 앞에서 언급한 네가지 요소들을 경쟁력 있는 실체로 구현할 수 있는가에 달려있다. 특히 이 네가지 요소들은 첨단 소프트웨어 기술에 의존하기 때문에 이를 구현할 수 있는 개발인력을 육성하고, SOA의 워크프로세스와 방법론을 내재화하는 것이 중요하다.

SDV에 대한 미래 분석과 기술 극복 과제

앞에서 SDV가 등장하게 된 당위성을 살펴보았고, 자연스럽게 SDV가 미래 자동차 시장에 끼치는 긍정적인 효과를 짐작할 수 있었다. 그러나 SDV가 과연 타산업에서와 같이 완벽한 수준의 서비스화를 달성할 수 있는가에 대해서 비판적으로 분석할 필요가 있다.[6]

결론적으로 요약한다면, 현재의 소프트웨어 기술만을 전제로 한다면 SDV의 유용성은 타산업에 비해 제한적일 수밖에 없다. 그 구체적인 예를 들어보자. SDV 기반 미래 자동차 산업에서 스마트폰과 같이 개방화된 생태계를 구현하는 것은 거의 불가능하다. 그 결정적인 이유는 자동차와 스마트폰이라는 제품의 특성 차이에 있다.

스마트폰에서 구동되는 소프트웨어는 "플랫폼 독립성(Platform Agnosticism)의 원칙"에 기반하고 있다. 즉, 각 소프트웨어가 상호간에 약속된 기능적 요구사항(Functional Requirement)만 만족시키면, 쉽게 결합되어 수행이 가능하다는 것이다. 그에 비해 SDV에서 구동되는 소프트웨어는 강력한 기능안전 요구를 충족시켜야 한다. 이를 위해 실시간성, 결함허용성, 신뢰성, 보안성과 같은 비기능적 요구사항(Non-functional Requirement)을 모두 만족시켜야 한다. 이런 비기능적 요구사항들은 타겟 플랫폼의 연산 성능, 통신 지연, 미들웨어의 구현 형태 등에 민감하기 때문에, 필연적으로 플랫폼 독립성의 원칙을 훼손하게 된다. 따라서 스마트폰 산업에서 구축된 많은 소프트웨어 기술들을 SDV에서 그대로 채용할 수 없게 된다.[6]

굳이 기존의 소프트웨어 기술만을 적용하여 SDV 생태계를 구축하고자 한다면, 완성차 기업들은 자신의 비기능적 요구사항을 공유할 수 있는 소수의 파트너들과 함께 폐쇄적인 생태계를 구축하여야 한다. 개별 완성차 기업들은 이런 폐쇄적인 생태계 안에서 신뢰할 수 있는, 선택된 소프트웨어 벤더들만을 통해 서비스를 공급받게 된다.

이런 문제를 극복하기 위해서는 SDV에 특화된 소프트웨어 기술들을 연구·개발하여야 한다. 구체적으로 (1) 기능적 요구사항과 함께 비기능적 요구사항을 표준 인터페이스화하는 기술, (2) 플랫폼 독립적으로 개발된 서비스의 채용을 통제하는 Admission Control 기술, (3) 임의의 소프트웨어 개발자가 플랫폼에 독립적으로 소프트웨어를 개발할 수 있도록 하는 FaaS(Function as a Service) 기술 등이 있다.[6]

차량의 Feature와 Function을 소프트웨어를 통해 정의하고, 차량을 서비스 중심적으로 개발하고자 하는 SDV는 제조업의 서비스화, 제조 공정의 효율성, 방대한 소프트웨어에 대한 공학적 관리라는 측면에서 미래 자동차의 핵심 기술로 부상하였다. 이미 다수의 글로벌 기업들이 속속 SDV 기반의 차량을 양산하는 계획을 천명하고 있다.

SDV라는 미래 지향적 기술의 부상에 신속하게 대응하고 있는 한국의 자동차 산업은 이 분야에서 빠르게 선도 기업들을 추격하고 있다. 그러나 SDV는 타산업과는 다르게 아직 미완성의 기술이며, 소프트웨어 공학적으로 많은 기술적 난관을 극복해야 할 도전 과제이다. 따라서 자동차 산업이 급격히 소프트웨어화 하는 흐름에 맞추어, SDV 소프트웨어 개발인력을 육성하고, 앞절에서 언급한 기술적 난제를 풀기위한 연구·개발에 적극적인 투자를 아끼지 말아야 할 것이다.

홍성수 한국자동차공학회 자동차반도체 및 시스템S/W부문회장
: sshong@redwood.snu.ac.kr

〈참고문헌〉

1. https://group-media.mercedes-benz.com/marsMediaSite/ko/en/46665504
2. https://www.volkswagenag.com/en/news/stories/2020/06/we-want-to-develop-our-own-software-platform.html#
3. https://youtu.be/c31cf2lO0-E
4. https://youtu.be/YhXyM8Squ2I
5. https://youtu.be/C_L-flYPMFs
6. https://youtu.be/eLFP8mH000c
7. 유연성, 임호순, 서비스화(servitization) 개념적 모형(conceptual framework)과 서비스화 사례연구, Journal of service research and studies, Vol.1 No.1, pp.1-27, 2011
8. https://en.wikipedia.org/wiki/Service-oriented_architecture