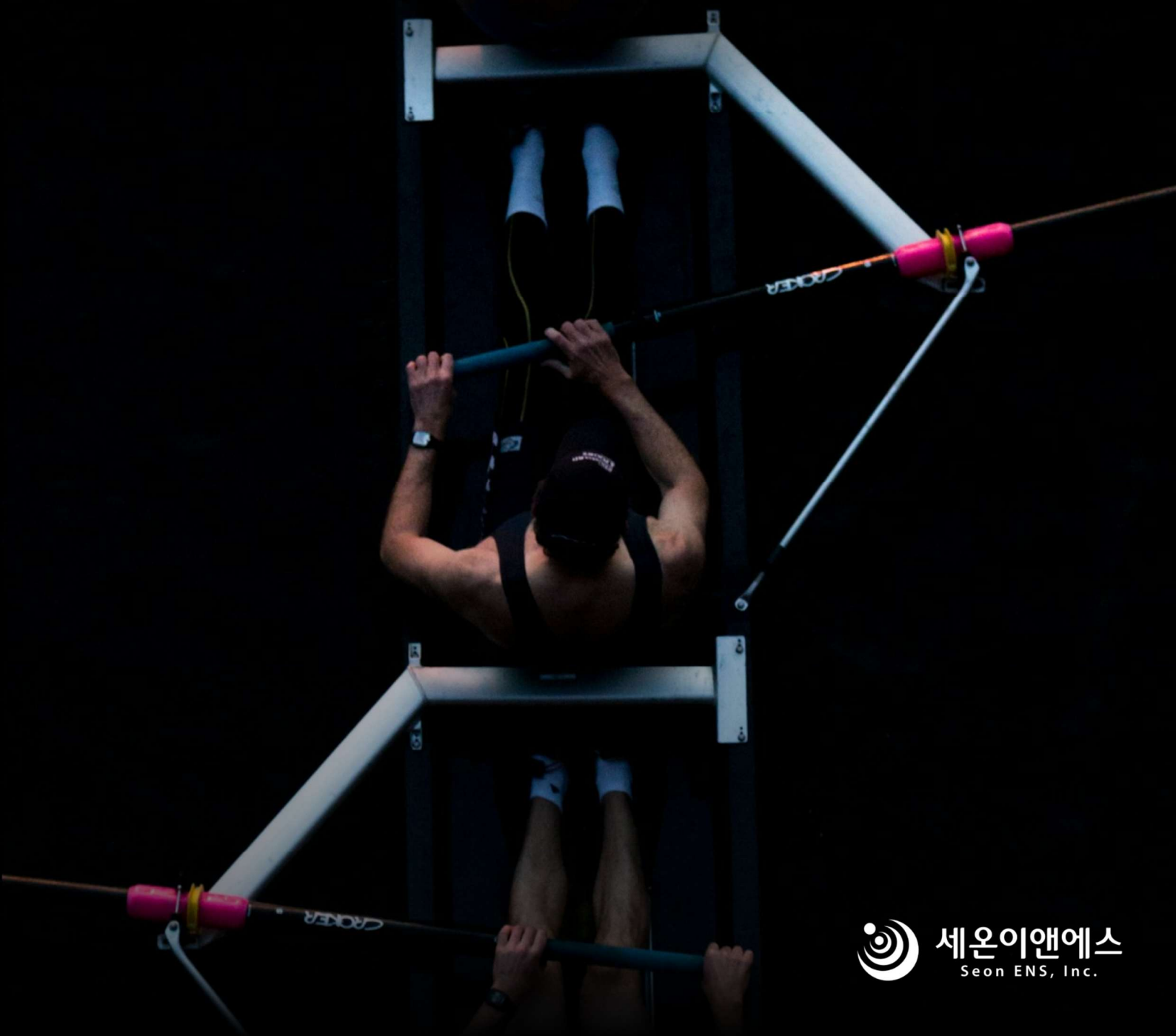
White Paper

The Practical Initiatives for AUTOMOTIVE AGILE

자동차 산업의 SDV 전환을 위한 실용적 애자일 가이드



세온이앤에스
Seon ENS, Inc.

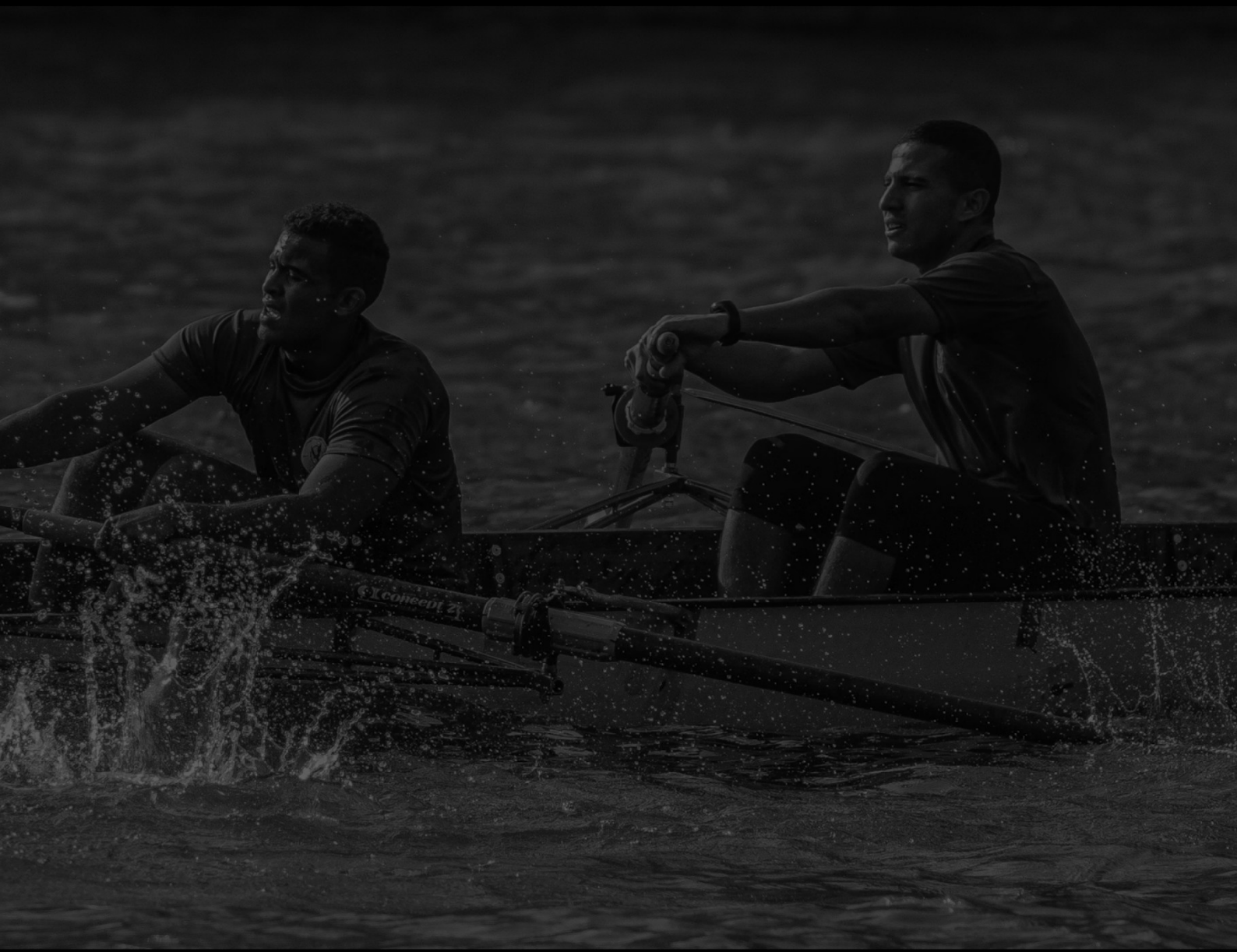
Why Now, AUTOMOTIVE AGILE?

변화의 물결, 선택의 기로

고객은 진화하는 제품을 원하고,
기술은 예측을 거부하며,
경쟁자는 규칙을 다시 쓰고 있습니다.

100년 제조 우수성의 유산과
소프트웨어 민첩성의 요구 사이에서,

자동차 산업은
새로운 균형점을 찾아야 합니다.



The Practical Initiatives for

AUTOMOTIVE AGILE

자동차 산업의 SDV 전환을 위한 실용적 애자일 가이드

세온이앤에스
Seon ENS, Inc.

목차

서문

Wicked Problem이란 무엇인가?

Wicked Problem(사악한 문제)은 1973년 도시계획학자 Horst Rittel과 Melvin Webber가 처음 정의한 개념으로, 명확하게 정의하기 어렵고 완벽한 해결책이 존재하지 않는 복잡한 문제를 의미합니다. 이들은 도시 계획의 복잡성을 설명하면서 이 용어를 만들었지만, 오늘날에는 소프트웨어 개발부터 기후 변화까지 다양한 분야에서 활용되고 있습니다.

이러한 문제들이 가진 핵심적인 특징들을 살펴보면, 먼저 문제 자체를 정의하기가 모호하며 문제를 완전히 이해하려면 해결책을 시도해봐야 한다는 순환적 특성이 있습니다. 또한 정답이 하나로 정해져 있지 않고 '맞다/틀리다'가 아닌 '더 낫다/덜 낫다'로만 평가할 수 있다는 점에서 전통적인 문제 해결 방식과 구별됩니다. 더욱이 해결책을 시도해보기 전까지는 그 효과를 알 수 없으며, 각 시도는 되돌릴 수 없는 결과를 낳기 때문에 실험의 비용이 매우 높습니다.

각 문제가 독특하고 반복 불가능하여 과거의 해결책을 그대로 적용할 수 없다는 점도 중요한 특징입니다. 마지막으로 이해관계자마다 문제를 다르게 인식하고 서로 상충되는 요구사항을 제시한다는 점에서 합의 도출 자체가 하나의 도전이 됩니다.

라이트 형제의 교훈: 빠른 반복의 힘

라이트 형제가 1903년 12월 17일 최초의 동력 비행에 성공한 비결은 경쟁자들보다 훨씬 빠른 반복 주기에 있었습니다. 이들의 성공 스토리는 단순한 행운이나 천재성의 결과가 아니라, 체계적인 실험과 학습의 산물이었습니다. 당시 스미소니언 협회의 막대한 지원을 받던 Samuel Langley는 5만 달러라는, 오늘날 가치로 환산하면 150만 달러가 넘는 거대한 예산으로 'Great Aerodrome'을 개발했습니다. 그는 완벽한 이론적 계산과 정밀한 설계를 통해 단번에 성공하려 했지만, 1903년 10월 7일과 12월 8일 두 차례 모두 포토맥 강에 추락하는 실패를 겪었습니다.

반면 라이트 형제는 오하이오주 데이턴에서 운영하던 자전거 가게의 수익으로 충당한 1,000달러 미만의 예산으로 전혀 다른 접근을 택했습니다. 그들은 1900년부터 1902년까지 노스캐롤라이나의 키티호크에서 매년 수개월씩 머물며 1,000회 이상의 글라이더 비행을 수행했고, 직접 제작한 풍동에서 200개 이상의 날개 형태를 체계적으로 테스트했습니다. 각 실험은 작고 빠르게 실패했지만, 매번 중요한 데이터와 통찰을 제공했습니다. 양력과 항력의 관계, 날개의 비틀림 제어, 러더의 역할 등 비행의 핵심 원리들을 하나씩 실증적으로 발견하고 개선해 나갔습니다. 이들의 성공은 비행이라는 거대한 wicked problem을 작은 실험들로 나누어 빠르게 가설을 검증하고 학습한 결과였습니다.

애자일: Wicked Problem 해결의 체계화

애자일 방법론은 이러한 "빠른 반복을 통한 학습"이라는 원칙을 소프트웨어 개발에

체계적으로 적용한 것입니다. 2001년 2월, 유타주 스노버드 스키장에 모인 17명의 소프트웨어 개발자들이 작성한 애자일 선언문은 단순한 방법론의 제안이 아니라 패러다임의 전환을 알리는 신호탄이었습니다. 그들은 수십 년간 소프트웨어 산업을 지배했던 폭포수 모델의 근본적 한계를 지적하며 각자 시도해왔던 새로운 접근법들을 확신하게 되었고 이러한 공식적인 선언을 통해 그 필요성을 역설했습니다.

폭포수 모델은 건축이나 제조업에서 차용한 방법론으로, 요구사항 분석에서 시작해 설계, 구현, 테스트, 유지보수로 이어지는 순차적 단계를 따릅니다. 이러한 접근법은 요구사항이 명확하고 변경이 적은 상황, 예를 들어 교량 건설이나 대형 건물의 설계처럼 물리적 법칙과 규정이 명확한 영역에서는 여전히 효과적입니다. 그러나 대부분의 소프트웨어 개발은 근본적으로 다른 특성을 가지고 있습니다.

첫째, 사용자조차 자신이 진정 원하는 것이 무엇인지 실제 제품을 사용해보기 전까지는 명확히 알지 못합니다. 둘째, 기술 환경이 프로젝트 진행 중에도 급격히 변화하여 초기 설계가 쓸모 없어지는 경우가 빈번합니다. 셋째, 비즈니스 요구사항이 시장 상황과 경쟁 환경에 따라 지속적으로 진화합니다. 이런 특성들 때문에 소프트웨어 개발 자체가 전형적인 wicked problem의 성격을 띠게 되는 것입니다.

스타트업과 애자일의 만남

실리콘밸리의 성공한 스타트업들이 애자일을 적극 도입한 이유는 명확합니다. 그들이 해결하려는 문제 자체가 전형적인 wicked problem이었기 때문입니다. 이들은 기존 시장에 존재하지 않던 새로운 가치를 창출하려 했고, 그 과정에서 끊임없는 실험과

학습이 필요했습니다.

Spotify는 2006년 스웨덴의 작은 스타트업으로 시작했을 때, 음악 스트리밍이라는 개념 자체가 생소한 시대였습니다. 당시 대부분의 사람들은 여전히 CD를 구매하거나 불법 다운로드를 통해 음악을 소비하고 있었습니다. Spotify의 초기 버전은 단순한 음악 재생 기능만을 제공했지만, 사용자들의 행동 패턴을 면밀히 관찰하고 분석하면서 플레이리스트의 중요성을 발견했습니다. 이후 머신러닝 알고리즘을 활용한 Discover Weekly 같은 개인화 추천 서비스를 개발했고, 이는 음악 소비 방식을 근본적으로 변화시켰습니다. 각 기능은 수천 명의 사용자를 대상으로 한 A/B 테스트를 통해 검증되었고, 데이터가 긍정적인 결과를 보여주는 기능만이 전체 사용자에게 배포되었습니다.

Netflix의 변신은 더욱 극적입니다. 1997년 DVD 우편 대여 서비스로 시작한 이 회사는 블록버스터와 같은 거대 경쟁자들과 맞서야 했습니다. 하지만 지속적인 실험과 피벗을 통해 2007년에는 스트리밍 서비스를 도입했고, 2013년 House of Cards를 시작으로 오리지널 콘텐츠 제작에 진출했습니다. 각 전환점마다 작은 실험으로 시작했다는 점이 중요합니다. 스트리밍 서비스는 처음에 기존 DVD 대여 고객들에게 무료 부가 서비스로 제공되었고, 사용자들의 반응을 보며 점진적으로 확대되었습니다.

자동차 산업에서 애자일이 주목받는 이유

자동차 산업은 100년 이상 하드웨어 중심의 개발 방식을 고수해왔습니다. 전통적인 V-사이클 모델은 시스템 요구사항 정의부터 시작해 컴포넌트 설계, 구현, 통합 테스트를 거쳐 최종 검증에 이르는 체계적이고 순차적인 프로세스였습니다. 이러한 접근법은 엔진

설계나 섀시 개발처럼 물리 법칙에 기반한 명확한 사양이 존재하고, 변경 비용이 막대한 영역에서는 여전히 유효하며 필수적입니다.

그러나 SDV(Software-Defined Vehicle) 시대가 도래하면서 자동차 산업의 근본적인 패러다임이 변화하고 있습니다. 테슬라가 2012년 Model S를 출시하며 OTA(Over-The-Air) 업데이트를 도입한 것은 단순한 기능 추가가 아니라 자동차의 정의 자체를 바꾸는 혁명적인 변화였습니다. 차량이 판매된 후에도 지속적으로 기능을 추가하고 개선할 수 있다는 것은, 자동차를 고정된 하드웨어 제품에서 진화하는 소프트웨어 플랫폼으로 재정의한 것입니다.

새로운 유형의 문제의 등장

E2E(End-to-End) 자율주행은 자동차 산업이 직면한 가장 복잡한 wicked problem입니다. "완전 자율주행이 어떤 모습이어야 하는가?"라는 질문에는 명확한 답이 없습니다. 각 나라마다 교통 문화가 다르고, 도로 인프라의 수준이 천차만별이며, 날씨와 지형 조건도 극도로 다양합니다. Waymo는 구글의 자율주행 프로젝트로 2009년부터 개발을 시작했지만, 15년이 지난 지금도 여전히 피닉스, 샌프란시스코 등 제한된 지역에서만 상용 서비스를 제공하고 있습니다. 이들은 수백만 마일의 실제 도로 주행과 수십억 마일의 시뮬레이션을 통해 지속적으로 시스템을 개선하고 있습니다.

인포테인먼트 서비스는 스마트폰 생태계에 익숙한 현대 소비자들의 높아진 기대치를 충족시켜야 하는 과제를 안고 있습니다. Mercedes-Benz의 MBUX나 BMW의 iDrive 같은 시스템들은 이제 단순한 내비게이션과 오디오 재생을 넘어, AI 음성 비서, 증강현실

내비게이션, 개인화된 추천 서비스 등을 제공합니다. 이러한 기능들은 정기적인 소프트웨어 업데이트를 통해 지속적으로 개선되고 있으며, 사용자들의 피드백과 사용 패턴 데이터를 기반으로 진화하고 있습니다.

커넥티드 서비스는 차량을 독립된 제품이 아닌 거대한 모빌리티 생태계의 일부로 만들고 있습니다. 실시간 교통 정보, 주차장 예약, 전기차 충전소 안내, 차량 공유 서비스 등은 모두 외부 시스템과의 지속적인 연결과 데이터 교환을 필요로 합니다. 이러한 서비스들의 요구사항은 시장 상황, 규제 변화, 새로운 파트너십 등에 따라 수시로 변경되며, 유연한 대응이 필수적입니다.

Automotive Agile의 실용적 도입

테슬라와 같은 신흥 OEM, 그리고 폭스바겐, 포르쉐, BMW 등의 전통적인 OEM 모두 SDV 시대의 전환기에 대응하기 위해서 각자의 방식으로 애자일을 도입하고 적용하고 있습니다. 테슬라는 가시적인 성과로서 그 가능성을 보여주기도 했고, 전통 OEM에서도 성과가 있다는 이야기가 있습니다. 다만 성공이든 실패 사례 든지 각각의 것에서, SDV 시대를 맞이하여 변화를 헤쳐 나아가야 하는 OEM, Tier사 모두에게 배워야 할 것도 있고 반면교사로 삼아서 동일한 실패를 겪지 말아야 할 것도 있습니다. 이 White Paper는 이런 관점에서 자동차 도메인에서 애자일을 도입하기 위해서 어떤 전략적인 선택을 하는 게 좋은지 인사이트를 제공해 드리고자 작성됐습니다.

White Paper 구성

White Paper는 크게 세 개의 부분으로 구성되어 있습니다.

첫 번째 파트는 애자일의 철학과 어떤 가치를 추구하는 것인지, 그리고 애자일 선언이 있은 후 30년이 흐르면서 다양한 Variation이 현장에 소개되어 성공을 하기도 하고 목표로 한 성과를 달성하지 못한 것도 있습니다. 이런 개선된 혹은 수정된 방법론을 성공적으로 적용하기 위해서 무엇을 유념해야 할지 정리해 봤습니다.

두 번째 파트는 OEM에서 애자일을 도입하는 사례를 정리해 봤습니다. 애자일 방법의 적용 사례가 그렇듯이, 각각의 적용 사례는 잘된 것도 있고 기대에 미치지 못한 것도 있습니다. 이런 사례 연구를 중심으로 현장에서 애자일을 도입할 때 어떤 점에 주목해야 할지 정리해 봤습니다.

끝으로 애자일 적용을 위해서 고민해 보실 필요가 있는 이니셔티브(Initiative)에 대해서 정리해 봤습니다. 애자일은 여러 방법론이 있지만, 해당 방법론이 모든 현장에 맞는 것은 아닙니다. 애자일이 근본적으로 추구하는 철학과 가치를 이해하지 못한 채 방법론에 익숙해지면, 오히려 애자일을 적용하지 않은 것보다 못한 결과를 얻을 때가 많습니다.

따라서 어떤 방법론을 선택하기 전에 애자일의 근본적인 철학과 지향점을 이해하시는 것을 추천 드립니다. 그 이후에 현장 상황에 맞는 프레임워크를 찾아 적용하셔도 늦지 않습니다. 다만 이런 과정에서 애자일이 다양한 산업 현장과 OEM들의 적용 사례를 봤을 때, 참고로 할만한 이니셔티브는 분명히 있습니다. 따라서 자동차 도메인에서 Automotive Agile을 적용하기 위해서 어떤 관점으로 접근하면 성공적일지 주요한

이니셔티브를 소개해 드립니다.

SDV의 격변기를 맞이하고 있는 대한민국의 자동차 설계/개발 현장에서 더 나은 제품을 만드시는 데, 본 White Paper에서 실용적인 도움을 얻으실 수 있기를 바랍니다.

세온이엔애스 드림.

Part 1.

애자일 정의와
다양한 프레임워크

애자일의 본질적 가치와 지향점

애자일은 단순히 프로젝트 관리 방법론이나 개발 프로세스가 아닙니다. 애자일은 근본적으로 불확실성과 복잡성에 대응하는 새로운 사고방식이자 철학입니다. 2001년 17명의 소프트웨어 개발자들이 모여 작성한 애자일 선언문은 네 가지 핵심 가치를 제시했습니다. 프로세스와 도구보다 개인과 상호작용을, 포괄적인 문서보다 작동하는 소프트웨어를, 계약 협상보다 고객과의 협력을, 계획을 따르는 것보다 변화에 대응하는 것을 더 중요하게 여긴다는 것입니다. 이러한 가치는 오른쪽에 있는 것들도 가치가 있지만, 왼쪽에 있는 것들에 더 높은 가치를 둔다는 의미입니다.

첫 번째 가치인 개인과 상호작용은 사람을 중심에 두는 것입니다. 전통적인 조직에서는 프로세스를 정의하고 사람들이 그것을 따르도록 요구했습니다. 도구를 도입하고 사람들이 그것을 사용하도록 강제했습니다. 그러나 애자일은 사람이야말로 가장 중요한 성공 요소라고 믿습니다. 훌륭한 팀원들이 모여 활발하게 소통하고 협력할 때, 어떤 정교한 프로세스나 첨단 도구보다 더 나은 결과를 만들어낼 수 있습니다. 이는 얼굴을 맞대고 대화하는 것이 가장 효과적인 의사소통 방법이라는 믿음으로 이어집니다. 문서화된 절차나 이메일 체인보다 직접적인 대화가 오해를 줄이고 신뢰를 쌓습니다.

두 번째 가치인 작동하는 소프트웨어는 실질적인 가치 전달을 의미합니다. 전통적인 방식에서는 상세한 요구사항 명세서, 설계 문서, 아키텍처 문서를 작성하는 데 많은 시간을 소비했습니다. 문서가 완벽해야만 개발을 시작할 수 있다고 생각했습니다. 그러나 애자일은 고객이 진정으로 원하는 것은 문서가 아니라 실제로 사용할 수 있는 제품이라고 말합니다. 물론 문서도 필요하지만, 문서 작성 자체가 목적이 되어서는 안 됩니다. 문서는 소프트웨어를 만드는 데 도움이 되는 범위 내에서만 작성되어야

합니다. 작동하는 소프트웨어는 또한 진척도를 측정하는 가장 정확한 지표입니다. 백 페이지의 문서보다 실제로 동작하는 기능 하나가 프로젝트의 진행 상황을 더 명확하게 보여줍니다.

세 번째 가치인 고객과의 협력은 관계의 본질을 바꾸는 것입니다. 전통적인 계약 관계에서 고객과 공급자는 대립적인 위치에 서게 됩니다. 계약서에 명시된 것을 정확히 제공했느냐, 추가 요청은 추가 비용을 지불해야 한다는 식의 논쟁이 벌어집니다. 그러나 애자일은 고객을 적이 아니라 파트너로 봅니다. 함께 최선의 결과를 만들어가는 동료입니다. 고객은 개발 과정에 지속적으로 참여하며, 정기적으로 피드백을 제공합니다. 개발팀은 그 피드백을 환영하고 빠르게 반영합니다. 이러한 협력적 관계에서는 요구사항이 변경되는 것이 문제가 아니라 자연스러운 일입니다. 오히려 변경을 통해 더 나은 제품을 만들 수 있다는 기회로 받아들입니다.

네 번째 가치인 변화에 대한 응답은 불확실성을 수용하는 것입니다. 전통적인 프로젝트 관리에서는 프로젝트 시작 시점에 모든 것을 계획하고, 그 계획을 충실히 따르는 것이 성공의 열쇠라고 믿었습니다. 변경은 실패의 신호이며, 변경 요청은 통제되고 제한되어야 했습니다. 그러나 현실 세계는 예측 불가능합니다. 시장이 변하고, 기술이 발전하며, 경쟁사가 새로운 제품을 출시하고, 고객의 요구가 진화합니다. 프로젝트를 시작할 때 알지 못했던 것들을 개발 과정에서 배우게 됩니다. 애자일은 이러한 변화를 저항할 대상이 아니라 경쟁 우위의 원천으로 봅니다. 변화에 빠르게 대응할 수 있는 능력이야 말로 오늘날과 같은 급변하는 환경에서 생존하고 번영하는 핵심 역량입니다.

이러한 네 가지 가치를 구체화한 열 두 가지 원칙들은 애자일을 실천하는 방법을 더욱 명확히 합니다. 가장 우선순위가 높은 원칙은 가치 있는 소프트웨어를 일찍, 그리고 지속적으로 전달함으로써 고객을 만족시키는 것입니다. 이는 몇 년 간의 개발 끝에 완성된 제품을 한 번에 출시하는 것이 아니라, 작은 증분을 자주 릴리스하는 것을 의미합니다. 고객은 기다리지 않고 빠르게 가치를 경험할 수 있으며, 개발팀은 실제 사용자의 반응을 보면서 배우고 조정할 수 있습니다.

변화하는 요구사항을 환영한다는 원칙은 네 번째 가치를 구체화합니다. 개발 후반부라 할지라도 요구사항 변경을 받아들입니다. 애자일 프로세스는 변화를 활용해 고객의 경쟁 우위를 높입니다. 이것이 가능한 이유는 짧은 주기로 작업하기 때문입니다. 작동하는 소프트웨어를 몇 주에서 몇 개월 단위의 짧은 기간 내에 자주 인도합니다. 짧은 주기일수록 더 선호합니다. 이렇게 짧은 주기로 일하면 피드백 루프가 빨라지고, 잘못된 방향으로 가더라도 빨리 발견하고 수정할 수 있습니다.

비즈니스 담당자와 개발자는 프로젝트 전체에 걸쳐 매일 함께 일해야 한다는 원칙은 협력의 중요성을 강조합니다. 개발팀을 고립된 공간에 두고 가끔씩 진행 상황을 보고받는 것이 아니라, 비즈니스와 기술이 긴밀하게 통합되어야 합니다. 이를 통해 비즈니스 목표와 기술적 구현 사이의 간극을 줄일 수 있습니다.

동기부여된 개인들을 중심으로 프로젝트를 구성한다는 원칙은 사람에 대한 신뢰를 보여줍니다. 그들이 필요로 하는 환경과 지원을 제공하고, 그들이 일을 완수할 것이라고 믿습니다. 이는 명령과 통제가 아니라 자율성과 신뢰를 기반으로 합니다. 사람들은

스스로 생각하고 결정할 수 있을 때 가장 창의적이고 생산적입니다. 마이크로매니징은 동기부여를 떨어뜨리고 혁신을 저해합니다.

개발팀 내에서 정보를 전달하는 가장 효율적이고 효과적인 방법은 얼굴을 맞대고 대화하는 것이라는 원칙은 직접 소통의 가치를 재확인합니다. 이메일, 메신저, 문서로는 뉘앙스와 맥락이 사라지기 쉽습니다. 얼굴을 보며 대화할 때 비언어적 신호를 읽을 수 있고, 즉각적으로 질문하고 명확히 할 수 있습니다. 물론 원격 근무가 일반화된 오늘날에는 화상회의도 직접 대화의 한 형태로 볼 수 있습니다.

작동하는 소프트웨어가 진척의 주된 척도라는 원칙은 구체적인 결과에 초점을 맞춥니다. 회의를 몇 번 했는지, 문서를 몇 페이지 작성했는지, 코드를 몇 줄 작성했는지가 중요한 것이 아닙니다. 실제로 동작하고 가치를 제공하는 기능이 얼마나 완성되었는지가 중요합니다. 이는 투명성을 높입니다. 누구나 실제 제품을 보고 진행 상황을 평가할 수 있습니다.

애자일 프로세스는 지속 가능한 개발을 장려한다는 원칙은 장기적 관점을 제시합니다. 후원자, 개발자, 사용자 모두 일정한 속도를 계속 유지할 수 있어야 합니다. 단기적인 성과를 위해 사람들을 혹사시키는 것은 애자일이 아닙니다. 번아웃은 생산성을 떨어뜨리고 품질을 저하시키며 사람들을 떠나게 만듭니다. 지속 가능한 속도로 일할 때 사람들은 더 창의적이고, 더 행복하며, 더 오래 함께할 수 있습니다.

기술적 탁월성과 좋은 설계에 대한 지속적 관심이 민첩성을 높인다는 원칙은 품질의

중요성을 말합니다. 빠르게 움직이기 위해 품질을 희생해서는 안 됩니다. 오히려 높은 품질이 속도를 가능하게 합니다. 깨끗한 코드, 좋은 아키텍처, 자동화된 테스트는 변경을 쉽게 만들고 버그를 줄입니다. 기술 부채가 쌓이면 처음에는 빠를 수 있지만 결국 속도가 느려지고 변경이 위험해집니다. 지속적으로 품질에 투자하는 것이 장기적으로 더 빠른 길입니다.

단순성, 즉 하지 않아도 될 일의 양을 최대화하는 기술이 필수적이라는 원칙은 낭비 제거를 강조합니다. 필요하지 않은 기능을 만들지 말아야 합니다. 사용되지 않을 문서를 작성하지 말아야 합니다. 복잡한 해결책보다 단순한 해결책을 선호해야 합니다. 단순함은 이해하기 쉽고, 유지보수하기 쉬우며, 변경하기 쉽습니다. 복잡성은 적이며, 의도적으로 단순함을 추구해야 합니다.

최고의 아키텍처, 요구사항, 설계는 자기조직화하는 팀에서 나온다는 원칙은 창발적 설계를 믿습니다. 위에서 아래로 설계를 강요하는 것이 아니라, 실제로 일하는 사람들이 스스로 최선의 방법을 찾도록 합니다. 팀은 문제를 가장 깊이 이해하고 있으며, 스스로 해결책을 만들 능력이 있습니다. 자기조직화는 주인의식을 높이고, 창의성을 발휘하게 하며, 더 나은 결과로 이어집니다.

마지막 원칙은 팀이 정기적으로 어떻게 하면 더 효과적이 될지 성찰하고, 그에 따라 팀의 행동을 조율하고 조정한다는 것입니다. 이것이 바로 지속적 개선의 정신입니다. 완벽한 프로세스는 없으며, 항상 개선의 여지가 있습니다. 회고를 통해 무엇이 잘 되었고 무엇이 잘못되었는지 솔직하게 이야기하고, 실험하고, 배우고, 적응합니다.

이러한 학습 문화가 조직을 성장시킵니다.

애자일 선언이 있은 후 20년이 넘는 시간이 지났습니다. 애자일의 가치는 변하지 않지만, 이 가치에 기반해서 각 조직과 그 조직이 다른 제품 혹은 팀의 문화에 맞춰서 애자일을 도입하기 위한 시도가 있었고, 이런 시도는 정형화된 프레임워크로 정리되기도 했습니다. 처음부터 애자일의 가치를 이해하고, 그에 맞는 방법론을 적용하는 것도 애자일 도입하는 방법이지만, 이런 프레임워크를 잘 사용하는 것도 다른 적용 방식일 수 있습니다. 이를 위해서 지금부터는 어떤 프레임워크가 있는지 살펴보고, 애자일의 가치와 원칙을 기준으로 프레임워크 도입에 대한 이야기를 이어가 보겠습니다.

SAFe (Scaled Agile Framework)

SAFe는 Dean Leffingwell이 개발한 프레임워크로, 대규모 조직에서 애자일을 구현하기 위한 가장 포괄적이고 구조화된 접근 방식입니다. 이 프레임워크는 팀, 프로그램, 대규모 솔루션, 포트폴리오의 네 가지 레벨로 구성되어 있으며, 각 레벨마다 명확한 역할, 이벤트, 산출물을 정의하고 있습니다. SAFe는 린 사고방식, 애자일 개발, 제품 개발 플로우, 시스템 사고 등 여러 방법론의 원칙을 통합하여 제공합니다.

SAFe의 가장 큰 장점은 상세하고 규범적인 가이드를 제공한다는 점입니다. 대규모 조직이 애자일 전환을 시작할 때 어디서부터 시작해야 할지 막막한 경우가 많은데, SAFe는 단계별로 구체적인 실행 방법을 제시합니다. PI(Program Increment) 플래닝과 같은 구조화된 이벤트를 통해 여러 팀 간의 조율과 정렬을 효과적으로 수행할 수 있습니다. 또한 광범위한 교육 프로그램과 인증 체계가 갖춰져 있어 조직 전반에 걸쳐

일관된 이해를 확산시키기 용이합니다. 특히 전통적인 워터폴 방식에 익숙한 대기업들이 애자일로 전환할 때 기존 구조와의 호환성을 유지하면서 점진적으로 변화할 수 있다는 점도 강점입니다.

그러나 SAFe에 대한 비판도 존재합니다. 가장 큰 비판은 프레임워크가 지나치게 무겁고 복잡하다는 점입니다. 애자일의 핵심 가치인 단순성과 유연성을 오히려 해친다는 지적이 많습니다. 많은 애자일 순수주의자들은 SAFe를 "워터폴에 애자일을 차용한 것"이라고 이야기를 하기도 합니다. 실제로 SAFe를 도입한 조직들이 형식과 절차에만 집중하고 애자일의 본질적인 마인드셋 변화는 이루지 못하는 경우가 있습니다.

또한 SAFe는 도입 시 비용이 듭니다. 교육, 컨설팅, 인증 등에 적지 않은 투자가 필요하며, 이 이유로 예산이 충분하지 않은 조직에서는 쉽게 도입하기 어렵다는 관점도 있습니다. 하향식 계획과 통제를 강조하는 측면이 있어 진정한 자율성과 자기조직화를 저해할 수 있다는 우려도 존재합니다.

LeSS (Large-Scale Scrum)

LeSS는 Craig Larman과 Bas Vodde가 개발한 프레임워크로, 스크럼의 원칙과 가치를 대규모 환경에서도 그대로 유지하려는 접근 방식입니다. LeSS의 철학은 "스크럼에 더하는 것이 아니라 빼는 것"으로 요약됩니다. 8개 팀 이하를 위한 기본 LeSS와 8개 팀 이상을 위한 LeSS Huge로 구성되며, 하나의 제품 백로그, 하나의 제품 오너, 하나의 스프린트로 여러 팀이 함께 일합니다.

LeSS의 주요 장점은 단순성입니다. SAFe처럼 새로운 역할이나 계층을 추가하지 않고, 스크럼의 본질을 유지하면서 확장합니다. 이는 조직의 복잡성을 줄이고 투명성을 높이는 데 기여합니다. LeSS는 시스템 사고를 강조하며, 부분 최적화보다 전체 최적화에 초점을 맞춥니다. 팀 간의 조율을 위해 무거운 프로세스를 추가하는 대신, 팀들이 직접 협력하고 커뮤니케이션하도록 장려합니다. 또한 조직 디자인 원칙을 제시하여 애자일을 지원하는 조직 구조로의 변화를 촉진합니다. 상업적 인증이 없고 오픈 소스 정신으로 운영되어 비용 부담이 적다는 것도 장점입니다.

그러나 LeSS의 한계는 그 단순성에서 비롯됩니다. 구체적인 가이드가 적기 때문에 조직이 스스로 많은 것을 파악하고 실험해야 합니다. 특히 애자일 성숙도가 낮은 조직에서는 실행이 어렵습니다. LeSS는 근본적인 조직 변화를 요구하는데, 기존 관리 계층의 축소, 기능 조직에서 피처 팀으로의 전환 등 저항이 큰 변화들입니다. 실제로 많은 조직들이 이러한 구조적 변화를 감당하지 못해 LeSS 도입에 실패합니다. 또한 하나의 제품 오너가 모든 것을 관리하는 것은 매우 큰 규모에서는 현실적으로 어려울 수 있습니다. LeSS는 이상적이지만 실용적이지 않다는 비판을 받기도 하며, 특히 전통적인 대기업 환경에서는 너무 급진적인 변화로 받아들여질 수 있습니다.

Disciplined Agile (DA)

Disciplined Agile은 Scott Ambler와 Mark Lines가 개발했으며, 2019년 PMI에 인수되었습니다. DA는 단일한 방법론이 아니라 상황에 맞는 선택을 할 수 있도록 돕는 의사결정 프레임워크입니다. "골 주도 접근"을 채택하여 조직이 달성해야 할 목표를 제시하고, 각 목표를 달성하기 위한 여러 옵션을 제공합니다. DA는 스크럼, 칸반, 린, SAFe, XP 등 다양한 방법론의 요소를 포함하는 하이브리드 툴킷입니다.

DA의 강점은 유연성과 실용주의입니다. 모든 조직과 팀이 다르다는 것을 인정하고, 획일적인 해결책 대신 선택의 프레임워크를 제공합니다. 이는 조직이 자신의 컨텍스트에 맞는 방식을 찾도록 돕습니다. DA는 전체 엔터프라이즈 수명주기를 다루며, 단순히 소프트웨어 개발뿐 아니라 데브옵스, 데이터 관리, 거버넌스, 재무, 인사 등 기업의 모든 측면을 포괄합니다. 또한 조직이 이미 사용하고 있는 방법론과 통합할 수 있어, 기존 투자를 보호하면서 개선할 수 있습니다.

하지만 DA의 복잡성은 양날의 검입니다. 너무 많은 선택지와 옵션은 오히려 혼란을 야기할 수 있습니다. 특히 애자일 경험이 적은 조직은 어떤 옵션을 선택해야 할지 판단하기 어려워할 수 있습니다. DA를 효과적으로 활용하려면 상당한 학습과 경험이 필요합니다. 또한 DA는 다른 프레임워크들만큼 커뮤니티가 크지 않아 참고할 수 있는 사례나 자료가 상대적으로 부족합니다. 아울러 프레임워크가 포괄적인 만큼 실제로 모든 것을 이해하고 적용하는 것은 현실적으로 어렵다는 지적도 있습니다.

Scrum@Scale

Scrum@Scale은 스크럼의 공동 창시자인 Jeff Sutherland가 개발한 프레임워크로, 스크럼을 조직의 모든 레벨로 확장하는 방식입니다. 이 프레임워크는 "스크럼 오브 스크럼"을 기반으로 하며, 두 개의 사이클로 구성됩니다. 즉, 스크럼 마스터 사이클과 제품 오너 사이클이 독립적으로 확장되면서도 상호작용합니다. 프랙탈 구조를 채택하여 5명에서 5천 명까지 동일한 패턴으로 확장할 수 있도록 설계되었습니다.

Scrum@Scale의 장점은 스크럼의 순수성을 유지한다는 점입니다. 새로운 역할이나 이벤트를 최소한으로 추가하며, 스크럼 가이드를 기반으로 합니다. 프랙탈 구조는 이론적으로 무한히 확장 가능하며, 조직의 크기에 관계없이 동일한 원칙을 적용할 수 있습니다. SAFe처럼 고정된 릴리스 트레인이 아니라 각 팀이 자신의 속도로 릴리스 할 수 있어 더 유연합니다. EAT(Executive Action Team)와 EMS(Executive MetaScrum)를 통해 경영진을 애자일 프로세스에 직접 참여시키는 것도 특징입니다. 비교적 가벼우면서도 명확한 구조를 제공한다는 평가를 받습니다.

그러나 Scrum@Scale은 상대적으로 덜 성숙한 프레임워크입니다. SAFe나 LeSS에 비해 역사가 짧고 적용 사례가 적습니다. 특히 대규모 조직에서의 검증된 성공 사례가 부족합니다. 프레임워크의 가이드가 다소 추상적이고 구체적인 구현 방법에 대한 설명이 부족하다는 지적도 있습니다. 두 개의 독립적인 사이클이 실제로 잘 조율되지 않으면 오히려 혼란이 가중될 수 있습니다. 또한 Jeff Sutherland라는 개인에게 크게 의존하고 있어, 다른 프레임워크만큼 커뮤니티 기반이 강하지 않다는 한계가 있습니다. 상업적 인증 프로그램도 운영되고 있어 비용 문제도 고려해야 합니다.

Spotify Model

Spotify Model은 엄격히 말하면 프레임워크라기보다는 Spotify가 자사의 조직을 어떻게 구성했는지에 대한 사례입니다. 이 모델은 Squad(팀), Tribe(부족), Chapter(챕터), Guild(길드)라는 독특한 용어를 사용합니다. Squad는 크로스펑셔널한 자율적 팀으로 특정 미션을 수행하며, 여러 Squad가 모여 Tribe를 형성합니다. Chapter는 같은 역량을 가진 사람들의 모임이고, Guild는 관심사를 공유하는 사람들의 커뮤니티입니다. 이는 기능 조직과 제품 조직의 장점을 결합한 매트릭스 구조라고 볼 수

있습니다.

Spotify Model의 매력은 자율성과 정렬의 균형을 추구한다는 점입니다. Squad에게 높은 자율성을 부여하면서도 Tribe를 통해 방향성을 정렬합니다. Chapter와 Guild를 통해 지식 공유와 전문성 개발을 촉진하며, 사일로를 방지합니다. 문화와 가치를 중시하며, 실험과 학습을 장려하는 환경을 만듭니다. 특히 기술 기업의 혁신적인 문화를 상징하는 모델로 인식되어 많은 조직들이 벤치마크하고 싶어 합니다. 공식적인 인증이나 비용이 없어 자유롭게 참고하고 적용할 수 있다는 것도 장점입니다.

그러나 Spotify Model의 한계는 명확합니다. 가장 큰 문제는 Spotify 자신도 더 이상 이 모델을 그대로 사용하지 않는다는 점입니다. Spotify의 여러 직원들이 이 모델이 과대평가되었으며, 실제로는 많은 문제점이 있었다고 공개적으로 언급했습니다. 이 모델은 Spotify의 특정 시점, 특정 컨텍스트에서 작동했던 것이지 보편적인 해결책이 아닙니다. 많은 조직들이 Squad와 Tribe라는 용어만 채택하고 본질적인 문화 변화는 이루지 못하는 "카고 컬트"에 빠집니다. 매트릭스 구조는 책임 소재가 불명확해지고 의사결정이 느려질 수 있습니다. 또한 명확한 가이드나 구현 방법이 없어 조직이 스스로 많은 것을 해석하고 적용해야 하는 어려움이 있습니다.

Nexus

Nexus는 Scrum.org의 Ken Schwaber가 개발한 프레임워크로, 3개에서 9개의 스크럼 팀이 하나의 제품을 개발할 때 사용하도록 설계되었습니다. Nexus는 스크럼 위에 최소한의 것만 추가하는 경량 프레임워크입니다. Nexus Integration Team이라는 역할을

도입하여 팀 간 통합과 의존성 관리를 지원하며, Nexus Sprint Planning, Nexus Daily Scrum, Nexus Sprint Review, Nexus Sprint Retrospective 등 통합 이벤트를 추가합니다.

Nexus의 강점은 명확한 범위와 단순성입니다. 3-9개 팀이라는 제한된 범위에 집중함으로써 불필요한 복잡성을 피합니다. 스크럼 가이드에 충실하며, 스크럼을 이미 잘 실행하고 있는 팀들에게 자연스러운 확장 경로를 제공합니다. Scrum.org의 공식 프레임워크로서 명확한 정의와 교육 자료가 잘 갖춰져 있습니다. SAFe처럼 무겁지 않으면서도 LeSS처럼 추상적이지 않은 중간 지점을 제공한다고 볼 수 있습니다. 특히 중소 규모의 제품 개발에 적합하며, 도입 장벽이 낮습니다.

그러나 Nexus의 한계는 확장성입니다. 9개 팀 이상으로는 확장할 수 없기 때문에 대규모 조직에는 적합하지 않습니다. 실제로 많은 대기업들은 수십 개 이상의 팀을 조율해야 하는데, Nexus는 이런 상황을 다루지 못합니다. 또한 Nexus Integration Team의 역할이 모호할 수 있습니다. 이 팀이 병목이 되거나 다른 팀의 자율성을 침해할 위험이 있습니다. 포트폴리오 관리, 거버넌스, 예산 등 엔터프라이즈 레벨의 문제들은 다루지 않습니다. 상대적으로 새롭고 커뮤니티가 작아 참고할 수 있는 사례와 경험이 제한적이라는 점도 한계입니다.

성공적인 애자일을 도입을 위해서

이러한 다양한 프레임워크들은 모두 대규모 조직에서 애자일을 실현하려는 목표를 공유하지만, 접근 방식과 철학이 다릅니다. SAFe는 규범적이고 구조화된 접근을, LeSS는

단순성과 원칙 중심 접근을, Disciplined Agile은 상황별 선택을, Scrum@Scale은 프랙탈 확장을, Spotify Model은 문화와 자율성을, Nexus는 제한된 범위에서의 통합을 강조합니다. 조직이 어떤 프레임워크를 선택할지는 조직의 규모, 문화, 성숙도, 산업 특성, 변화 준비도 등 여러 요인을 종합적으로 고려해야 합니다. 중요한 것은 프레임워크 자체가 아니라 애자일의 본질적인 가치와 원칙을 이해하고 실천하는 것이며, 프레임워크는 그것을 돕는 도구일 뿐이라는 점을 잊지 말아야 합니다.

이번 파트 제일 처음에 말씀드린 바는 애자일의 가치를 실천하는 것이 애자일의 본질이라는 점입니다. SAFe, LeSS, Disciplined Agile, Scrum@Scale, Spotify Model, Nexus와 같은 프레임워크들은 이러한 가치와 원칙을 대규모 조직에서 실현하기 위한 도구입니다. 그러나 프레임워크를 도입하는 것 자체가 애자일이 되는 것은 아닙니다. 프레임워크의 형식만 따르면서 본질을 놓치는 것을 "애자일 워싱"이라고 부릅니다. 겉으로는 애자일 용어를 사용하고 애자일 행사를 진행하지만, 여전히 명령과 통제로 운영되고, 변화를 두려워하며, 사람들을 자원으로 취급하는 조직들이 있습니다.

진정한 애자일 전환은 마인드셋의 변화를 요구합니다. 예측 가능성과 통제에서 적응과 학습으로, 효율성과 최적화에서 효과성과 가치로, 개인의 성과에서 팀의 협력으로, 단기적 결과에서 장기적 지속가능성으로 사고방식을 바꿔야 합니다. 이는 편안한 변화가 아닙니다. 기존의 권력 구조가 흔들리고, 익숙한 방식을 버려야 하며, 불확실성을 받아들여야 합니다. 그러나 이러한 변화 없이는 아무리 좋은 프레임워크를 도입해도 진정한 애자일의 혜택을 누릴 수 없습니다.

프레임워크를 선택할 때 가장 중요한 질문은 "이 프레임워크가 우리 조직에 애자일의 가치와 원칙을 실현하는 데 도움이 되는가?"입니다. SAFe가 구조화된 접근을 제공한다면, 그것이 사람들의 자율성을 침해하지 않고 오히려 협력을 촉진하는가? LeSS가 단순성을 추구한다면, 그것이 조직의 현재 상태에서 현실적으로 적용 가능한가? Spotify Model이 자율성을 강조한다면, 우리 조직의 리더들이 진정으로 통제를 내려놓을 준비가 되어 있는가? 이러한 질문들에 답하면서 프레임워크를 선택하고 적용해야 합니다.

또한 프레임워크를 교조적으로 따를 필요는 없습니다. 애자일의 원칙 중 하나가 회고를 통한 지속적 개선입니다. 프레임워크 역시 조직의 맥락에 맞게 조정되고 진화해야 합니다. 어떤 조직은 SAFe의 구조를 빌려오되 더 가볍게 만들 수 있고, 다른 조직은 LeSS의 원칙을 따르되 점진적으로 적용할 수 있습니다. 중요한 것은 프레임워크의 노예가 되는 것이 아니라, 프레임워크를 활용해 애자일의 가치를 실현하는 것입니다.

궁극적으로 애자일은 고객에게 더 나은 가치를 더 빠르게 전달하고, 변화하는 환경에 효과적으로 대응하며, 사람들이 의미 있고 지속 가능한 방식으로 일할 수 있게 하는 것입니다. 프레임워크는 그 여정을 돕는 지도일 뿐, 목적지 자체가 아닙니다. 조직이 애자일 프레임워크를 도입할 때 이러한 본질을 잊지 않고, 형식보다 정신을, 도구보다 사람을, 계획보다 학습을 우선시한다면, 어떤 프레임워크를 선택하든 성공적인 애자일 여정을 시작할 수 있을 것입니다.

Part 2.

자동차 분야에서
애자일 도입 사례

들어가며

이번 파트는 폭스바겐, 메르세데스-벤츠, BMW, 테슬라의 애자일을 도입하는 사례를 정리해 봤습니다. 애자일 방법의 적용 사례가 그렇듯이, 각각의 적용 사례는 잘된 것도 있고 기대에 미치지 못한 것도 있습니다. 이런 사례 연구를 중심으로 현장에서 애자일을 도입할 때 어떤 점에 주목해야 할지 정리해 봤습니다.

폭스바겐 사례

폭스바겐 그룹의 애자일 도입 사례는 전사적인 디지털 전환 노력과 핵심 소프트웨어 조직(CARIAD)의 실패라는 두 가지 상반된 축으로 서술될 수 있습니다.

폭스바겐의 전사적 전환과 애자일 원칙 도입

폭스바겐 그룹은 자동차 제조업체에서 e-모빌리티 제공업체로의 전환이라는 회사 역사상 가장 큰 변혁 과정에 직면해 있습니다. 이 변화를 촉진하기 위해 폭스바겐은 애자일 소프트웨어 개발 원칙에 기반한 신속한 설계 및 개발 프로세스를 도입했습니다. 핵심 목표는 시장 출시 기간(time-to-market)을 단축하고, 사용자 중심성을 높여 고객에게 탁월한 사용자 경험을 제공하는 것입니다.

기술 개발(Technical Development, TD) 부문의 재편 폭스바겐은 볼프스부르크의 TD 부문(11,500명 규모의 최대 엔지니어링 조직)을 시스템 및 기능 중심으로 재설계했습니다. 이는 구성요소 중심이 아닌 소프트웨어 중심, 고객 요구사항 중심의 접근 방식인 시스템 엔지니어링(systems engineering)을 적용하는 것을 의미합니다. 이러한

시스템 엔지니어링과 애자일 방법론에 대한 집중을 통해 차량 개발 시간을 기존 54개월에서 약 25% 단축된 40개월로 줄이는 것을 목표로 합니다. 또한, 약 8억 유로를 투자하여 볼프스부르크에 애자일 업무 구조를 위한 캠퍼스 잔트캄프 (Campus Sandkamp)를 건설할 계획입니다.

구체적인 애자일 및 린(Lean) 적용 사례

애자일 UX 테스트 (User Days): 폭스바겐은 기업의 애자일 프로세스 모델에 맞는 UX 설계 및 품질 보증 프로세스를 개발하기 위해 협력했습니다. 이전에는 스프린트 계획에 사용자 연구 및 유용성 테스트가 포함되지 않았으나, 이제는 '유저 데이(User Days)'라는 애자일 UX 테스트 형식을 개발했습니다. 이 테스트는 월 2~3일 특정 대상 그룹 사용자들이 현장 테스트 스튜디오에서 피드백을 제공하도록 일정을 고정하고, 제품 팀은 현재 스프린트의 질문에 맞춰 자발적으로 사용자 피드백을 요청할 수 있습니다.

설계 및 생산: 폭스바겐은 순차적인 워터폴 방식에서 벗어나 짧은 사이클(보통 2~4주)의 스프린트를 기반으로 한 반복적이고 협력적인 애자일 원칙을 차량 설계 프로세스에 적용했습니다. 예를 들어, ID.3 전기차 개발 프로젝트는 크로스 펑셔널 팀이 일일 스탠드업 회의를 통해 진행 상황을 논의하며 신속하게 프로토타입을 제작했습니다. 또한, 도요타가 개척한 린(Lean) 원칙을 생산에 통합하여 재고를 최소화하는 적시 생산(JIT) 시스템을 운영하고, 지속적인 개선(Kaizen) 문화를 통해 직원들이 개선 사항을 제안하도록 독려하고 있습니다.

IT 및 테스트 환경: IT 부문에서는 애자일 3개월 스프린트(trimester sprints)에 중점을

둔 'Global IT Top 10' 프로그램을 통해 비즈니스 영향과 속도를 높이고 있습니다. 또한, Red Hat 기술을 사용하여 가상 및 실제 테스트를 결합한 혼합 모드 테스트 환경을 구축했습니다. 이는 소프트웨어 기능의 조기 통합 테스트를 가능하게 했으며, 시스템 테스트 비용을 50% 절감하고 팀 간 협업을 개선하는 데 기여했습니다.

소프트웨어 중앙 조직 CARIAD의 난관

2020년, 폭스바겐은 그룹 전체를 위한 중앙 소프트웨어 조직인 CARIAD를 설립하여 완벽한 비전을 제시했습니다: 모든 브랜드(VW, Audi, Porsche)를 위한 단일 중앙 소프트웨어 아키텍처 구축. 그러나 실행 단계에서 심각한 문제에 봉착했습니다.

초기 혼란과 급속한 확장: CARIAD는 작은 규모에서 시작하는 대신 급격하게 확장하여 수천 명의 직원을 VW, Audi, Porsche 및 외부에서 이관받았으나, 명확한 개념, 정의된 역할, 실질적인 권한이 없었습니다. 신규 채용은 몇 달 만에 6,000명으로 급증했지만, 많은 관리자가 하드웨어 출신이었고 애자일 방법론은 생소한 개념이었습니다.

구조적 결함: CARIAD는 비용을 브랜드로부터 조달받았기 때문에 소프트웨어를 개발해야 했지만 의사 결정 권한이 없었습니다. 모든 권한은 여전히 돈을 지불하는 브랜드에 있었습니다. 그 결과, CARIAD 내부는 각 브랜드(Audi, Porsche, VW)의 구조와 프로세스를 재현한 '미니 기업들의 조합(patchwork of mini-corporations)'이 되었고, 브랜드 간의 싸움이 지속되어 동일한 기능을 여섯 번씩 개발하는 비효율이 발생했습니다.

레거시 부담: CARIAD는 미래 플랫폼(Platform 2.0)을 구축해야 했지만, 2021년 Audi와 Porsche가 실패했던 플랫폼 1.1 및 1.2와 같은 복잡하고 문제가 많은 레거시 프로젝트를 떠맡게 되면서, 혁신 대신 '소방수 역할(fighting fires)'에만 집중하게 되었습니다.

위기와 현재 상황: 소프트웨어 개발 지연으로 모델 출시가 연기되었고, 2022년에는 맥킨지 보고서를 통해 구조적 문제가 드러났습니다. 2023년 새로운 경영진이 구조 조정을 시도하고 2,000명 감원을 발표했으나, 2024년 폭스바겐이 CARIAD를 배제하고 Rivian과의 대규모 합작 투자(JV)를 체결하면서 상황은 악화되었습니다. 이로 인해 CARIAD의 혁신 프로젝트는 Rivian JV로 넘어가고, CARIAD는 2029년까지 기존 소프트웨어 유지 보수(legacy maintenance)와 같은 축소된 역할만 담당하게 되었습니다. 현재 대규모 감원 프로그램이 진행 중이며, 직원들의 동기 부여는 저하된 상태입니다.

전환의 성공과 실패로부터의 교훈

폭스바겐의 애자일 전환 노력과 CARIAD의 사례는 전통적인 자동차 제조업체가 소프트웨어 중심 기업으로 변모할 때 겪는 근본적인 어려움을 명확하게 보여줍니다.

권력과 재정의 독립성 확보의 중요성 진정한 혁신에는 실질적인 독립성이 필수적입니다. CARIAD의 실패 원인 중 하나는 자체 예산 없이 브랜드를 통해 자금을 조달하면서 브랜드에 모든 권한이 남아있었다는 점입니다. 최고 경영진이 내부 권력 다툼을 극복하지 못하면 어떤 전환도 실패할 수밖에 없습니다.

문화 및 리더십의 전환 부재 많은 경영진이 하드웨어 배경을 가지고 있었고 소프트웨어 및 애자일 방법론에 익숙하지 않았습니다. 기존 기업의 관리자들을 소프트웨어 회사 문화로 강제 전환하려 한 결과, 비싸기만 한 기업의 복제품이 탄생했습니다. 애자일은 기술적인 변화뿐만 아니라 조직 내의 문화적 변화를 요구하며, 이는 포괄적인 교육과 변화 관리 노력을 통해 해결해야 합니다.

레거시와 혁신의 엄격한 분리 레거시 프로젝트(기존 시스템)와 혁신 프로젝트를 한 지붕 아래 두는 것은 효과적이지 않았습니다. CARIAD가 문제 플랫폼(1.1 및 1.2)을 떠안으면서 미래 아키텍처(2.0) 개발 능력이 마비된 것이 그 예입니다. 성공적인 전환을 위해서는 레거시와 혁신 간의 엄격한 분리가 필요합니다.

작은 팀과 권한 부여 실질적인 의사 결정 권한 없이 6,000명이나 되는 대규모 조직은 혼란만을 야기했습니다. 폭스바겐은 CARIAD 사례를 통해 진정한 책임을 가진 작은 규모의 팀이 더 많은 성과를 달성할 수 있음을 배워야 합니다.

내재화의 어려움 (Make or Buy) CARIAD는 소프트웨어 회사로 출범했음에도 불구하고, 실제로는 코딩을 거의 하지 않고 외부 서비스 제공업체에 의존하는 '비싼 중개인 (expensive middleman)' 역할을 했습니다. 기술 기업과의 파트너십을 배제하고 인하우스 개발을 고집했지만, 결국 핵심 목표를 달성하지 못했습니다. 이는 대기업이 복잡한 기술 결정에서 내부 기술 전문성의 한계를 인정하고, 시장 혁신을 보완하며 활용하는 것이 중요함을 보여줍니다.

메르세데스-벤츠 사례

메르세데스-벤츠의 애자일 방법론 도입은 전통적인 자동차 산업의 한계를 돌파하고 디지털 전환을 가속화한 주요 사례로 평가받습니다. 그러나 벤츠가 제시하는 성과와 주장을 원자료에 기반하여 비판적인 시각으로 분석하면, 이러한 전환이 가지는 구조적 한계와 성공 요인의 복합성을 파악할 수 있습니다.

메르세데스-벤츠는 2017년부터 글로벌 운영 전반에 걸쳐 애자일 전환을 시작했으며, 이는 자동차 제조업체에서 모빌리티 서비스 제공업체로 변모하기 위한 광범위한 디지털 전략의 일환이었습니다.

프레임워크 및 규모

벤츠는 대규모 조직의 민첩성을 유지하고 글로벌 정렬을 보장하기 위해 SAFe를 핵심 조정 메커니즘으로 구현했습니다. 2017년 고객 대면 도메인에서 시작된 SAFe는 현재 포트폴리오 수준까지 확장되어, 전 세계 6개 기술 허브에서 1,000명 이상의 기술 전문가를 조정하는 데 사용됩니다. 이들은 90일 주기의 프로그램 증분(PI)과 애자일 릴리스 트레인(ART)을 통해 전 세계 팀을 동기화합니다.

팀 구조 및 개발 접근 방식

개발은 8~10명으로 구성된 교차 기능적 애자일 포드 및 스쿼드를 중심으로 이루어지며,

이들은 2주 기반의 스프린트 개발 주기를 가집니다. 각 팀은 특정 제품 영역을 제공할 완전한 권한(autonomous decision-making)을 가집니다. 벤츠는 또한 팀원들이 배포 스쿼드와 넓은 실습 커뮤니티(예: 백엔드) 모두에 소속되도록 하는 "투 홈(two-home) 개념"을 활용하여 지식 공유와 팀 집중도를 동시에 확보합니다.

하이브리드 방법론과 기술 투자

벤츠는 애자일이 자동차 개발의 모든 측면에 적합하지 않음을 인식하고 하이브리드 애자일-워터폴 방법론을 채택했습니다. 소프트웨어 시스템(예: 인포테인먼트 시스템)은 순수 스크럼(Scrum) 및 신속한 반복을 따르는 반면, 하드웨어 개발이나 안전 필수 시스템은 규제 및 안전 요구 사항을 충족하기 위해 수정된 워터폴 접근 방식이나 구조화된 마일스톤을 유지합니다.

기술적으로는 개발 속도를 높이기 위해 CI/CD 파이프라인, 자동화된 테스트, 그리고 물리적 테스트 이전에 실제 조건을 시뮬레이션하는 디지털 트윈(Digital twins)을 도입했습니다. 특히 OTR(One Touch Retail) 프로젝트에서는 소프트웨어 배포 빈도를 월 1회에서 하루에 여러 번으로 늘리는 지속적 배포(continuous delivery) 프로세스를 실현했습니다.

시사점

벤츠의 애자일 전환 사례는 시간 단축, 품질 향상, 고객 만족도 개선 등 다양한 혜택을 주장하고 있지만, 이러한 혜택을 단순히 애자일 방법론 자체의 직접적인 결과로 해석하는 것에 대해서는 비판적인 검토가 필요합니다.

속도 향상(Time-to-Market)의 성과

벤츠는 2017년 이후 제품 출시 역량이 600% 증가하고, 금융 승인 처리 시간이 며칠에서 2.3분으로 단축되었다는 구체적인 성과를 제시합니다. 이러한 혁신적인 속도 향상은 애자일 방법론의 적용보다는 근본적인 기술 인프라와 아키텍처의 전환에서 비롯된 일회성 구조 개혁의 결과일 가능성이 높습니다.

참고 문헌들은 애자일 전환과 함께 n-tier 아키텍처를 마이크로 서비스 아키텍처로, 로컬 인프라를 클라우드 기반 구성 요소로 전환하고, 수동 테스트를 완전 자동화된 테스트로 대체했으며, CI/CD 파이프라인을 구축했음을 명시합니다. 즉, 속도의 폭발적 증가는 애자일 원칙 적용의 결과라기보다는, DevOps와 클라우드 기반 인프라를 구현함으로써 달성한 기술적 기반의 현대화에 더 크게 의존했을 수 있습니다.

또한 "가장 중요한 것은 애자일 원칙을 적용하는 것"이라고 강조하면서도, 마이크로 서비스 아키텍처와 같은 특정 기술적 전제 조건 없이는 팀 자율성(autonomy) 강화나 빈번한 배포 자체가 불가능했음을 인정하고 있습니다.

문화적 주장과 조직적 피로도의 상충 가능성

벤츠는 리더십이 마이크로매니지먼트에서 전략 수립으로 전환되었고, 실험과 학습을 장려하는 "실패해도 안전한 환경(safe-to-fail environment)"을 조성했다고 주장합니다. 이로 인해 직원 만족도가 25% 향상되고 기술 인재 이직률이 30% 감소했다고

보고했습니다. 벤츠는 5가지 패러다임 전환을 동시에 해결해야 한다고 명시했으며, 이러한 광범위한 동시 변화(waterfall에서 agile로, 프로젝트에서 제품으로, 개발/운영에서 DevOps로 등)는 일반적으로 "조직 변화 피로(organizational change fatigue)"를 유발할 수 있다고 스스로 언급합니다.

즉 조직 전체에 걸쳐 수많은 변화를 동시에 강요하는 구조적 압력과, 팀 단위에서 주장되는 "실패해도 안전한 환경" 사이에 긴장이 존재할 수 있습니다. 특히 벤츠는 TwiceAsFast와 같이 속도를 전략적 이점으로 활용하는 것을 강조했는데, 이러한 명확한 고속 성과 요구는 실패에 대한 심리적 안전망을 약화시키고 팀에 실질적인 압박으로 작용할 가능성이 있습니다.

SAFe 도입의 유연성 제한 가능성

벤츠는 애자일 방법론을 엄격하게 따르기보다 애자일 원칙에 집중하는 것이 중요하며, 팀들에게 특정 방법론(스크럼, 칸반)이나 기술(Clojure, Kotlin)을 선택할 수 있는 자유를 부여했다고 설명합니다. SAFe는 대규모 조직의 정렬을 위해 설계된 프레임워크로, 90일 단위의 PI 계획과 ART를 통한 엄격한 조정 메커니즘을 요구합니다.

참고 문헌들은 SAFe가 'faux agile' 또는 'cargo cult agile'이 될 수 있다는 비판적 시각이 있음을 인정합니다. 벤츠가 개별 팀에 유연성을 부여했다고 주장하더라도, SAFe의 중앙 집중적인 포트폴리오 수준 OKR(Objectives and Key Results) 설정과 정기적인 동기화 세션은 필연적으로 팀의 완전한 자율성과 창의성(Design Thinking에서 요구되는)을 특정 방향으로 강하게 정렬시키며, 결과적으로 SAFe의

무거운 구조가 개별 팀의 진정한 유연성을 제한하는 요소로 작용할 수 있습니다. 벤츠가 주장하는 "원칙 적용"이 SAFe의 구조적 요구 사항 내에서만 허용되는 제한적인 자유일 수 있다는 비판적 검토가 필요합니다.

BMW 사례

BMW 그룹은 소프트웨어 개발에서 전통적인 워터폴 모델을 포기하고 '100% 애자일'로 전환하는 과정을 겪고 있습니다. 이는 시장 변화와 고객의 새로운 기능에 대한 빈번한 요구 증가에 대응하기 위한 조치였습니다. BMW는 소프트웨어 제공의 유연성을 확보하고, 새로운 것을 빠르게 시도하며, 신속히 학습하여 속도를 높이는 것을 목표로 하고 있습니다.

이러한 전환은 프로세스, 구조, 기술, 문화의 네 가지 핵심 영역의 변화를 수반하였습니다. 특히 IT 부문에서는 애자일과 비(非)애자일 팀이 함께 일하는 데서 발생하는 비효율성을 극복하기 위해 노력하였으며, DevOps 조직 구조를 구현하였습니다. BMW 그룹은 여러 부서와 프로젝트에서 애자일 방식을 도입하였으며, 그중 주요 사례는 다음과 같습니다:

자율 주행 부서 (ADD)의 LeSS Huge 도입

BMW 그룹은 자율 주행(Autonomous Driving, AD) 부서에서 LeSS를 채택하여 조직 설계를 크게 변경하였습니다. 이 여정은 2016년 중반부터 2019년 10월까지 진행되었으며, 조직 설계의 목표는 경쟁사보다 빠르게 학습하고, 가장 높은 고객 가치를 제공하며, 쉽게 적응 가능한 조직을 만드는 것이었습니다.

ADD는 LeSS Huge 조직 구조를 설계하며, 기존의 계층 구조 레벨을 제거하고(C-4 레벨 팀 리더 역할 제거), Product Owner(PO), 개발 부서(Development Department), 역량 및 코칭 부서(Competence and Coaching Department)의 세 가지 주요 부서를 구성하였습니다. 도입 초기에는 하나의 요구사항 영역(Requirement Area)을 중심으로 시작하여 단계적으로 확장하는 원칙을 따랐습니다.

통합 판매 플랫폼 (Unified Sales Platform, USP) 구축

2012년 2월, BMW i 자동차의 새로운 직접 판매 프로세스를 지원하는 USP 시스템 개발이 스크럼 및 애자일 엔지니어링 관행을 사용하여 시작되었습니다. USP는 30개 이상의 외부 시스템 인터페이스를 통합하였으며, 초기에는 전통적인 프로그램 관리 시스템(BMW i 프로그램)에 포함되어 있었고, 다른 협력 프로젝트는 비(非)애자일 모델을 유지하였습니다. USP는 2년 이상의 개발 끝에 높은 고객 만족도와 높은 품질로 제때 출시되었습니다. 코치들은 이후 더 큰 릴리스를 위해 USP 프로젝트 그룹을 Multiple Feature Teams 및 LeSS 채택으로 재편성할 것을 제안하였으며, 이는 승인되었습니다.

남아프리카 온라인 판매 웹사이트 구축

2019년 5월부터 12월까지 BMW 남아프리카는 3개국 9개 팀, 75명의 인력이 협력하여 고객이 집에서 구매, 거래, 시승 예약을 할 수 있는 최초의 엔드투엔드 온라인 판매 웹사이트를 구축하였습니다. 이는 기존 사일로 방식으로 6개월간 진전이 없던 팀이 애자일 및 Scrum 원칙을 도입하여 성공적으로 전환한 사례입니다.

IT 인프라 및 도구 활용

BMW는 애자일 툴체인(Agile Toolchain)을 IT 개발의 기반으로 구축하였습니다. 2018년 6월부터 20,000명의 사용자가 이 모델을 활용하였으며, 3,000개 이상의 서브 프로젝트를 지원하였습니다. 또한, 2011년부터 Planview와 협력하여 가치 스트림 관리(VSM)와 애자일 작업 방식의 데이터 통합 문제를 해결하였으며, 중국 R&D 센터에서 프로젝트를 1년 앞당겨 완료하였습니다.

시사점

애자일 전환 과정에서 BMW 그룹은 목표를 달성하였으나, 여러 내부적 어려움과 애자일 원칙에 위배되는 행동 패턴을 드러냈으며, 이는 전환의 지속 가능성에 중요한 시사점을 제공합니다.

급격한 확장과 '강제된 자원자' 문제

ADD의 LeSS 도입 초기, 코치 계약 요구사항(70명 동시 투입)과 여름 휴가철 인력 부족으로 인해 '강제된 자원자'로 팀을 구성하였습니다. 이는 LeSS의 '자원자 활용' 원칙에 위배되며, 일부 자원자들이 새로운 작업 방식에 대한 이해 부족이나 의지 부족으로 시스템의 제약을 받았습니다.

또한, 'Deep and Narrow' 원칙에도 불구하고 내부 기대와 문제 해결 압박으로 조직을 지나치게 빠르게 확장하였습니다. 연속적 개선을 프로젝트로 간주하며 계획대로 확장

하려는 시도는 조직이 적응력 있는 그룹 대신 고착화된 조직 단위(Requirement Areas)로 전환되는 결과를 초래하였습니다.

조직 문화 및 보상 시스템과의 충돌

BMW의 기존 계층적이고 경쟁 중심의 문화는 협업 중심의 애자일 문화로 전환하는 데 어려움을 겪었습니다. 특히 다음 요소들이 애자일 원칙과 충돌하였습니다. 관리자 주도의 주관적 성과 평가와 승진 시스템이 유지되어 팀워크와 자기 관리 환경을 저해하고 개인주의를 조장하였습니다. 낮은 관리자 역량, 직원 불만 등의 문제 해결을 위해 C-4 레벨 관리자나 전문가 팀을 부활시키는 임시방편을 사용하였습니다. 이는 단기적으로 문제를 완화하였으나, 장기적으로 애자일 구조와 자기 관리 문화로의 전환을 방해하였습니다. 관리자들은 자기 관리 팀을 지원하는 대신 팀을 방치하거나, C-4 레벨 관리자를 복직시키는 등 혼란을 겪었습니다.

제품 백로그 및 기술적 부채 문제

ADD의 제품 백로그는 고객 중심 기능 대신 기술적 작업 목록으로 채워지는 경우가 많았습니다. 이는 다음과 같은 문제를 초래하였습니다. 기술적 작업 중심의 백로그는 팀의 초점을 좁은 컴포넌트에 국한시켜 전체 제품 이해와 우선순위 지정을 어렵게 하였습니다. PO는 기술적 세부 사항에 압도되어 백로그 우선순위 지정을 어려워 하였으며, 기술 관련 APO에 의존하며 권한이 약화되었습니다 LeSS 도입 전 누적된 기술적 부채와 낮은 소프트웨어 장인 정신은 지속적 통합을 저해하였으며, 팀이 공유 코드 작업을 꺼리게 되어 컴포넌트 팀으로 회귀하였습니다.

'LeSS by the book' 적용의 한계와 프레임워크 선택 문제

BMW 그룹은 애자일 전환에 '은탄'이 없으며, 조직별 맞춤 방법론이 필요함을 인식하였습니다. 도입 코치 Marcus Raitner는 LeSS의 붕괴적 이점을 선호하였으나, 'by-the-book' 방식의 LeSS 적용이 현실적으로 어려워 일부 축소되었다고 밝혔습니다. 특히, 50개 이상의 IT 시스템을 단일 제품으로 묶는 환경에서 피처 팀 개념이 적용되기 어려웠습니다. Raitner는 SAFe가 전통적 기업의 불안감을 완화할 수 있으나, 기존 역할만 이름이 바뀌는 문제를 지적하며, LeSS의 붕괴적 접근이 필요하다고 강조하였습니다. 그러나 일부 BMW 팀은 SAFe를 채택하여 PI 플래닝을 선호하였습니다.

테슬라 사례

테슬라는 디지털 기업으로 운영되며, 제품을 소프트웨어로 정의되는 디지털 장치로 간주하고 있습니다. 이러한 배경과 실리콘 밸리에 뿌리를 둔 소프트웨어 우선 접근 방식 덕분에, 테슬라는 전통적인 자동차 제조업체와 달리 소프트웨어 개발에서 흔히 사용되는 애자일 및 반복적 개발 방식(Agile and Iterative Development)을 제품 개발 및 제조 전반에 걸쳐 채택하고 있습니다.

혁신 속도와 짧은 주기

테슬라 경영진은 장기적으로 기업에게 중요한 것은 혁신의 속도(Pace of Innovation)라고 주장하며, 프로세스를 혁신 속도에 최적화하고 있습니다. 이는 문제가 감지된 시점부터 고객에게 가치로 전달되기까지 걸리는 시간을 최소화하는 것을 의미합니다.

전통적인 자동차 산업이 5~7년 주기로 모델을 변경하거나, 도요타가 약 2년의 변경 사이클을 갖는 것과 대조적으로, 테슬라의 혁신 사이클 타임(스프린트 단위)은 3시간이라고 알려져 있습니다. 테슬라의 프레몬트 공장에서는 매주 평균 20개의 하드웨어 부품이 교체되거나 개선되며, 변경된 차량이 형식 승인을 받고 즉시 출고될 수 있습니다. 이는 현재 출고되는 차량과 바로 전에 출고된 차량의 부품이 다를 수 있음을 의미합니다. 테슬라는 변화를 적용하는 데 드는 한계비용(Marginal Cost of Change)을 0에 가깝게 만듦으로써 무한한 혁신이 가능하다고 봅니다.

하드웨어에 적용된 소프트웨어 원칙

테슬라는 소프트웨어 개발의 원칙들을 하드웨어 엔지니어링 및 제조 현장(Shop Floor)으로 가져왔습니다. 소프트웨어 분야에서 자동화된 테스트를 거쳐 고객에게 즉시 배포되는 지속적 인도의 개념을 하드웨어 생산에도 적용했습니다. 이는 '유동적인 사양(liquid specifications)'으로 이어져, 같은 모델의 차량이라도 사양에 차이가 있을 수 있으며, 지속적인 제품 개선이 조립 라인에 통합됩니다. 테스트가 개발 단계 자체에 통합되는 TDD 원칙을 수용합니다. 테슬라는 '팩토리 모드(factory mode)'라는 소프트웨어 및 하드웨어 자체 테스트 프로세스를 확립하여, 파괴적이지 않은 모든 인증 테스트를 차량 스스로 자동화하여 수행합니다. 이를 통해 피드백이 며칠, 몇 달이 아닌 몇 분 안에 제공될 수 있습니다.

자율 조직화 팀 및 극단적 권한 위임

테슬라는 '두 발의 법칙(Law of Two Feet)'에 따라 엔지니어들이 가장 가치 있는 곳으로 이동하며, 해결해야 할 핵심 문제를 중심으로 팀이 모였다가 해결 후 해체되는 '번개

모임(Self-organizing Teams)' 방식으로 운영됩니다. 전 테슬라 애자일 리드였던 조 저스티스(Joe Justice)는 이를 급진적인 공유 소유권(radical shared ownership)이자 몹 작업(mob work)과 유사하다고 설명합니다. 조직은 계층이 거의 없는 평면적인 구조입니다. 직원들에게는 완전한 책임과 의사 결정 권한이 부여되며, 누구와도 소통할 수 있고, 만약 누군가 이를 막으려 한다면 즉시 해고될 수 있다는 문화가 있습니다. 설계와 생산은 동일하며, 부품 설치 팀이 곧 R&D 팀인 기능 횡단적인 통합이 이루어집니다.

시사점

테슬라의 애자일 도입은 경이로운 혁신 속도를 가능하게 했지만, 이러한 극단적인 운영 방식은 동시에 지속 가능성 및 인적 자원 관점에서 다음과 같은 이슈와 주장들을 내포하고 있습니다. 이는 광범위한 연구로 검증되었다기보다는, 테슬라의 문화를 경험한 전문가의 주장이나 관찰에 기반한 중립적인 이슈로 다루어집니다.

업무 강도와 지속 가능한 속도(Sustainable Pace)의 상충

애자일 방법론은 일반적으로 지속 가능한 속도를 중요하게 다루지만, 테슬라의 근무 환경은 극도의 강도를 요구합니다. 테슬라의 운영은 24시간 연속 가동되며, 직원들은 12시간 교대 근무를 합니다. 조 저스티스는 자신이 매우 건강함에도 불구하고 빠르게 지쳤으며, 결국 피로 누적과 개인적인 이유(아들 생일)로 퇴사하게 되었다고 밝혔습니다. 이러한 극도의 업무 강도(매일이 게임 데이)를 지속 가능하게 유지하기 위해 테슬라 직원들은 업무를 '전문 스포츠'처럼 취급해야 합니다. 이는 수면 패턴과 식단을 관리하고, 배우자나 친구의 지원 또는 생활 단순화(예: 아이들을 조부모에게 맡기거나, 잔디를 깎을 필요 없는 아파트에 거주)와 같은 강력한 지원 시스템을 갖출 것을 요구

하는 라이프스타일 선택이 됩니다.

성과에 대한 공식적인 인정 부재와 소진(Burnout) 가능성

테슬라는 혁신의 속도를 극대화하는 대신, 성과 인정(Recognition) 시스템이 결여되어 있거나 미미하다는 주장이 제기됩니다. 테슬라에서는 외부 기업에서라면 부사장급 승진이나 상을 받을 정도의 큰 문제 해결이 하루에도 여러 번 발생하지만, 실제로 이를 위한 축하 파티나 생산 중단이 없습니다. 이는 단순히 기대되는 결과이며, 직원들은 스스로를 축하해야 한다고 합니다. 조 저스티스는 테슬라처럼 성과가 매우 높은 회사에서도 애자일리스트들이 확립한 잘 연구된(research-backed) 간단한 규칙들 (예: 닷 보팅(dot voting), 칸반 유량 WIP 기능 등)이 부족하면 번아웃으로 이어질 수 있다고 지적했습니다.

테슬라는 원가 목표나 판매 목표와 같은 목표가 없고, 제품에 대한 벤치마크의 추세만 측정합니다. 그 결과, 개인이나 팀에 대한 측정 또는 관리 자체가 없습니다. 팀의 기여도는 효율성과 자본 지출 측면에서 측정되지만, 제품 목표 달성이 개인의 기여와 거의 완전히 독립적이라는 이슈가 존재합니다.

급진적 권한 위임의 실질적인 제약 및 비전형적인 팀 운영 방식

극단적인 권한 위임이 모든 문제 해결을 보장하는 것은 아니며, 팀 운영 방식에는 중립적인 이슈가 있습니다. 직원에게 수십억 달러에 접근할 권한이 주어지는 극단적인 권한 위임이 있음에도 불구하고, 조 저스티스가 퇴사할 때까지 해결하려고 노력했던 가장 어려운 구체적인 문제 중 하나는 작업 라인 근처에 화장실을 설치하는

것이었습니다. 이는 이전 토요타/GM 공장의 기술 부채(technical debt)와 인프라 제약 때문에 발생하는 현실적인 문제였습니다.

팀원들이 언제든지 동료에게 출근하지 말라고 요청할 수 있는 문화가 있습니다. 이는 매우 애자일한 실천 방식(팀이 직접 채용을 결정)이기는 하지만, 동시에 팀원 간의 신뢰나 직장 안정성 측면에서 '매우 불건전하게 들릴 수 있다'는 인식을 동반합니다.

Part 3.

AUTOMOTIVE AGILE을 위한 실용적인 이니셔티브

들어가며

자동차 산업은 지난 100년간 제조업의 정점에서 물리적 제품을 완성도 높게 생산하는 프로세스를 발전시켜 왔습니다. V-model과 Stage-Gate®는 이러한 제조 중심 사고방식의 산물로, 명확한 요구사항과 긴 개발 주기, 그리고 엄격한 품질 관리를 전제로 합니다. 이러한 프로세스들은 엔진, 섀시, 전기 시스템 등 물리적 컴포넌트를 개발하는 데 있어 탁월한 성과를 거두었고, 자동차의 안전성과 신뢰성을 보장하는 기반이 되었습니다. 그러나 소프트웨어 인텐시브 시스템이 차량의 핵심 가치를 결정하는 SDV 시대에 이르러, 이러한 전통적 프로세스는 어려움에 직면했습니다.

애자일은 바로 이러한 제조 중심 프로세스가 소프트웨어 개발에 강제됨으로써 생겨난 부작용에 대한 반동으로 등장했습니다. 소프트웨어는 본질적으로 불확실성이 높고, 요구사항이 지속적으로 변화하며, 고객이 실제 제품을 사용하기 전까지는 진정으로 원하는 것을 알 수 없습니다. 6개월에서 1년 이상 소요되는 상세 요구사항 분석과 설계 단계를 거쳐 개발에 착수하고, 개발이 거의 완료된 후에야 통합 테스트를 시작하는 폭포수 방식은 소프트웨어 프로젝트에서 반복적으로 실패를 낳았습니다.

애자일은 이러한 실패로부터 학습한 결과로, 불확실성을 인정하고, 빠른 피드백을 통해 학습하며, 고객 가치에 집중하는 철학을 제시했습니다. 짧은 반복, 지속적 통합, 고객 협력, 변화에 대한 대응 등 애자일의 핵심 원칙은 소프트웨어 개발에서 검증되었고, 실리콘밸리의 성공한 스타트업들이 이를 적극적으로 채택하면서 더욱 확산되었습니다.

그러나 애자일 역시 지난 20여 년간 다양한 실패 패턴을 겪었으며, 이는 자동차 산업이

애자일을 도입할 때 반드시 인식해야 할 교훈입니다. 첫째, 애자일 근본주의의 함정이 있습니다. "스크럼 가이드를 철저히 따르면 반드시 성공한다"거나 "XP의 모든 실천법을 도입해야만 진정한 애자일이다"와 같은 교조적 접근은 조직의 맥락과 제약을 무시하고 형식만 남기는 결과를 초래했습니다.

둘째, 자격증과 인증 중심의 형식화와 관료화입니다. CSM(Certified ScrumMaster), SAFe Agilist 등의 인증을 획득하는 것이 목적이 되고, 실제 가치 전달과 지속적 개선보다 "애자일 프로세스를 얼마나 잘 따르는가"가 평가 기준이 되는 현상이 생기기도 했습니다. 셋째, 도구 만능주의입니다. Jira나 Confluence 같은 협업 도구를 도입하면 애자일 전환이 완료되었다고 착각하거나, 비싼 ALM(Application Lifecycle Management) 솔루션을 구입하면 자동으로 민첩해질 것이라는 환상이 있습니다.

넷째, 규모 확장(Scaling)의 환상입니다. SAFe 같은 대규모 애자일 프레임워크를 무비판적으로 도입하여 오히려 복잡성과 조정 비용만 증가시키거나, 수백 명의 조직에 일률적으로 스크럼을 강제하여 혼란만 가중시키는 사례가 빈번했습니다. 다섯째, 맥락 무시입니다. 실리콘밸리의 10명짜리 스타트업이 성공한 방식을 10만 명의 글로벌 자동차 회사에 그대로 적용하려 하거나, B2C 모바일 앱 개발의 성공 사례를 안전이 최우선인 차량 임베디드 시스템 개발에 무작정 이식하려는 시도는 실패할 수밖에 없습니다.

Automotive Agile은 이러한 반성 위에서, 자동차 임베디드 시스템 개발이라는 특수한 맥락에 맞는 실용적 접근을 모색합니다. 이는 전통적 하드웨어 개발 프로세스를 전면

부정하는 것이 아니라, 소프트웨어 중심의 새로운 현실과 조화롭게 통합하는 것을 목표로 합니다. 물리적 차량 플랫폼은 여전히 긴 개발 주기와 엄격한 검증이 필요하며, V-model이나 Stage-Gate®가 더 적합할 수 있습니다. 반면 인포테인먼트, 커넥티드 서비스, 일부 ADAS 기능은 애자일하게 개발될 수 있습니다. Automotive Agile의 핵심은 "모든 것을 애자일로"가 아니라 "무엇을 어떻게 애자일하게 할 것인가"를 현명하게 선택하는 것입니다.

또한 Automotive Agile은 일반 소프트웨어 산업이 애자일을 넘어 DevSecOps, 지속적 배포(Continuous Delivery), 사이트 신뢰성 엔지니어링(SRE) 등으로 진화해 온 것처럼, 자동차 임베디드 시스템 개발도 더 나은 관행으로 발전할 수 있고 발전해야 한다는 믿음을 기반으로 합니다. 애자일 스프린트만으로는 부족합니다. 자동화된 빌드와 테스트, 지속적 통합, 보안이 개발 초기부터 통합되는 Shift Left 접근, 생산 환경(실제 차량)에서의 모니터링과 피드백 루프 등이 함께 갖춰져야 진정한 민첩성을 확보할 수 있습니다. 특히 기능안전, 사이버보안 등의 규제 준수는 더 이상 애자일을 방해하는 장애물이 아니라, 개발 프로세스에 자연스럽게 통합되어야 할 필수 요소입니다.

본 White Paper는 OEM들의 시행착오를 직시하고, 그로부터 배운 교훈을 바탕으로 실무에 적용 가능한 이니셔티브를 제시합니다. 폭스바겐 CARIAD의 실패는 권한 없는 중앙 조직의 위험성을, 메르세데스-벤츠의 사례는 기술 인프라 전환과 애자일 효과를 구분해야 할 필요성을, BMW의 경험은 조직 문화 변화 없는 프레임워크 도입의 한계를, 테슬라의 극단적 접근은 지속 가능성과 인적 비용의 균형을 각각 보여줍니다.

무엇보다 중요한 것은, Automotive SPICE®와 같은 자동차 산업의 표준 프로세스 평가 모델이 애자일과 어떻게 공존하고 통합될 수 있는가에 대한 실질적 해답을 찾는 것입니다. INTACS®는 Automotive SPICE®가 애자일 개발과 양립 가능하다고 주장하지만, 현실에서 대부분의 Automotive SPICE® 이행은 여전히 애자일 이전의 폭포수 모델, 단계별 게이트 승인, 방대한 문서 중심으로 진행되고 있습니다. 이는 Automotive SPICE® 자체의 문제라기보다는, Assessor들의 관행, OEM의 계약 조건, Tier 업체의 안전 지향적 해석이 복합적으로 작용한 결과입니다. Automotive Agile이 실제로 작동하려면 OEM, Tier, 규제 기관, Automotive SPICE® Assessor등 핵심 이해관계자들이 함께 모여 "애자일한 방식으로 Automotive SPICE®를 이행하는 것"이 무엇인지 구체적인 가이드라인과 사례를 만들어야 합니다. 이는 개별 기업이나 팀 차원에서 해결할 수 있는 문제가 아니라, 산업 전체가 협력해야 할 과제이기도 합니다.

이러한 맥락에서 본 White Paper가 제시하는 실용적 이니셔티브들은 이상적인 애자일 세계와 자동차 산업의 현실 사이의 다리를 놓으려는 시도입니다. 우리는 테슬라처럼 모든 것을 파괴하고 새로 시작할 수 없습니다. 그러나 현재의 관행을 그대로 유지하면서 SDV 시대의 경쟁에서 살아남을 수도 없습니다. Automotive Agile은 이 둘 사이의 실용적 경로를 찾는 여정입니다.

Automotive Agile의 정의와 필요성

Automotive Agile은 자동차 임베디드 시스템 개발에 특화된 애자일 접근법으로, 전통적인 제조 중심 프로세스(V-model, Stage-Gate®)와 소프트웨어 중심의 애자일 방법론을 실용적으로 통합한 개발 철학입니다. 이는 단순히 스크럼이나 칸반 같은 특정 프레임워크를 자동차 개발에 적용하는 것을 넘어서, 자동차 산업 고유의 제약 조건—

안전 규제, 하드웨어-소프트웨어 통합, OEM-Tier 공급망 구조, 장기 제품 수명 주기—을 인정하면서도 소프트웨어 인텐시브 시스템 개발의 불확실성과 빠른 변화에 효과적으로 대응하는 방법을 모색합니다.

Automotive Agile은 "애자일 선언문"의 네 가지 핵심 가치—프로세스와 도구보다 개인과 상호작용, 포괄적인 문서보다 작동하는 소프트웨어, 계약 협상보다 고객과의 협력, 계획을 따르는 것보다 변화에 대응—를 존중하되, 자동차 산업의 현실에서 이것이 의미하는 바를 재해석합니다. 예를 들어, "포괄적인 문서보다 작동하는 소프트웨어"는 ISO 26262가 요구하는 안전 문서화를 무시하는 것이 아니라, 문서 작성을 개발과 동시에 진행하고 자동화하여 부담을 줄이는 것을 의미합니다. "변화에 대응"은 물리적 차량 플랫폼의 근본적 재설계를 의미하지 않고, 소프트웨어 기능과 사용자 경험을 빠르게 반복하고 개선하는 것을 의미합니다.

하드웨어와 소프트웨어 개발 주기의 의도적 분리와 선택적 통합

자동차 개발에서 모든 것을 동일한 방식으로 다룰 수는 없습니다. 물리적 차량 플랫폼(차체, 섀시, 파워트레인)은 여전히 3~5년의 긴 개발 주기와 대규모 자본 투자를 필요로 하며, 한 번 생산이 시작되면 근본적 변경이 거의 불가능합니다. 이러한 영역에서는 전통적인 Stage-Gate® 프로세스가 여전히 유효하고 필수적입니다.

반면 소프트웨어로 구현되는 기능—인포테인먼트 시스템, 커넥티드 서비스, OTA로 업데이트 가능한 ADAS 기능, 사용자 인터페이스—은 훨씬 짧은 주기로 개발되고 반복될 수 있습니다. Automotive Agile은 이 두 세계를 명확히 구분하고, 하드웨어

플랫폼은 안정적인 "토대"로 확립한 후, 그 위에서 소프트웨어를 빠르게 진화시키는 이원화된 전략을 취합니다.

핵심은 두 세계의 인터페이스를 명확히 정의하는 것입니다. 예를 들어, ECU의 하드웨어 사양, 통신 프로토콜(CAN, Ethernet), 전력 예산 등은 초기에 안정화되어야 하며, 이를 기반으로 소프트웨어 팀은 상대적 자율성을 갖고 기능을 개발할 수 있습니다. 테슬라가 자체 중앙 컴퓨팅 플랫폼을 설계한 이유도 바로 이러한 명확한 하드웨어-소프트웨어 분리를 통해 소프트웨어 개발의 민첩성을 확보하기 위함이었습니다.

안전 규제(ISO 26262, ISO/SAE 21434) 준수와 빠른 시장 대응의 균형

자동차는 생명을 다루는 제품이며, 기능 안전과 사이버보안은 타협할 수 없는 요구사항입니다. ISO 26262는 위험 분석, 안전 요구사항 도출, 설계, 검증, 추적성 등 상세한 문서화를 요구하며, 이는 애자일의 "작동하는 소프트웨어가 포괄적 문서보다 우선"이라는 가치와 직접적으로 충돌하는 것처럼 보입니다.

Automotive Agile의 접근은 안전 활동을 스프린트 밖의 "사후 작업"으로 미루는 것이 아니라, 스프린트 내에 통합하는 것입니다. 각 스프린트의 Definition of Done에 다음과 같은 항목이 포함되어야 합니다.

- 해당 기능에 대한 HARA(Hazard Analysis and Risk Assessment) 완료
- 안전 요구사항(Safety Requirements) 도출 및 백로그 항목과의 추적성 확보
- 단위 테스트뿐만 아니라 안전 관련 테스트 케이스 실행
- 필요한 안전 문서 작성 또는 자동 생성

이를 위해서는 기능 안전 엔지니어들이 개발 팀에 내재되어야 하며, 그들은 규제 준수의 "경찰" 역할이 아니라 팀이 안전하게 빠르게 개발할 수 있도록 돕는 "조력자" 역할을 해야 합니다. 또한 요구사항 관리 도구(예: Polarion, Codebeamer, IBM DOORS)와 개발 도구(JIRA, GitLab)를 통합하여 추적성 매트릭스가 자동으로 생성되도록 하면 문서화 부담을 크게 줄일 수 있습니다.

기존 프로세스 표준와의 공존 및 점진적 전환

Automotive SPICE®는 자동차 산업에서 프로세스 성숙도를 평가하는 사실상의 표준입니다. 대부분의 OEM은 Tier 공급업체에 특정 Automotive SPICE®레벨(보통 Level 2 또는 3)을 요구하며, 이는 계약 조건의 일부입니다. Automotive SPICE®는 시스템 엔지니어링 및 소프트웨어 엔지니어링 프로세스에 무엇을 추구해야 하는지 잘 설명해 줍니다.

문제는 대부분의 Automotive SPICE® 이행이 여전히 폭포수 모델을 전제로 한다는 점입니다. 요구사항 분석, 아키텍처 설계, 상세 설계, 구현, 통합, 테스트가 순차적으로 진행되고, 각 단계의 완료는 방대한 문서로 입증되어야 합니다. Automotive SPICE® Assessor들도 이러한 전통적 접근에 익숙하며, "반복적 개발"이나 "지속적 통합"을 어떻게 평가해야 할지 명확한 가이드가 부족합니다.

그러나 Automotive SPICE®와 애자일은 본질적으로 양립 불가능한 것이 아닙니다. INTACS®는 "Automotive SPICE®는 방법론 중립적"이라고 명시하며, 폭포수든

애자일이든 사용할 수 있다고 주장합니다. 핵심은 Automotive SPICE®가 요구하는 Process Purpose와 Process Outcomes가 달성되었는지를 입증하는 것이지, 특정 문서 템플릿을 사용하거나 특정 단계적 순서를 따르는 것이 아닙니다.

Automotive Agile의 접근은 Automotive SPICE® 프로세스를 스프린트 단위로 재해석하는 것입니다. 예를 들어

- SYS.2 (시스템 요구사항 분석): 전체 요구사항을 프로젝트 초기에 완전히 확정하는 대신, 매 스프린트에서 해당 스프린트에 포함될 기능의 요구사항을 상세화합니다. 이는 백로그 정제(Backlog Refinement) 활동과 자연스럽게 연결됩니다.
- SWE.1 (소프트웨어 요구사항 분석): 사용자 스토리를 작성하고 인수 조건(Acceptance Criteria)을 정의하는 것이 소프트웨어 요구사항을 분석하는 것입니다.
- SWE.2 (소프트웨어 아키텍처 설계): 전체 아키텍처는 초기에 설정하되, 상세 설계는 스프린트마다 진화합니다. 아키텍처 결정은 ADR(Architecture Decision Records)로 문서화됩니다.
- SWE.5 (소프트웨어 통합 및 검증): 지속적 통합(CI)과 자동화된 테스트가 이를 충족합니다.

핵심은 "무엇을(What)" 입증할 것인가는 Automotive SPICE®가 정의하지만, "어떻게(How)" 입증할 것인가는 조직이 선택할 수 있다는 점입니다. 이를 위해서는 OEM, Tier, Automotive SPICE® Assessor, 도구 공급업체가 함께 모여 "애자일한 Automotive SPICE® 이행"의 구체적 사례와 가이드라인을 개발해야 합니다. 이는 단일 기업이

해결할 수 있는 문제가 아니라 산업 차원의 협력이 필요한 과제입니다.

지속적 학습과 점진적 개선의 문화

Automotive Agile의 마지막 핵심 특징은 "완성된 프로세스"가 아니라 "지속적으로 진화하는 방식"이라는 점입니다. 자동차 산업은 100년 이상의 역사를 가진 성숙한 산업이며, 오랜 기간 축적된 관행과 문화가 있습니다. 이를 일시에 변화시키는 것은 불가능하며, 시도해서도 안 됩니다.

대신 Automotive Agile은 작은 실험으로 시작하여, 학습하고, 성공한 것은 확대하며, 실패한 것은 빠르게 중단하는 실험적 접근을 권장합니다. 한두 개의 파일럿 팀에서 스크럼을 시도하고, 3개월 후 회고를 통해 무엇이 효과가 있었고 무엇이 없었는지 솔직하게 평가합니다. 조직의 맥락에 맞게 조정하고, 점진적으로 확대합니다.

이는 전통적인 "Big Bang" 변화 관리 프로그램—3년 로드맵, 전사적 교육, 일제히 전환—과는 다릅니다. 오히려 Automotive Agile의 도입 자체가 애자일해야 합니다.

왜 지금 Automotive Agile인가

현대의 자동차는 더 이상 단순한 "바퀴 달린 기계"가 아닙니다. 프리미엄 차량에는 1억 줄 이상의 소프트웨어 코드가 탑재되며, 이는 보잉 787 드림라이너(약 6,500만 줄)보다도 많습니다. 차량의 가치와 차별화 요소는 엔진 성능이나 서스펜션 튜닝에서 소프트웨어로 구현되는 기능—자율주행, 인포테인먼트, 커넥티드 서비스, OTA 업데이트

—으로 급속히 이동하고 있습니다.

테슬라는 이러한 변화를 상징하는 기업입니다. Model S를 구매한 고객은 차량을 받은 후에도 지속적으로 새로운 기능을 받습니다. 자동 주차 기능이 OTA로 추가되고, 가속 성능이 소프트웨어 업데이트로 향상되며, 사용자 인터페이스가 완전히 재설계됩니다. 차량은 스마트폰처럼 "진화하는 제품"이 되었습니다.

전통적 OEM들도 이를 인식하고 있습니다. 폭스바겐은 "e-모빌리티 제공업체"로의 전환을 선언했고, 메르세데스-벤츠는 "럭셔리 소프트웨어 회사"를 지향하며, BMW는 "디지털 기업"이 되겠다고 공표했습니다. 이러한 선언은 단순한 마케팅이 아니라, 생존을 위한 절박한 전략입니다.

SDV 시대에 소프트웨어 개발 능력은 선택이 아니라 필수입니다. 그리고 소프트웨어 개발에서 전통적인 폭포수 방식이 실패해 왔다는 것은 이미 증명된 사실입니다. Automotive Agile은 SDV 시대의 생존 전략입니다.

제조 프로세스의 난관

자동차 산업이 지난 100년간 발전시켜 온 제조 중심 프로세스는 예측 가능하고 반복 가능한 작업에 최적화되어 있습니다. 동일한 엔진을 수백만 대 생산하거나, 충돌 테스트 기준을 충족하는 차체를 설계하는 것은 물리 법칙과 공학 원리에 기반하며, 초기에 상세히 계획하고 엄격히 실행하는 것이 효과적입니다.

그러나 소프트웨어는 본질적으로 다릅니다

- 불확실성: 고객조차 자신이 진정으로 원하는 것을 실제 제품을 사용하기 전까지는 알지 못합니다. "자율주행 차량에서 운전자가 무엇을 할 수 있어야 하는가?"라는 질문에 명확한 답이 있을까요? 사용자가 실제로 기능을 경험하고 피드백을 주기 전까지는 알 수 없습니다.
- 빠른 기술 변화: 소프트웨어 기술과 표준은 프로젝트 진행 중에도 변합니다. 프로젝트를 시작할 때는 4G LTE가 표준이었지만, 3년 후 출시할 때는 5G가 대세가 될 수 있습니다. 딥러닝 알고리즘의 발전 속도는 더욱 빠릅니다. 초기에 확정한 아키텍처와 설계가 쓸모 없어질 수 있습니다.
- 시장과 경쟁 환경의 급변: 경쟁사가 혁신적인 기능을 출시하면, 고객의 기대가 순식간에 바뀝니다. 3년 전에 수집한 고객 요구사항은 출시 시점에는 이미 구식이 될 수 있습니다.

이러한 특성 때문에 소프트웨어 개발은 전형적인 Wicked Problem입니다. 문제를 완전히 이해하려면 해결책을 시도해봐야 하고, 정답이 하나로 정해져 있지 않으며, 각 시도는 되돌릴 수 없는 결과를 낳습니다. 라이트 형제가 1,000회 이상의 글라이더 비행 실험을 통해 비행의 원리를 발견한 것처럼, 소프트웨어 개발도 빠른 반복과 학습을 필요로 합니다.

Stage-Gate® 프로세스로 3년간 요구사항을 분석하고 설계한 후 개발을 시작하는 것은, 라이트 형제가 한 번의 실험으로 완벽한 비행기를 만들려 한 Samuel Langley의

접근과 같습니다. 우리는 그 결과를 알고 있습니다.

고객 기대의 변화

오늘날의 소비자는 스마트폰과 디지털 서비스에 익숙합니다. 그들은 스마트폰에서 매주 앱 업데이트를 받고, Netflix나 Spotify에서 지속적으로 개선되는 추천 알고리즘을 경험하며, Tesla에서 OTA로 새로운 기능을 받는 것을 봅니다.

이러한 경험은 자동차에 대한 기대를 변화시킵니다.

- "왜 내 차의 내비게이션은 스마트폰보다 불편한가?"
- "왜 새로운 기능을 위해 3년 후 새 차를 사야 하는가?"
- "왜 소프트웨어 버그를 고치기 위해 딜러에 방문해야 하는가?"

전통적 OEM의 5~7년 모델 사이클은 이러한 기대를 충족시킬 수 없습니다. 차량이 출시되는 순간, 이미 소프트웨어는 구식이 됩니다. 고객은 "완성된 제품"이 아니라 "지속적으로 진화하는 제품"을 원합니다. 이는 개발 방식의 근본적 전환을 요구합니다.

경쟁 환경의 변화

자동차 산업의 경쟁 구도가 변하고 있습니다. 더 이상 미국, 독일, 일본의 전통적 OEM들만의 경쟁이 아닙니다.

- 신생 전기차 업체: Tesla, Rivian, Lucid는 전통적 제약 없이 소프트웨어 중심으로 차량을 설계했습니다. 그들의 혁신 속도는 전통 OEM을 압도합니다.

- 중국 업체들의 부상: BYD, NIO, XPeng는 빠른 반복과 과감한 실험으로 무장하고 있으며, 자국 시장에서 이미 Tesla를 앞서고 있습니다.
- 빅테크의 진입 가능성: Apple의 자율주행차 프로젝트(중단되었지만), Google의 Waymo, Amazon의 Zoox 등은 소프트웨어와 데이터를 핵심 역량으로 가진 기업들입니다.

전통 OEM이 100년간 축적한 제조 노하우와 브랜드 가치는 여전히 중요합니다. 그러나 이것만으로는 충분하지 않습니다. 소프트웨어 개발 능력과 빠른 혁신 속도를 갖추지 못하면, Nokia가 스마트폰 시대에 몰락한 것처럼 역사의 뒤안길로 사라질 수 있습니다.

Automotive Agile은 전통 OEM이 자신들의 강점—제조 우수성, 품질, 안전, 공급망—을 유지하면서도 신생 업체들의 민첩성과 혁신 속도를 확보하기 위한 전략입니다.

기존 애자일 적용의 반성: 실패로부터 배우기

애자일은 만능 해결책이 아닙니다. 지난 20여 년간 수많은 조직이 애자일을 도입했지만, 많은 경우 기대했던 성과를 얻지 못했습니다. 자동차 산업이 같은 실수를 반복하지 않으려면, 일반 소프트웨어 산업에서 애자일 도입이 실패한 패턴을 이해해야 합니다.

실패 패턴 1: 근본주의의 함정

증상: "스크럼 가이드를 한 글자도 빠짐없이 따라야 한다", "XP의 12가지 실천법을 모두 도입해야만 진정한 애자일이다", "스프린트는 정확히 2주여야 한다" 같은 교조적 태도.

결과: 조직의 맥락과 제약을 무시하고 방법론의 형식만 따르게 됩니다. 팀은 "애자일을 하고 있다"는 느낌을 받지만, 실제로는 기계적으로 의식을 치르고 있을 뿐입니다. Daily Standup이 상사에게 보고하는 시간이 되고, Sprint Planning이 작업 할당 회의가 되며, Retrospective가 형식적 회의로 전락합니다.

자동차 산업에의 교훈: Automotive Agile은 특정 프레임워크를 맹목적으로 따르는 것이 아니라, 애자일의 가치와 원칙을 자동차 도메인의 맥락에 맞게 해석하고 적용하는 것입니다. "스크럼을 하는 것"이 목표가 아니라 "고객에게 더 빠르게 가치를 전달하고 변화에 효과적으로 대응하는 것"이 목표입니다.

실패 패턴 2: 형식화와 관료화

증상: CSM(Certified ScrumMaster), PSM(Professional Scrum Master), SAFe Agilist 같은 인증을 획득하는 것이 목적이 됩니다. 조직은 "우리는 Certified Agile Organization"이라고 선전하지만, 실제 업무 방식은 변하지 않습니다.

결과: 애자일이 새로운 형태의 관료제가 됩니다. "Agile Coach", "Scrum Master", "Release Train Engineer" 같은 새로운 역할들이 추가되지만, 의사결정은 여전히 느리고 계층적입니다. 오히려 복잡성과 조정 비용만 증가합니다.

자동차 산업에의 교훈: 인증과 교육은 유용할 수 있지만, 그 자체가 목적이 되어서는

안 됩니다. 중요한 것은 실제 결과—출시 주기 단축, 품질 향상, 팀 몰입도 증가—입니다. BMW가 LeSS 도입 과정에서 "by-the-book" 접근의 한계를 인식한 것은 좋은 사례입니다.

실패 패턴 3: 도구 만능주의

증상: "Jira를 도입했으니 우리는 애자일이다", "우리는 Confluence에 모든 것을 문서화하고 있다", "SAFe 도구를 구매했으니 확장은 해결되었다".

결과: 비싼 도구는 구매했지만, 사람들의 사고방식과 협업 방식은 변하지 않습니다. Jira는 또 다른 작업 추적 스프레드시트가 되고, Confluence는 또 다른 문서 저장소가 됩니다. 도구가 프로세스를 규정하게 되고, 팀은 도구에 종속됩니다.

자동차 산업에의 교훈: 도구는 사람과 프로세스를 지원해야 하며, 그 반대가 되어서는 안 됩니다. 폭스바겐이 Red Hat 기술을 활용하여 테스트 자동화를 구축한 것은 올바른 접근이지만, 도구 도입 자체가 애자일 전환을 의미하지는 않습니다. 먼저 "무엇을 달성하려 하는가"를 명확히 하고, 그것을 지원하는 도구를 선택해야 합니다.

실패 패턴 4: 규모 확장의 환상

증상: 10명짜리 팀에서 스크럼이 잘 작동했으니, 1,000명 조직에도 동일하게 적용하면 된다는 순진한 믿음. 또는 SAFe 같은 대규모 프레임워크를 도입하면 자동으로 문제가 해결될 것이라는 기대.

결과: 조직 전체에 애자일을 일률적으로 강제하면서 혼란만 가중됩니다. 수십 개의 스크럼 팀이 생기지만, 팀 간 조율이 안 되고, 의존성 관리가 악몽이 됩니다. SAFe를 도입하면 PI Planning 같은 거대한 이벤트가 추가되지만, 실제 의사결정 속도는 느려집니다.

자동차 산업에의 교훈: 폭스바겐 CARIAD가 급속히 수천 명으로 확장하면서 겪은 혼란이 좋은 교훈입니다. Automotive Agile은 작게 시작하여(예: 1~2개 파일럿 팀), 학습하고, 점진적으로 확장하는 것을 권장합니다. 또한 조직의 모든 부분이 애자일에 적합한 것은 아닙니다. 애자일이 가치를 창출할 수 있는 영역을 식별하고, 그곳에 집중해야 합니다.

실패 패턴 5: 맥락 무시 (Context Blindness)

증상: 실리콘밸리 스타트업의 성공 사례를 대기업에 그대로 적용하려는 시도. "Spotify 모델이 성공했으니 우리도 스쿼드와 트라이브를 만들자", "Netflix는 Chaos Engineering을 하니 우리도 해야 한다".

결과: 다른 조직, 다른 산업, 다른 제약 조건에서 작동한 것이 자신의 조직에서는 작동하지 않습니다. Spotify 모델은 Spotify의 특정 시점, 특정 맥락에서 작동했던 것이며, Spotify 자신도 더 이상 그대로 사용하지 않습니다.

자동차 산업에의 교훈: Tesla의 극단적 접근—24시간 가동, 매우 높은 업무 강도, 텐트에서의 생산—을 전통 OEM이 그대로 모방할 수는 없습니다. 강력한 노동조합, 공동결정제도, 장기 고용 관행을 가진 독일 OEM의 맥락은 Tesla와 완전히 다릅니다. Automotive Agile은 자동차 산업의 특수성—안전 규제, 하드웨어-소프트웨어 통합, 공급망 구조, 제품 수명 주기—을 인정하고, 그 안에서 작동하는 방법을 찾습니다.

결론: 실용주의가 답이다

이러한 실패 패턴들의 공통점은 실용주의의 부재입니다. 애자일은 도그마가 아니라 도구입니다. 특정 맥락에서 특정 문제를 해결하기 위한 접근 방식입니다. Automotive Agile의 핵심은 애자일의 가치와 원칙을 존중하되, 자동차 산업의 현실에서 실제로 작동하는 방법을 찾는 실용주의입니다.

Initiative #1: 소프트웨어와 하드웨어 개발의 의도적 분리와 선택적 통합

자동차 도메인에서 애자일 도입 경험은 모든 영역에 동일한 애자일 접근을 강제하는 것이 비현실적임을 보여줍니다. 물리적 차량 플랫폼 개발은 여전히 긴 주기와 엄격한 검증이 필요하며, 전통적 스테이지-게이트 프로세스가 더 적합합니다. 반면 인포테인먼트, 커넥티드 서비스, 그리고 일부 ADAS 기능은 애자일하게 개발될 수 있습니다. 실용적 접근은 하드웨어 플랫폼을 안정적인 기반으로 확립하고, 그 위에서 소프트웨어를 빠르게 반복하는 이원화된 전략입니다. 이를 위해서는 하드웨어와 소프트웨어 인터페이스를 명확히 정의하고, 소프트웨어 팀에게 하드웨어 플랫폼 내에서의 자율성을 보장해야 합니다.

Initiative #2: 조직 현실을 우선시하는 애자일 프레임워크 선택

BMW가 LeSS와 SAFe 사이에서 망설인 것은 각 프레임워크가 요구하는 조직 재편의 정도가 달랐기 때문입니다. 실용적 접근은 조직이 감당할 수 있는 변화의 속도를 정직하게 평가하고, 그에 맞는 프레임워크를 선택하는 것입니다. 강력한 노동조합과 공동결정제도가 있는 독일 자동차 회사들에게 급진적인 LeSS는 정치적으로 불가능할 수 있습니다. 이 경우 SAFe의 점진적 접근이 더 현실적입니다. 핵심은 프레임워크를 교조적으로 따르는 것이 아니라, 조직의 제약 조건 내에서 작동하는 최소한의 구조를 만드는 것입니다.

Initiative #3: 측정 지표의 근본적 재설계

KPI를 통해 애자일의 가치를 입증하는 사례가 있었지만, 전통적 프로젝트 관리 지표를 애자일 환경에 적용하는 것은 모순입니다. 실용적 접근은 산출물 중심 지표에서 결과 중심 지표로의 전환입니다. 얼마나 많은 기능을 개발했는가보다 고객 만족도가 얼마나 개선되었는가, 출시 후 결함률이 얼마나 감소했는가, 시장 변화에 대한 반응 시간이 얼마나 단축되었는가가 더 중요합니다. 차량의 장기 내구성과 안전성은 즉각적으로 평가될 수 없기 때문에, 단기 속도 지표와 장기 품질 지표를 균형있게 추적하는 이중 측정 시스템이 필요합니다.

Initiative #4: 애자일 도입 범위의 전략적 제한

폭스바겐과 BMW가 전사적 전환을 시도하면서 겪은 어려움은 조직의 모든 부분이 애자일에 적합한 것은 아님을 보여줍니다. 실용적 접근은 애자일이 가장 큰 가치를 창출할 수 있는 영역을 식별하고, 그곳에 자원을 집중하는 것입니다. 일반적으로 고객

접점 디지털 서비스, 차량 소프트웨어 기능, 그리고 신규 기술 프로토타이핑이 좋은 시작점입니다. 반면 재무, 인사, 법무와 같은 지원 부서에 애자일을 강제하는 것은 종종 혼란만 초래합니다. 애자일 도입의 성공은 조직 전체를 전환하는 것이 아니라, 적절한 경계를 설정하고 애자일 영역과 비애자일 영역이 효과적으로 협력하도록 만드는 것입니다.

Initiative #5: 물리적 프로토타이핑 속도의 현실적 수용

테슬라가 텐트에서 생산 라인을 구축하는 극단적 유연성을 보였지만, 이것이 품질 문제와 생산 지옥으로 이어진 것도 사실입니다. 전통적 OEM과 Tier 공급사들은 테슬라의 무모함을 모방하기보다는, 물리적 제약을 인정하면서도 반복 주기를 단축할 수 있는 현실적 방법을 찾아야 합니다. 디지털 트윈과 시뮬레이션 기술은 물리적 프로토타입 제작 전에 더 많은 반복을 가능하게 합니다. BMW가 AWS와 협력하여 가상 ECU를 개발한 것은 좋은 예입니다. 핵심은 소프트웨어처럼 빠른 반복은 불가능하지만, 전통적 방식보다는 빠른 중간 지점을 찾는 것입니다.

Initiative #6: 안전 규제와의 통합

자동차는 생명을 다루는 제품이며, ISO 26262, ISO/SAE 21434와 같은 기능 안전 표준 및 사이버보안은 상세한 문서화와 추적성을 요구합니다. 애자일의 "작동하는 소프트웨어가 포괄적 문서보다 우선"이라는 원칙은 규제 환경에서는 받아들여질 수 없습니다. 실용적 접근은 애자일 스프린트 내에 안전 활동을 통합하는 것입니다. 각 스프린트는 기능 개발뿐만 아니라 해당 기능에 대한 위험 분석, 안전 요구사항 정의, 그리고 검증 활동을 포함해야 합니다. 문서화는 스프린트가 끝난 후 일괄적으로 하는

것이 아니라, 개발과 동시에 진행되어야 합니다. 이를 위해서는 기능 안전 엔지니어들이 개발 팀에 내재되어야 하며, 그들은 규제 준수를 방해자가 아닌 가능하게 하는 역할을 해야 합니다.

Initiative #7: Automotive SPICE®와의 실용적 통합

Automotive SPICE®는 본질적으로 방법론 중립적이며, 중요한 것은 프로세스 목적과 결과를 달성하는 것이지 폭포수 방식의 순차적 단계를 강제하는 것이 아닙니다. 매 스프린트의 백로그 정제에서 해당 기능의 요구사항을 상세화하고, 사용자 스토리의 인수 조건을 소프트웨어 요구사항으로, 아키텍처 결정 기록을 설계 문서로, 지속적 통합과 자동화된 테스트를 검증 증거로 활용할 수 있습니다. 요구사항 관리 도구와 개발 도구를 통합하여 추적성 매트릭스를 자동 생성하면 문서화 부담을 크게 줄일 수 있습니다.

각 스프린트의 Definition of Done에 Automotive SPICE®가 요구하는 필수 작업 산출물—요구사항 문서화, ADR 작성, 추적성 확보, 안전 관련 활동—을 명시하면 Automotive SPICE® 준수가 개발의 일부가 됩니다. 자동화 도구를 활용하면 개발과 동시에 증거가 자동 생성되어 이중 작업을 피할 수 있습니다. 심사 직전 증거 만들기보다는 프로젝트 시작부터 Assessor와 협력하는 것이 효과적입니다. 킥오프 미팅에서 애자일 접근 방식을 설명하고, Assessor에게 스프린트 리뷰 참관 기회를 제공하며, 중간 점검으로 방향을 조정합니다.

Automotive SPICE®를 애자일을 못하는 변명으로 삼아서는 안 됩니다. 문제는 Automotive SPICE® 자체가 아니라 잘못된 해석과 전통적 관행입니다. "충분한(just

enough)" 문서화를 추구하고, 자동 생성 가능한 것은 자동화해야 합니다. Automotive SPICE®와 애자일의 통합은 개별 기업만의 문제가 아니며, OEM, Tier, 평가자, 도구 공급업체가 함께 가이드라인과 성공 사례를 개발하는 산업 차원의 협력이 필요합니다. Automotive SPICE®와 애자일은 상호 보완적입니다.

Initiative #8: DevSecOps로의 진화

Automotive Agile은 단순히 애자일 도입에 그치지 않고, 일반 소프트웨어 산업이 DevSecOps로 진화한 것처럼 더 나은 개발 관행으로 발전해야 합니다. ISO/SAE 21434가 요구하는 사이버보안은 제품 수명주기 전체에 걸친 체계적 접근이 필요하며, DevSecOps의 Shift Left 원칙—문제를 개발 초기에 발견하고 해결—을 적용해야 합니다. 사용자 스토리 작성 시 위협 모델링을 수행하고, Definition of Done에 보안 테스트를 포함시키며, 코드 커밋 시 자동화된 정적 분석으로 취약점을 스캔하여 출시 직전 보안 문제로 인한 일정 지연을 방지합니다.

CI/CD는 자동차 임베디드 시스템에도 필요합니다. 코드 커밋 시 자동 빌드, 테스트 실행, 품질 검증이 이루어지고, HIL이나 SIL 환경에서 자동 테스트를 수행하며 결과가 실시간 대시보드에 표시됩니다. OTA 업데이트가 가능한 SDV 시대에는 실제 차량에서 수집되는 데이터—오류 로그, 성능 메트릭, 사용 패턴—를 개발 팀에 피드백해서 지속적으로 개선해야 합니다. 테슬라처럼 전 세계 차량 데이터로 알고리즘을 개선하고 버그를 조기 발견하는 피드백 루프는 실험실 테스트로는 찾기 어려운 실제 문제를 발견하게 합니다.

애자일 스프린트에서 단기 목표만 추구하면 기술 부채가 누적됩니다. 각 스프린트에 리팩토링 시간을 할당하고, 코드 리뷰와 페어 프로그래밍으로 품질을 유지하며, 정적 분석으로 복잡도를 추적해야 장기적으로 지속 가능한 개발 속도를 유지할 수 있습니다. DevSecOps는 도구만으로 달성되지 않으며, 개발자가 보안과 운영을 자신의 책임으로 인식하고 실패를 학습 기회로 삼는 문화 변화가 핵심입니다. 자동차 임베디드 시스템의 실시간성, 안전, 하드웨어 의존성이라는 제약이 있지만, 산업 전체가 협력하여 자동차 특화 DevSecOps 툴체인과 모범 사례를 개발해야 합니다.

Initiative #9: 문화 변화에 대한 현실적 시간 프레임 설정

애자일은 도입하는 OEM은 문화가 애자일 전환의 가장 큰 장애물임을 발견했습니다. 실용적 접근은 문화 변화가 최소 5년에서 10년이 걸리는 장기 프로젝트임을 인정하는 것입니다. 빠른 승리를 기대하기보다는, 세대 교체와 함께 점진적으로 변화가 일어나도록 해야 합니다. 이는 신규 채용에서 애자일 마인드셋을 가진 인재를 우선시하고, 기존 직원들에게는 선택의 여지를 주는 것을 의미합니다. 애자일 팀과 전통적 팀이 공존하는 이원화된 조직 구조를 일정 기간 유지하는 것이 현실적일 수 있습니다. 핵심은 애자일을 도덕적 우월성의 문제로 만들지 않고, 특정 상황에 더 적합한 작업 방식의 선택으로 프레임하는 것입니다.

Initiative #10: 리더십 역할의 재정의

진정한 애자일 전환은 명령과 통제에서 지원과 가능하게 하기로의 리더십 스타일 변화를 요구합니다. 실용적 접근은 관리자들을 위협하기보다는 재교육하는 것입니다. 프로젝트 매니저는 스크럼 마스터나 애자일 코치로 전환될 수 있습니다. 기능 관리자는

챕터 리드나 커뮤니티 리더로 역할을 변경할 수 있습니다. 핵심은 그들이 가치 없어진다고 느끼지 않도록 하면서, 동시에 팀의 자율성을 침해하지 않도록 명확한 경계를 설정하는 것입니다.

Initiative #11: Over-the-Air 업데이트 능력의 신중한 활용

테슬라는 OTA를 통해 지속적으로 차량을 개선했지만, 동시에 충분히 검증되지 않은 소프트웨어를 배포하는 위험도 감수했습니다. 전통적 OEM들은 테슬라의 공격성을 모방하기보다는, OTA를 더 보수적으로 활용해야 합니다. 실용적 접근은 OTA 업데이트를 안전 영향도에 따라 계층화하는 것입니다. 인포테인먼트와 편의 기능은 빠르게 업데이트 할 수 있지만, 파워트레인이나 안전 시스템은 훨씬 더 엄격한 검증을 거쳐야 합니다. 또한 고객에게 업데이트 선택권을 주고, 강제 업데이트는 중대한 안전 문제에만 국한해야 합니다. OTA는 가능성을 여는 기술이지만, 그것을 어떻게 사용하는가는 브랜드 가치와 일치해야 합니다.

Initiative #12: 디지털 인재 확보와 기존 인력의 공존 전략

직원의 역량이 중요해진 시대에 디지털 인재를 확보하는 것은 필수적입니다. 그러나 실리콘밸리나 베를린에서 채용한 소프트웨어 엔지니어들과 수십 년간 회사에서 일한 전통적 엔지니어들 사이의 문화적 간극을 어떻게 메울 것인가가 과제입니다. 실용적 접근은 통합을 강제하기보다는 각 그룹의 강점을 인정하고 상호 학습을 촉진하는 것입니다. 멘토링 프로그램을 양방향으로 운영하여, 젊은 소프트웨어 엔지니어는 자동차 도메인 지식을 배우고, 베테랑 엔지니어는 새로운 개발 방법론을 배울 수 있습니다. 점진적으로 두 세계를 통합하는 경로를 계획해야 합니다.

Initiative #13: 실패에 대한 조직적 학습

테슬라의 "생산 지옥"은 실패를 통한 학습의 극단적 사례입니다. 전통적 자동차 회사들은 실패를 훨씬 덜 관대하게 다루며, 이는 혁신을 억제합니다. 실용적 접근은 통제된 실험을 위한 안전한 공간을 만드는 것입니다. 예를 들어, 신규 전기차 플랫폼이나 자율주행 기능 개발에서는 "실패할 권리"가 명시적으로 보장되어야 합니다. 실패로부터 체계적으로 학습하고, 그 교훈을 조직 전체와 공유하는 메커니즘이 필요합니다. 사후 검토는 비난이 아니라 학습의 기회가 되어야 하며, 실패를 공유한 팀을 처벌하기보다는 투명성을 보상해야 합니다.

Initiative #14: 고객 피드백 루프의 구조화

폭스바겐 User Days 프로그램은 고객을 개발 프로세스에 참여시키려는 시도였습니다. 그러나 프리미엄 브랜드에게 고객 참여는 양날의 검입니다. 너무 많이 듣는 것은 브랜드의 비전을 희석시킬 수 있고, 너무 적게 듣는 것은 시장과의 괴리를 만듭니다. 실용적 접근은 고객 피드백을 계층화하는 것입니다. 대중적 기능과 서비스에 대해서는 광범위한 고객 조사와 베타 테스트를 수행하지만, 브랜드 정체성을 정의하는 핵심 요소는 회사의 비전을 우선시합니다. 고객 피드백을 정량적 데이터와 정성적 통찰로 구분하고, 둘 사이의 균형이 중요합니다.

Initiative #15: 애자일 성과에 대한 현실적 기대 설정

자동차 도메인의 사례 발표들은 성공을 강조하지만, 실제로는 훨씬 더 복잡하고 고르지 못한 진전이 있었습니다. 실용적 접근은 이사회와 투자자들에게 애자일 전환이

단기적으로는 혼란과 생산성 저하를 가져올 수 있음을 명확히 하는 것입니다. 유명한 "J-커브" 효과는 변화 초기에 성과가 떨어졌다가 점차 회복되는 패턴을 설명합니다. 경영진이 이를 이해하고 인내심을 가질 때만 의미 있는 변화가 가능합니다. 애자일 전환을 위한 보호된 예산과 시간을 확보하고, 단기 성과 지표와 분리하여 평가하는 것이 필요합니다.

Initiative #16: 외부 컨설턴트 의존의 전략적 관리

OEM과 Tier가 다양한 컨설팅 회사들과 협력한 것처럼, 외부 전문성은 유용합니다. 그러나 컨설턴트 의존은 내부 역량 구축을 지연시킬 수 있습니다. 실용적 접근은 컨설턴트를 촉매로 사용하되, 그들이 떠난 후에도 지속될 내부 역량을 동시에 구축하는 것입니다. 이는 컨설턴트와 내부 직원의 페어링, 지식 이전을 명시적으로 계약에 포함시키기, 그리고 점진적인 자립 계획을 수립하는 것을 의미합니다. 컨설팅 프로젝트가 끝났을 때 남는 것이 멋진 발표 자료가 아니라 실제로 작동하는 프로세스와 그것을 운영할 수 있는 사람들이어야 합니다.

Initiative #17: 애자일 전환 자체를 애자일하게 수행

이는 역설적으로 들리지만 중요합니다. 많은 회사들이 애자일 전환을 거대한 변화 관리 프로그램으로 설계하고, 상세한 로드맵과 마일스톤을 정의합니다. 그러나 이것은 애자일이 해결하려는 경직성을 전환 과정 자체에 적용하는 것입니다. 실용적 접근은 작게 시작하고, 학습하며, 적응하는 것입니다. 한두 개의 파일럿 팀으로 시작하여 무엇이 작동하는지 배우고, 조직의 특수성에 맞게 조정한 후 점진적으로 확대합니다. 실패한 실험을 빠르게 중단하고, 성공한 것을 강화합니다. 3년 전환 계획보다는 3개월

학습 주기가 더 현실적입니다.

정리하며

이러한 실용적 이니셔티브들은 이상적인 애자일 세계와 자동차 산업의 현실 사이의 타협입니다. 진정한 실용주의는 변화의 필요성을 인정하면서도 조직이 감당할 수 있는 속도로 진행하고, 혁신을 추구하면서도 안전과 품질을 타협하지 않으며, 새로운 방법론을 도입하면서도 조직의 역사적 강점을 버리지 않는 균형을 찾는 것입니다. 애자일 도입의 성공은 얼마나 순수하게 스크럼 가이드를 따르는가가 아니라, 얼마나 조직의 제품과 고객에게 실질적 가치를 전달하는가로 측정되어야 합니다.

부록: 애자일 전환 실무 점검 체크리스트

도입 목적의 명확화

핵심 질문:

- 우리는 왜 애자일을 하려 하는가? (속도? 품질? 고객가치? 직원 몰입도? 혁신 문화?)
- '애자일'이 목표인가, 아니면 더 나은 가치를 위한 수단인가?

실행 포인트:

- 경영진과 실무진이 합의한 "전환 목적 선언문"을 한 문장으로 정의한다.
- 전환 성공의 기준(예: 고객 만족도 향상, 시장 출시 주기 단축, 팀 몰입도 증가)을 수치나 사례 중심으로 설정한다.

주의사항:

- "남들이 하니까", "유행이라서" 같은 이유는 실패의 전형적 징후다.
- 애자일을 도입하면 통제력을 잃는다는 오해를 해소해야 한다. (통제를 버리는 게 아니라 '가시성과 적응력'을 높이는 것임)

프레임워크 적합성 점검

핵심 질문:

- 우리 조직의 **규모·복잡도·의사결정 구조**에 적합한 프레임워크는 무엇인가?
- 리더십은 자율성과 분권적 의사결정을 받아들일 준비가 되어 있는가?

실행 포인트:

- 현재 조직의 애자일 성숙도(초기/중간/확장단계)를 진단한다.
- 각 프레임워크(SAFe, LeSS, Scrum@Scale, Spotify Model 등)의 철학, 특징, 필요 조건을 비교한다.
- 프레임워크를 '복제'하지 말고 '참조'하라. 조직 맥락에 맞게 조정, 실험할 것.

주의사항:

- 프레임워크 도입이 목표가 되면 "애자일 워싱" 위험이 높다.
- 조직의 권한 구조·보상 체계·성과 지표가 변화하지 않으면 어떤 프레임워크도 작동하지 않는다.

성과 및 측정 지표(Measurement)

핵심 질문:

- 우리는 무엇으로 성공을 판단하고 있는가?
- "문서·회의·보고"가 아니라 "작동하는 제품과 고객 피드백"으로 진척을 평가하고 있는가?

실행 포인트:

- 정량 지표: 고객 만족도(NPS), 배포 주기, 리드타임, 결함률, 팀 헬스 체크 등
- 정성 지표: 고객 인터뷰 결과, 팀 회고 피드백, 이해관계자 만족도
- KPI보다 **OKR**(Objectives and Key Results) 방식으로 목표를 관리하고, 고객 가치 중심 Key Result를 설정한다.

주의사항:

- “속도(velocity)”만으로 성과를 평가하지 않는다.
- 팀 간 비교를 피하고, 지속적 학습과 개선 중심으로 데이터를 해석한다.

기술적 토대(Technical Excellence)

핵심 질문:

- 빠르고 안전하게 변경할 수 있는 기술적 기반이 마련되어 있는가?
- 코드 품질, 자동화, 테스트, 배포 체계는 민첩성을 뒷받침하고 있는가?

실행 포인트:

- 지속적 통합(CI) / 지속적 배포(CD) 환경 구축
- 자동화 테스트, 코드 리뷰, 리팩토링 문화 정착
- 기술 부채(Technical Debt) 관리 프로세스 마련
- 아키텍처 의사결정 기록(ADR) 등 투명한 설계 공유 문화

주의사항:

- “빨리 만들기 위해 품질을 희생”하면 결국 속도가 느려진다.
- 기술 부채를 외면한 애자일은 단기적으로만 민첩하다.

회고와 학습 문화(Learning & Retrospective)

핵심 질문:

- 우리는 정기적으로 우리 방식 자체를 점검하고 개선하고 있는가?
- 회고 결과가 실제 행동 변화로 이어지고 있는가?

실행 포인트:

- 스프린트 회고(또는 주기적 리뷰)를 일정에 공식적으로 포함
- 회고에서 도출된 개선 항목(Action Item)을 추적하고, 다음 스프린트에 반영
- 실패를 ‘비난’이 아닌 ‘학습’으로 다루는 문화 조성
- 실험 기반 학습(작은 가설→실행→피드백→적응) 장려

주의사항:

- 회고가 단순한 “형식적 회의”로 전락하지 않도록 구체적 실행 계획을 세운다.
- 개선 항목의 우선순위를 정하고, 한 번에 너무 많은 항목을 시도하지 않는다.

사람과 팀 중심의 문화

핵심 질문:

- 구성원은 자율적으로 일할 수 있는 환경을 갖추고 있는가?

- 리더는 통제자가 아니라 지원자(Servant Leader)로 행동하고 있는가?

실행 포인트:

- 팀에 명확한 목표와 결정 권한을 부여
- 신뢰 기반의 피드백 문화 활성화 (예: 1:1 미팅, 팀 건강 체크)
- 개인 성과보다 팀 성과를 중시하는 보상 체계 설계
- 역할보다 역량 중심의 팀 구성 (Cross-functional Team)

주의사항:

- 자율은 방임이 아니다. 명확한 목표와 책임이 함께 있어야 한다.
- 리더의 마이크로매니지먼트는 애자일 문화를 무너뜨린다.

고객 중심(Customer Focus)

핵심 질문:

- 고객이 진정으로 원하는 것은 무엇인가?
- 우리는 정기적으로 고객 피드백을 받고 있는가?

실행 포인트:

- 제품/서비스의 가설을 고객과 함께 검증 (예: MVP, A/B 테스트)
- 고객 여정 맵(Journey Map), 페르소나, VOC 관리 시스템 활용
- 정기적 리뷰 세션에 고객 혹은 고객 대리인 참여
- 내부 이해관계자보다 실제 사용자 데이터를 우선 고려

주의사항:

- 내부 ‘요구사항 관리자’의 의견을 고객 의견으로 착각하지 않는다.
- 고객과의 협력은 문서가 아닌 **대화와 공동 탐색**을 기반으로 해야 한다.

지속 가능성과 확장(Scaling & Sustainability)

핵심 질문:

- 우리는 단기 목표가 아니라 장기적 지속 가능성을 고려하고 있는가?
- 조직 확장 시에도 애자일 문화가 유지될 수 있는가?

실행 포인트:

- 지속 가능한 속도(Sustainable Pace) 유지: 과도한 야근/번아웃 방지
- 학습·멘토링 시스템 구축 (신입·신규 팀 온보딩 체계화)
- 조직 성장 시 커뮤니케이션 구조(Tribe, Chapter, Guild 등) 설계
- 전사적 투명성 확보(공유 보드, 대시보드, 문서화 등)

주의사항:

- 단기 프로젝트 중심으로만 애자일을 운영하면 조직 수준의 학습이 쌓이지 않는다.
- 확장(Scaling)은 속도보다 **일관성·협력 문화 유지**가 핵심이다.

참고 문헌

폭스바겐 애자일 사례

https://www.volkswagen-newsroom.com/en/press-releases/volkswagen-realigns-technical-development-shorter-product-cycles-and-faster-digital-offerings-7768

https://annualreport2024.volkswagen-group.com/group-management-report/sustainable-value-enhancement/information-technology.html

https://www.usability.de/en/clients/casestudies/case-study-volkswagen-user-days.html

https://www.redhat.com/en/success-stories/the-volkswagen-group

https://arxiv.org/abs/2103.05537

https://arxiv.org/abs/1909.01624

https://www.linkedin.com/pulse/transforming-automotive-excellence-volkswagens-agile-lean-mccreery-xw6be

https://go.capgeminigroup.com/VWG-Lean-Agile-Summit-2022

https://germanautopreneur.com/p/cariad-volkswagen-software-failure-lessons

https://insideevs.com/news/724619/rivian-volkswagen-explained-cm/

https://www.forbes.com/sites/noahbarsky/2023/06/01/volkswagen-fires-tech-c-suite-after-3-avoidable-digital-strategy-errors/

https://www.electrive.com/2025/10/06/vw-reassigns-cariad-as-coordinator-of-rivian-and-xpeng-software/

https://www.heise.de/en/news/Cariad-VW-subsidiary-largely-discontinues-its-own-software-development-10712012.html

https://www.wikiwand.com/en/Cariad

https://www.medium.com/%40jsemrau/the-volkswagen-cariad-data-leak-9938ef948cec

메르세데스-벤츠 애자일 사례

https://scaledagile.com/case_study/mercedes-benz-mobility-ag/

https://www.dartai.com/blog/how-mercedes-applies-agile-in-project-management

https://www.thoughtworks.com/en-us/insights/blog/mastering-digital-transformation-case-study-part-1

https://www.mercedes-benz.com/careers/about-us/insights/it/miruja-ratnasingam.html

https://www.agilebusiness.org/resource/agile2024-the-european-experience-culture-eats-agile-for-breakfast-bmw-case-study.html

BMW 애자일 사례

https://less.works/case-studies/bmw-group

https://less.works/case-studies/bmw-group-autonomous-driving

https://info.seibert.group/display/AGILEHIVE/Scaled%2BAgile%2Bat%2BBMW%2B-%2BAn%2Binterview%2Bwith%2BMarcus%2BRaitner%2Bon%2BLeSS%2Band%2BSAFe

https://www.itpro.com/agile-development/31552/how-bmw-embraced-agile-to-hit-new-speeds

https://www.planview.com/resources/case-study/bmw-group/

테슬라 애자일 사례

https://www.agile-academy.com/en/organizational-development/agile-entwicklung-tesla-joe-justice/

https://edwardlowe13.medium.com/teslas-agile-obsession-and-how-software-principles-are-eating-car-manufacturing-d280e545d902

https://organicmedialab.com/2023/02/15/tesla-agile-pace-of-innovation/

https://castman.co.kr/ko/%EC%9E%91%EC%97%85-%ED%98%84%EC%9E%A5%EC%9D%98-%EC%95%A0%EC%9E%90%EC%9D%BC-%EB%B0%A9%EB%B2%95%EB%A1%A0-%ED%85%8C%EC%8A%AC%EB%9D%BC-%EC%83%9D%EC%82%B0-%EC%8B%9C%EC%8A%A4%ED%85%9C/

https://www.projectmanagertemplate.com/post/tesla-s-project-management-approach

https://productside.com/tesla-agile-development-product-management-at-its-best/

Disclaimer

본 White Paper는 정보 제공을 목적으로 작성된 자료로, 어떠한 경우에도 법적, 재정적, 투자적 조언을 제공하기 위한 문서가 아닙니다.

본 문서에 기재된 내용은 참고 문헌의 광범위한 조사를 토대로 작성된 것이므로, 실제의 것과 다를 수 있습니다.

회사는 본 White Paper의 내용 또는 그 사용으로 인해 발생할 수 있는 직접적, 간접적, 부수적, 결과적 손해에 대해 어떠한 법적 책임도 지지 않습니다.

또한, 본 White Paper의 어떠한 부분도 계약적 권리나 법적 의무를 발생시키지 않으며, 회사 또는 관련 기관의 공식적인 제안, 약속, 보증으로 해석되어서는 안 됩니다.

White Paper의 모든 정보는 “있는 그대로(as is)” 제공되며, 독자는 본 문서의 내용을 참고함에 있어 자신의 판단과 책임하에 행동해야 합니다.

Lean Tech. Fast Results.
SDV 시대, 빠르게 대응하는 통합형 기술 파트너
세온이앤에스

세온이앤에스는 자동차 전장 소프트웨어와 기능안전, Automotive SPICE®, 사이버보안 등 미래차 핵심 분야에 특화된 전문 기업입니다. 검증된 기술력과 컨설팅 경험을 기반으로 고객의 빠른 시장 대응을 지원하며, 공인 교육기관으로서 글로벌 표준을 이끄는 역량을 갖추고 있습니다.

끊임없는 혁신과 전문성을 바탕으로, SDV 시대의 든든한 파트너가 되겠습니다.

세온이앤에스에 대해서 더 자세히 알아 보세요!
https://www.seonens.com