기술

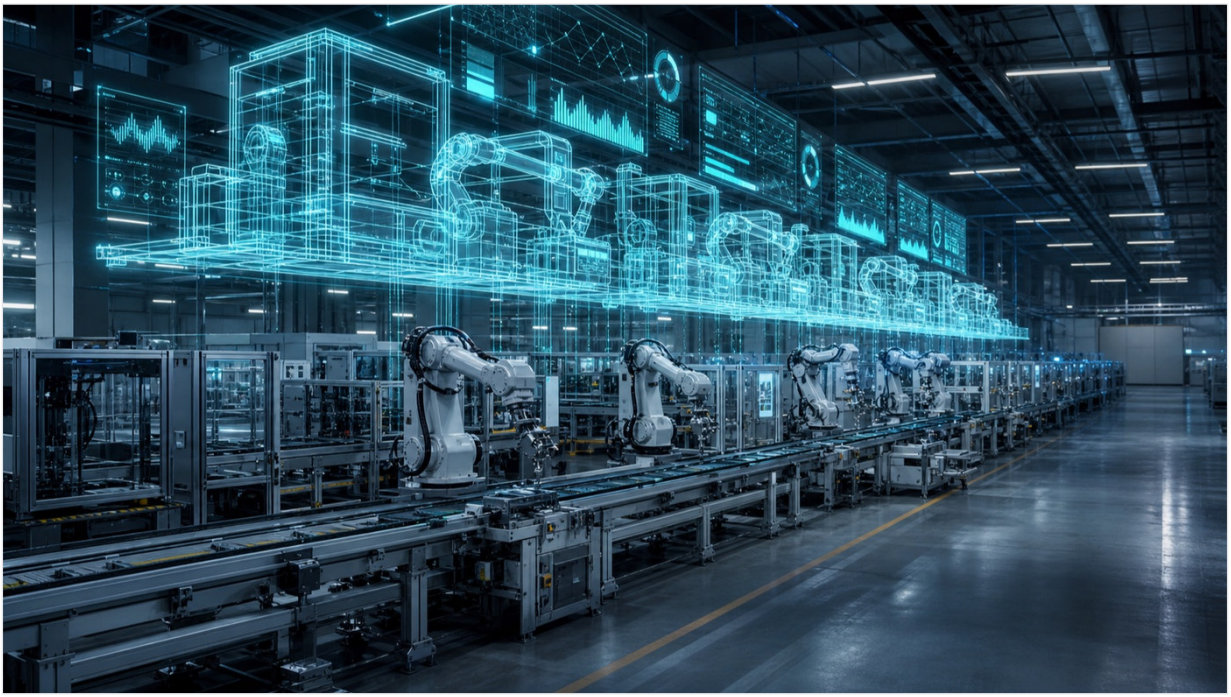
화면 속 공장은 멈춰 있는데 현장은 돌아갈 때: 디지털트윈을 진짜 쓰려면

방동걸 (주)웨이스 대표이사 2026년 6월 21일

데모를 켜면 모니터 한가운데에 공장이 통째로 떠 있습니다. 설비가 줄지어 서 있고, 조명도 그럴듯하고, 카메라를 돌리면 라인 사이사이가 다 보입니다. 처음 보는 분들은 보통 "오, 멋있다" 하고 한마디 하시죠. 그런데 현장을 좀 아는 분은 거기서 멈추지 않습니다. 화면을 한참 보다가 옆에 있는 실제 라인을 슬쩍 보고, 다시 화면을 보고는 이렇게 묻습니다. "근데 저 화면, 지금 라인이랑 똑같아요? 아니면 그냥 만들어 놓은 모형이에요?" VEXPLOR 디지털트윈 데모를 들고 다니는 웨이스가 현장에서 실제로 받은 질문입니다.

솔직히 이 질문이 디지털트윈의 전부라고 해도 과언이 아닙니다. 많은 디지털트윈 프로젝트가 딱 여기서 무너지거든요. 3D 공장 모델은 기가 막히게 잘 만들어 놓고, 그다음부터 그 모형이 실제 공장과 슬슬 어긋나기 시작합니다. 화면 속 설비는 가만히 멈춰 있는데 현장은 돌아가고, 막상 시뮬레이션을 돌려 보면 결과가 실제랑 달라서 의사결정에는 못 씁니다. 결국 비싸게 만든 3D 모형이 회의실 벽에 걸린 액자 신세가 되는 거예요.

오늘은 그 얘기를 좀 풀어 보려고 합니다. 예쁜 3D가 어쩌다 액자가 되는지, 그리고 그걸 안 되게 하려고 저희가 뭘 붙잡고 있는지. 결론부터 말하면 두 가지를 동시에 풀어야 합니다. 하나는 화면이 현실과 어긋나지 않게 실시간으로 따라붙는 것이고, 다른 하나는 그 위에서 돌리는 시뮬레이션이 "이 정도면 믿고 의사결정 하겠다" 싶은 수준으로 현실을 재현하는 것. 이 둘이 같이 돌아갈 때 비로소 디지털트윈이 액자에서 내려와 일을 하기 시작합니다.



화면 속 공장이 현장과 같은 순간을 가리켜야 트윈이라 부를 수 있습니다.

01

어긋나지 않게 따라붙기

VEXPLOR 디지털트윈은 현장과 가상 공장의 시차를 지연 500ms 이하로 맞춥니다. 사람 눈으로는 거의 실시간으로 느껴지는 수준이죠. 그런데 이게 그냥 "데이터를 빨리 쏘면 되겠지"로 풀리는 문제가 아닙니다. 수백 대 설비가 동시에 상태를 바꾸는데 그걸 끊김 없이, 그것도 가볍게 받아야 하니까요. 저희는 이걸 세 가지를 겹쳐서 풉니다.

먼저 설비 텔레메트리는 WebSocket(MQTT over WS)으로 받습니다. MQTT는 제조 IoT의 사실상 표준 프로토콜인데, 굳이 표준이 된 이유가 있어요. 수많은 설비가 쉴 새 없이 상태를 흘려보내는 상황에서, 그걸 일일이 무겁게 주고받으면 망이 견디질 못합니다. MQTT는 경량 pub/sub 구조라 이런 다대다 고빈도 상황을 낮은 오버헤드로 나릅니다. 비유하자면 모든 설비한테 일대일로 전화를 거는 대신, 다들 같은 라디오 주파수에 떠들게 하고 필요한 쪽만 그 채널을 듣는 식이에요.

그렇게 받은 메시지는 Protocol Buffers로 직렬화합니다. 사람이 읽기 편한 JSON 대신 컴팩트한 바이너리로 싸는 거죠. 메시지 한 건만 보면 차이가 미미해 보이는데, 수백 대 설비가 초 단위로 업데이트를 쏟아내는 규모에서는 이 차이가 누적돼서 대역폭과 파싱 비용을 확 줄여 줍니다. 짐을 옮길 때 박스를 헐겁게 싸느냐 빈틈없이 싸느냐의 차이라고 보시면 됩니다.

마지막 한 겹이 칼만 필터(Kalman filter)입니다. 아무리 빨리 받아도 센서 데이터에는 노이즈가 끼고, 업데이트가 때로는 띄엄띄엄 들어옵니다. 그대로 화면에 그리면 설비가 덜덜 떨거나 순간이동을 해 버려요.

칼만 필터는 노이즈를 깎아 내고, 데이터가 안 들어오는 사이의 상태를 추정해서 빈틈을 메웁니다. 그래서 데이터가 좀 끊겨도 화면은 튀지 않고 부드럽게 "지금"을 따라가는 거죠.

제가 강조하고 싶은 건 이 세 개가 따로 노는 게 아니라 한 파이프라인으로 맞물린다는 점입니다. MQTT가 가볍게 나르고, protobuf가 빈틈없이 싸고, 칼만 필터가 끊긴 자리를 메우니까, 결과적으로 화면이 "지금의 공장"을 끊김 없이 비춥니다. 한 단계만 빠져도 어딘가에서 어긋나기 시작하고, 그 순간 다시 액자로 돌아가는 거고요.

02

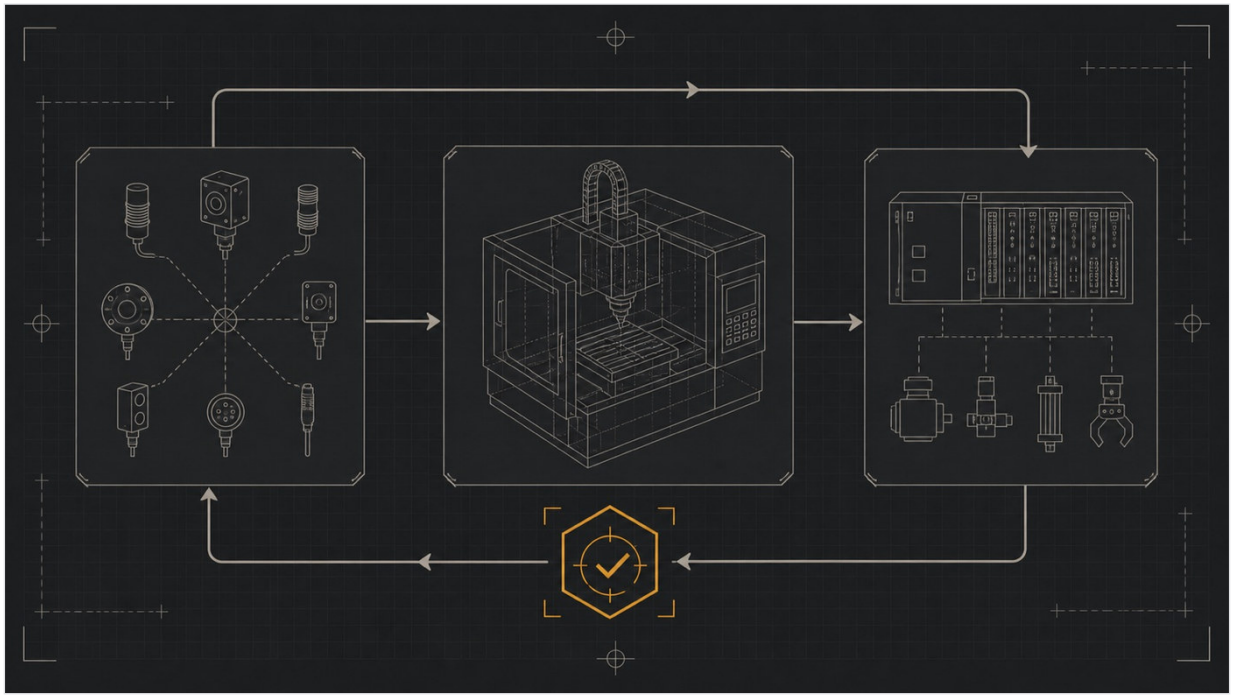
지금을 비췄으면, 이제 만약을 물어볼 차례

동기화가 "지금 어떻게 돌아가고 있나"를 비추는 일이라면, 그다음 질문은 자연스럽게 "그럼 이렇게 바꾸면 어떻게 될까"로 넘어갑니다. 새 설비를 여기에 놓으면? 공정 순서를 바꾸면? 이 만약에 답하는 게 시뮬레이션이에요. 그런데 공장이라는 무대는 한 가지 방식으로 깔끔하게 모델링되지 않습니다. 그래서 VEXPLOR는 여러 패러다임을 섞어서 갑니다.

공정의 흐름은 DES(이산사건 시뮬레이션)로 봅니다. 작업 투입, 가공 완료, 설비 고장처럼 "사건"이 벌어지는 시점을 기준으로 흐름과 대기, 병목을 짚는 방식이에요. 어디서 줄이 길어지고 어디서 막히는지를 보는 데 강합니다. 반면 설비나 AMR처럼 각자 규칙을 가지고 서로 부딪히며 움직이는 개체들의 창발적 동작은 ABS(에이전트 기반 시뮬레이션)로 모델링합니다. 흐름만 봐서는 안 잡히는, 개체들이 얽혀서 만들어 내는 상황이 여기서 드러나죠.

여기에 공간과 충돌은 물리엔진이 책임집니다. NVIDIA PhysX와 Bullet으로 충돌, 운동학 같은 물리를 계산하는데, AMR이 통로에서 마주쳤을 때 실제로 부딪히는지 비껴가는지 같은 건 흐름도나 사건 목록으로는 답이 안 나오거든요. 그리고 마지막으로 몬테카를로로 변동성을 다룹니다. 고장이 언제 날지, 사이클타임이 얼마나 흔들릴지를 한 번만 돌려 보는 게 아니라 반복해서 샘플링해, 결과를 단일 값이 아니라 분포로 봅니다. 운이 좋은 날 한 번이 아니라 "보통은 이렇고 나쁘면 이 정도"를 보는 거예요.

이쯤에서 "그냥 제일 정교한 거 하나로 통일하면 안 되나" 싶으실 수도 있습니다. 그런데 흐름에 강한 방식은 공간을 못 보고, 공간에 강한 방식은 흐름을 못 봅니다. 한 가지만으로는 "흐름도 맞고 공간도 맞는" 공장을 동시에 재현하기가 어려워요. 그래서 욕심내지 않고 하이브리드로 갑니다. 각자 제일 잘하는 자리에 두고 결과를 합치는 쪽이, 만능 하나를 쥐어짜는 것보다 현실에 가깝더라고요.



동기화는 한 방향 복사가 아니라 검증을 낀 순환입니다.

03

위험한 건 가상에서 먼저 깨먹기

여기서 한 발 더 나갑니다. 시뮬레이션으로 "이렇게 바꾸면 좋아지겠다"는 결론이 나왔다고 칩시다. 그럼 바로 라인에 적용하면 될까요? 현실은 그렇게 호락호락하지 않습니다. 새 설비 배치나 공정 변경을 실제 라인에서 바로 시험하는 건 비용도 비용이지만 위험이 큽니다. 잘못 건드리면 멀쩡히 돌던 라인이 멈출 수도 있으니까요.

그래서 VEXPLOR는 가상 시운전이라는 단계를 끼워 넣습니다. 표준 시나리오 6종을 가상에서 먼저 돌려 보고, 거기서 검증된 변경만 PLC에 적용합니다. 그리고 혹시 적용 후에 문제가 생기면 롤백 안전장치로 되돌립니다.

말로만 들으면 좀 추상적이니, 한 장면을 따라가 볼게요. 현장에서 "3호기 앞 버퍼를 줄이고 그 자리에 검사 스테이션을 하나 넣자"는 아이디어가 나왔다고 해봅시다.

먼저 이 변경을 가상 공장에 그대로 반영합니다. 그러고는 표준 시나리오 6종을 돌려요. 평소 물량, 주문이 몰리는 날, 설비 하나가 갑자기 멈추는 날 같은 식으로요. DES가 흐름이 어디서 막히는지를 보고, ABS가 AMR들이 새 검사 스테이션을 두고 어떻게 움직이는지를 보고, 물리엔진이 좁아진 통로에서 충돌이 안 나는지를 따지고, 몬테카를로가 "운 나쁜 날에는 어디까지 나빠지나"를 분포로 보여 줍니다. 여기까지 와서 결과가 괜찮으면, 그제야 그 변경 명령을 PLC로 내보냅니다. 만약 적용 후에 예상 못 한 일이 벌어지면? 롤백으로 이전 상태로 복귀하고요.

이게 바로 sim-to-real의 안전한 형태입니다. 가상에서 충분히 깨먹어 본 변경만 현실 제어로 넘기고, 언제든 되돌아올 길을 열어 두는 거죠. 사실 이건 AI 판단을 현장에 적용할 때 저희가 늘 지키는 거버넌스(권한, 승인, 롤백)와 정확히 같은 철학입니다. 똑똑한 걸 믿는 게 아니라, 틀려도 안전한 구조를 믿는 쪽이에요.

04

한 벤더에 갇히면 결국 막힌다

기술적으로 동기화도 잘 되고 시뮬레이션도 그럴듯한데, 그래도 디지털트윈이 흐지부지되는 경우가 있습니다. 한 벤더 안에 갇혀 버릴 때예요. 처음엔 잘 돌아가다가, 다른 시스템이랑 엮으려는 순간 벽에 부딪히고 확장이 막힙니다. 그럼 또 액자가 되는 거죠.

그래서 VEXPLOR는 AAS(Asset Administration Shell, IEC 63278) Submodel을 실시간으로 바인딩하고 MES와 양방향으로 연동합니다. AAS는 인더스트리 4.0의 자산 표현 표준으로, 설비와 자산을 표준 모델로 기술해서 시스템끼리 말이 통하게 해 주는 약속이에요. 쉽게 말해 우리끼리만 알아듣는 사투리로 데이터를 적어 두는 게 아니라, 누구나 알아듣는 표준어로 적어 둔다는 겁니다. 이 위에 올려 두면 디지털트윈이 한 번 보여 주고 끝나는 닫힌 데모가 아니라, 다른 시스템과 계속 주고받으며 굴러가는 운영 인프라가 됩니다.

05

그래서 실제로 됩니까

여기까지 읽고 "말은 좋은데 진짜 규모에서 돌아가긴 하느냐"는 의심이 드신다면, 그게 맞는 질문입니다. 그래서 숫자로 답하겠습니다. 한 프레스 가공 현장에서 프레스 186대, AMR 25대, EMS 6대를 동시에 동기화했고, 디지털트윈(DES) 정확도 85%를 실증 현장에서 확인했습니다. 우리 입으로 주장한 수치가 아니라 실제로 돌아가는 라인의 데이터와 대조해 나온 숫자라는 점이 핵심이에요. 모형이 예쁜 게 아니라, 현장과 어긋나지 않는다는 걸 실물 가동으로 확인했다는 뜻이니까요.

렌더링 얘기도 잠깐 보태면, 평소엔 Three.js와 WebGL2로 웹 네이티브하게 그립니다. 따로 무거운 프로그램 깔 필요 없이 브라우저에서 바로 돌아가는 거죠. 그러다 부하가 큰 시나리오는 Unity 2022 LTS와 URP, WebGL 조합으로 처리하는데, 여기에는 20 FPS SLA를 걸어 둡니다. 화려하게만 만들고 끊기면 의미가 없으니, 무거운 상황에서도 보장하는 최소 부드러움을 명시해 둔 거예요.

06

같이 풀 사람을 찾습니다

디지털트윈이 어려운 진짜 이유는, 평소엔 따로 노는 분야들이 한 곳에서 동시에 만난다는 데 있습니다. 그래픽스도 알아야 하고, 시뮬레이션도 알아야 하고, 제어와 엣지도 엮어야 하고요. 그래서 재밌는 미해결 문제가 잔뜩 남아 있어요. 수백 대를 실시간으로 동기화할 때 지연 예산을 어떻게 쪼개 쓸지, DES와 ABS와 물리엔진을 묶을 때 일관성을 어떻게 지킬지, 가상과 현실의 sim-to-real 격차를 어떻게 더 좁힐지, AAS 위에서 상호운용을 어디까지 넓힐지. 풀고 나면 보람이 꽤 쏠쏠한 것들입니다.

VEXPLOR는 제조 AI OS를 만드는 (주)웨이스의 제품입니다. 위에 늘어놓은 문제들 중에 "어 이거 한번 붙어보고 싶은데" 싶은 게 있으셨다면, 아마 잘 맞으실 거예요.

- 관심 분야: 실시간 3D(Three.js/WebGL/Unity) / 시뮬레이션(DES·ABS·물리엔진) / 제어·엣지(MQTT·PLC) / 산업 표준(AAS)
- 더 알아보기: 채용·회사 소개 https://wace.me · 기술 문의 contact@wace.me

WACE(웨이스)는 공장이 스스로 판단하고 멈추지 않게 하는 제조 AI 운영체제 VEXPLOR를 만드는 회사다.

원문	blog.vexplor.com/post/3d-digital-twin-realtime-sync
발행	2026년 6월 21일 · 방동걸 (주)웨이스 대표이사

회사	주식회사 웨이스 (WACE Inc.) · 대표이사 방동걸 · 사업자등록번호 439-87-02852
주소	(18462) 경기 화성시 동탄구 동탄영천로 150, 에이동(A동) 1702-1708호 (영천동, 실리콘앨리동탄)
연락처	전화 070-4730-0629 · contact@wace.me
홈페이지	wace.me · linkedin.com/company/wace-inc

원문 홈페이지 링크드인