Case study

Project Everest

Moving a utility-scale battery maker out of the hardware business, and shipping the first integrated system in twelve months.

Engagement series.

How lateralworks stood up a dedicated core team, fused firmware and software into a single critical path, and proved a new integrator-plus-data business model on the company's first co-developed product.

Prepared by

lateralworks
Program direction and acceleration

Sector

Grid-scale energy storage
Utility-scale battery systems

Online

lateralworks.com
Fast-time-to-market practice

Table of contents

Project Everest

Core thesis. A hardware company cannot out-price a commoditized, China-dominated supply chain. It can out-integrate it. Everest moved the client off the factory floor and onto the software and data layer, and proved the move on a single product, on time, with a dedicated team that owned one schedule end to end.

Overview

Overview

The client built utility-scale batteries, and the business under them was disappearing. Chinese manufacturers had turned cells, modules and containers into a commodity, and were driving prices down faster than any Western manufacturer could follow. The company's own next-generation program had taken more than three years to complete, its flagship customer project was carrying the risk of liquidated damages, and its engineering organization was fractured into silos with no single owner from concept to commissioning [18].

Figure 1. A utility-scale storage installation of the class Everest was built to serve: containerized battery systems co-located with generation, each unit costing on the order of half a million dollars inside multi-million-dollar projects.

lateralworks was retained to answer a hard question: could this company get out from under a losing hardware fight and still grow? The answer we built with them was to stop competing on the commodity and start competing on integration and data. Buy the hardware from the people who make it best and cheapest. Own the firmware, the software stack, and the customer-facing intelligence that turns a battery into a managed, optimized asset.

Everest was the first product built to prove that model. It is a price-competitive, roughly 5 MWh system: hardware developed in China and Korea, firmware supplied by an external development partner, and a full software stack ported and extended by the client's own software team. lateralworks stood up a dedicated core team outside the normal engineering hierarchy, put a heavyweight technical manager in charge, and drove firmware and software as one program on a single critical path. The team qualified the integrated system, carried it through certification, and released it inside the twelve-month target [18].

> **What the numbers say.** The program held to a positive schedule trend against a cost of delay the team modeled at roughly $100K in lost profit for every day of slip. The two replacement products moved ahead of the sales pipeline, and the flagship project was recovered without incurring liquidated damages [18].

Summary

The engagement at a glance

Role

Program direction and acceleration

lateralworks embedded as program leadership, not advisors: we built the team, the schedule, and the operating cadence, and ran them.

Scope

A new product and a new model

Deliver the first co-developed system on time, and prove the shift from hardware manufacturer to systems integrator and data company.

The hard part

Integration across three geographies

Fuse externally developed firmware, an internal software stack, and Asian-built hardware into one certified system, with no lost time.

What we delivered

On-time release in twelve months

A qualified, certified, price-competitive system; a repeatable core-team framework; and a working cost-of-delay business case.

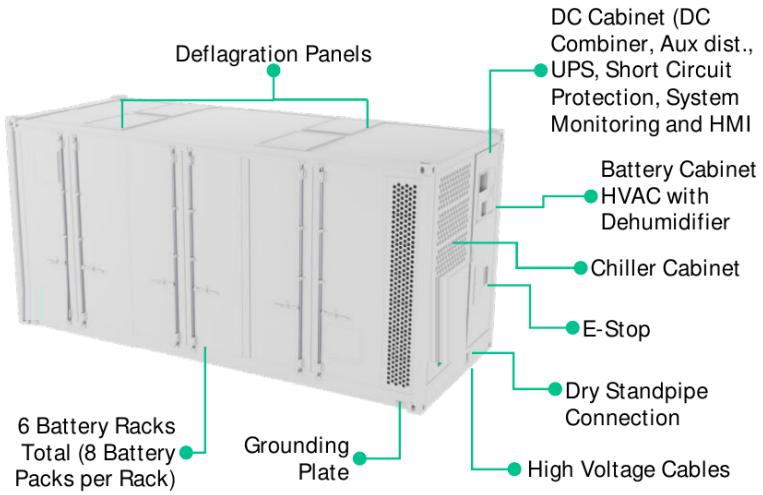


Figure 2. The Everest system architecture: six battery racks, DC combiner and controls cabinet, integrated thermal management, deflagration protection, and safety systems in a single containerized unit. Client identification has been removed for publication.

01

Section one

The market, and what was at stake

Utility-scale energy storage is one of the fastest-growing markets in the world, and one of the hardest to win. Understanding both facts is the key to understanding why Everest was built the way it was.

Grid-scale battery storage was worth roughly $10.7 billion in 2024 and is forecast to reach about $44 billion by 2030, a compound growth rate near 27 percent [2]. The physical build-out is even more dramatic: global installed capacity rose roughly twelve-fold in four years, and the United States alone commissioned about 11.9 GW of battery storage in 2024, a record fifth straight year of growth [1][4]. Solar and storage together now make up the majority of new generating capacity added to the U.S. grid [4].

Chinese scale turned the hardware into a commodity

The same growth that makes the market attractive has made the hardware brutal to sell. China accounts for close to 60 percent of global battery manufacturing capacity, and a handful of Chinese manufacturers dominate cell and system supply [1][10]. Their scale has collapsed prices. Industry surveys put lithium-ion pack prices at about $108 per kWh in 2025, down from $139 in 2023, and stationary-storage packs fell roughly 45 percent in a single year to near $70 per kWh, now the lowest-priced battery segment on the market [3]. A turnkey system that costs about $101 per kWh in China still runs well over $200 per kWh installed in the United States [3][9].

For a Western hardware manufacturer, this is an unwinnable race. When the lowest-cost producers set the price and add capacity faster than demand, the container is a commodity and margin evaporates. Tellingly, even the vertically integrated cell makers have been losing system-level share: the combined market share of integrated cell manufacturers in the stationary storage market fell from above 40 percent in 2023 to under 30 percent by the first half of 2025, as nimble integrators who buy cells rather than make them gained ground [5][6].

The value is moving to software and data

As hardware commoditizes, the margin migrates to the intelligence layer: the energy management system, the dispatch and optimization engine, the market-bidding and analytics software that let an asset owner stack revenue across several grid-service markets at once. The market for this software is projected to grow from about $3.6 billion in 2025 to roughly $16.6 billion by 2032 [7]. This is the ground a Western company can actually hold, and it is the ground Everest was designed to take.

And the systems are genuinely hard to build

None of this makes the engineering easy. A utility-scale unit packs five to six MWh of energy into a single container, which means thermal-runaway and fire propagation are first-order safety problems, not afterthoughts. Compliance with UL 9540 and the UL 9540A fire-propagation test method is effectively mandatory, referenced directly by the fire codes, and the 2025 revision tightened the requirements further [8]. Grid interconnection, long-duration operation, and multi-vendor integration all layer on top. A system that is cheap but uncertified, or certified but late, is worth nothing. Everest had to be price-competitive, safe, certified, and on time, all at once.

02

Section two

The company we walked into

Before we could build a new product, we had to be honest about the organization that would build it. lateralworks began with a four-week assessment: structured one-on-one interviews, an anonymous survey, and a document review across the product-creation process [18].

The findings were consistent, and they were not comfortable. What follows is the environment Everest had to overcome. These were not abstractions. They were the specific failures the program was set up to fix.

Silos, with no one owning the product

Interview after interview opened the same way: a highly siloed organization with little visibility from one function into the next, made worse by the distance between the U.S. teams and the engineering and manufacturing teams in Asia. Engineering verified its own designs. No one owned the product end to end, which meant no one was accountable for its success or failure in the field. When the front end of product creation is disconnected from the back end of delivery, the result is predictable: reliability, cost, and quality problems that surface at the customer site, where they are most expensive to fix [18].

No program management, and no real schedule

There was no program management structure spanning concept to customer, only weak coordination inside each function. There was no integrated master schedule, no single critical path, no gap analysis, and therefore no early warning. A basic product-improvement effort had taken more than three years, in part because no one could see the impact of scope as it expanded. When adding scope looks free, teams always take it [18].

Worse, schedules were manufactured to fit the time available. They were driven top-down, and bottom-up estimates from the people doing the work were routinely overridden by executives who "knew better." A top-down schedule disempowers the team that has to deliver it, and in our experience it always fails. It hides the gap between what the customer wants and what is actually possible, and a hidden gap surfaces late, when there is no time left to close it [18].

A culture that discounted risk

The dominant reflex was to discount risk in favor of schedule. Known risks were assumed away to save time, which is the classic signature of an organization that is chronically late. Being late forces compromises, usually less testing and less cross-functional scrutiny, and those compromises create the very rework that makes the next program late. The team had cut the basic checks and balances, system-level prototyping, engineering, design, and production validation testing, in the name of going faster, and was not going faster as a result [18].

This is the company Everest walked into. The program’s design, a dedicated team, a real schedule, a single owner, and a disciplined weekly cadence, was a direct, point-by-point answer to each of these failures.

03

Section three

Out of hardware, into systems and data

The strategic move was to stop fighting the commodity war and change the business the company was in. Instead of designing and building hardware to compete with Chinese scale, the client would procure that hardware and add the layer where value and margin actually live.

In the new model, partners deliver the container and battery packs. The client delivers the firmware, the full software stack, the integration, and the certification, and sells a complete, branded, supported system. Over time, the differentiator is not the box but the intelligence on top of it: optimizing a customer's stored energy as a managed data service. We refer to this as AI-DaaS, data-as-a-service built on the software layer rather than on the sale and installation of hardware.

A structurally better business

The economics of the two models are not close. Vertical hardware manufacturing carries high capital cost, commodity margins, and full warranty exposure on equipment the company cannot price-lead. The integrator-and-data model carries lower capital intensity, higher margins, and a path to recurring software revenue. It also decouples the company from the one race it could never win. This is consistent with what the wider market has shown: integrators who buy cells and compete on system value have been taking share from the vertically integrated manufacturers who make them [5][6].

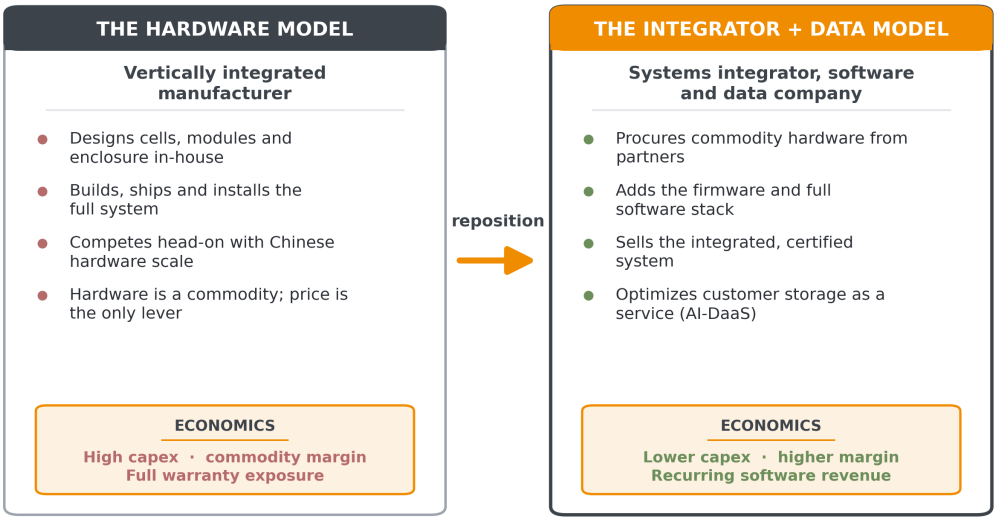


Figure 3. The strategic reposition Everest was built to prove: from a vertically integrated hardware manufacturer competing on price, to a systems integrator and data company competing on software, integration, and optimization.

This was the bet. Everest was how the company would find out whether it could actually execute the new model, on a real product, for a real customer, on a real deadline.

04

Section four

Everest: the first proof

A strategy is only as good as the first product that proves it. Everest was that product: the company’s first co-developed system, and the first test of whether the integrator model could be executed under deadline.

The mission was specific. Commission a price-competitive, roughly 5 MWh system at a first customer in North America inside the twelve-month window, then make it available worldwide. The system would replace the aging legacy platform, and it had to be delivered, installed, commissioned, and supported by the client, even though the hardware came from elsewhere [18].

Three geographies, one system

The build spanned three centers of gravity. The container and battery packs were developed and manufactured in China and Korea. The firmware was developed by an external software development partner. The client's own software team ported and further developed the full customer-facing stack on top of that firmware. lateralworks' job was to make these three streams behave as one program rather than three hand-offs.

Integration and certification were the crux. The team integrated the externally developed firmware with the internally developed software, qualified the complete system, and walked it through the UL 9540 certification process, including the re-certification work triggered whenever the firmware changed. Because certification gated first customer shipment, it sat squarely on the critical path and was managed there, not treated as a downstream formality [8][18].

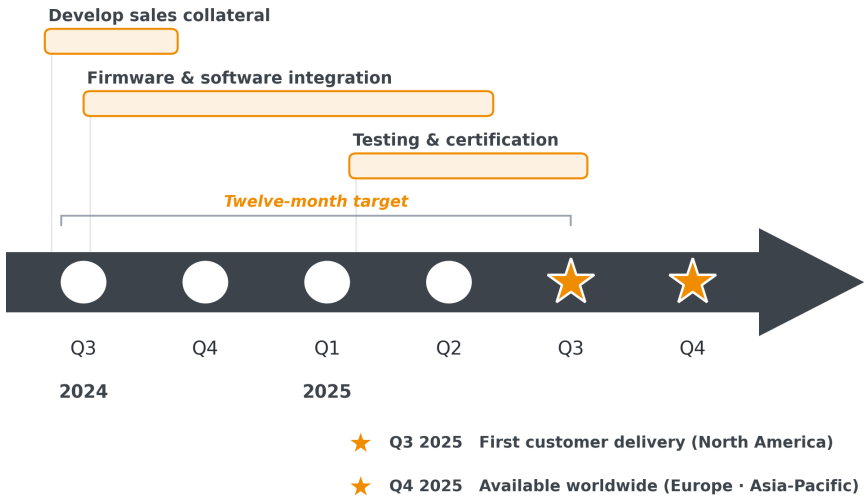


Figure 4. The Everest delivery timeline: sales collateral and firmware-and-software integration, through testing and certification, to first customer delivery in North America and worldwide availability, all inside the twelve-month target.

The product itself was engineered to the new model's logic: reduce the cost of the hardware platform, improve availability with more reliable components, make installation and service easier, and use the client's own firmware and software to deliver a consistent customer experience and industry-leading insight. Priced in the range of half a million dollars per unit inside multi-million-dollar projects, the system had to compete on total value, not on the cell price it could never win [18].

05

Section five

One team, one schedule

The organization that produced Everest looked nothing like the one described in the assessment. That was the point. lateralworks built a different structure alongside the existing hierarchy, and ran the program through it.

Fast time to market is not the product of working harder. It comes from a specific set of practices, installed together, that turn the schedule into a living instrument and the team into its owner [16][17]. Four of those practices did the heavy lifting on Everest.

A dedicated core team, outside the hierarchy

We stood up a single, dedicated, cross-functional core team and set it up outside the normal engineering reporting lines, so it could move without waiting on the silos. The team owned the outcome end to end, the accountability that had been missing across the whole company. This is the integrated core team pattern lateralworks has used across engagements since 1988: members dedicated to the program, empowered over resources and decisions, and responsible for the result [15].

A heavyweight technical leader

The program was directed by a heavyweight technical engineering manager with real depth across hardware, firmware, and software systems, not a coordinator moving status between functions. A leader who understands the whole system can make the fast cross-domain trade-offs that a committee cannot, and can read what the schedule is actually telling the team. Wheelwright and Clark named this the heavyweight team structure, and its advantage in speed and integration is well documented [12]. On Everest it meant one person could own the firmware-software-hardware boundary where most integration programs fail.

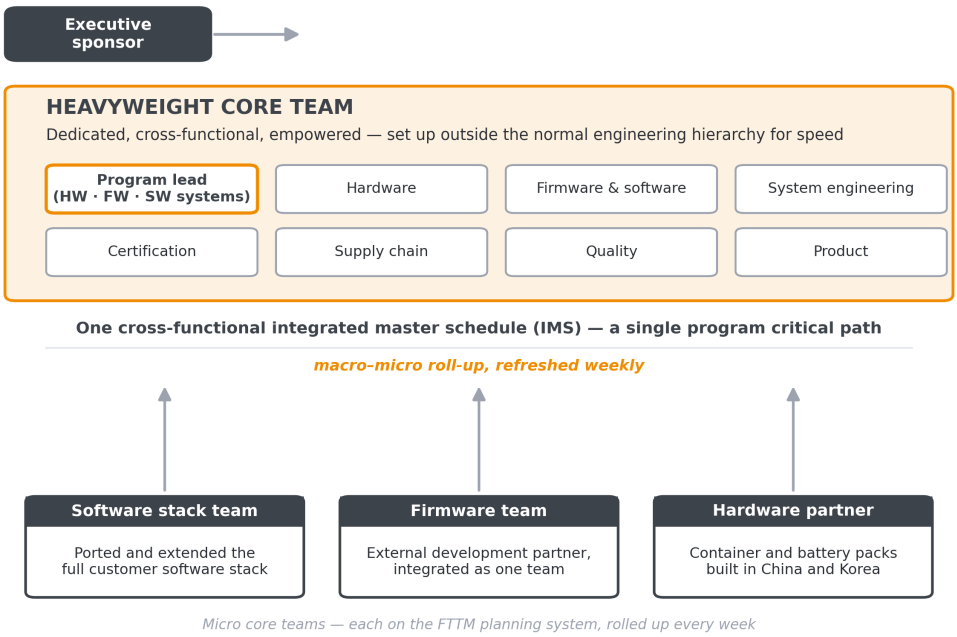


Figure 5. The Everest team structure: an executive sponsor, a heavyweight cross-functional core team set up outside the normal hierarchy, and three micro core teams whose detailed schedules rolled up weekly into a single integrated master schedule. Personnel and partner names removed for publication.

One schedule, and a weekly refresh

The schedule was the instrument that fused the team into a single program. Firmware and software were each run as their own micro core team, and both rolled up every week into one master Everest critical path. Both teams used the same FTTM planning system, so they shared one methodology, one set of scheduling practices, and one definition of done. This macro-micro roll-up is what let independent work-streams behave as one contiguous program rather than as parties to a hand-off [17][18].

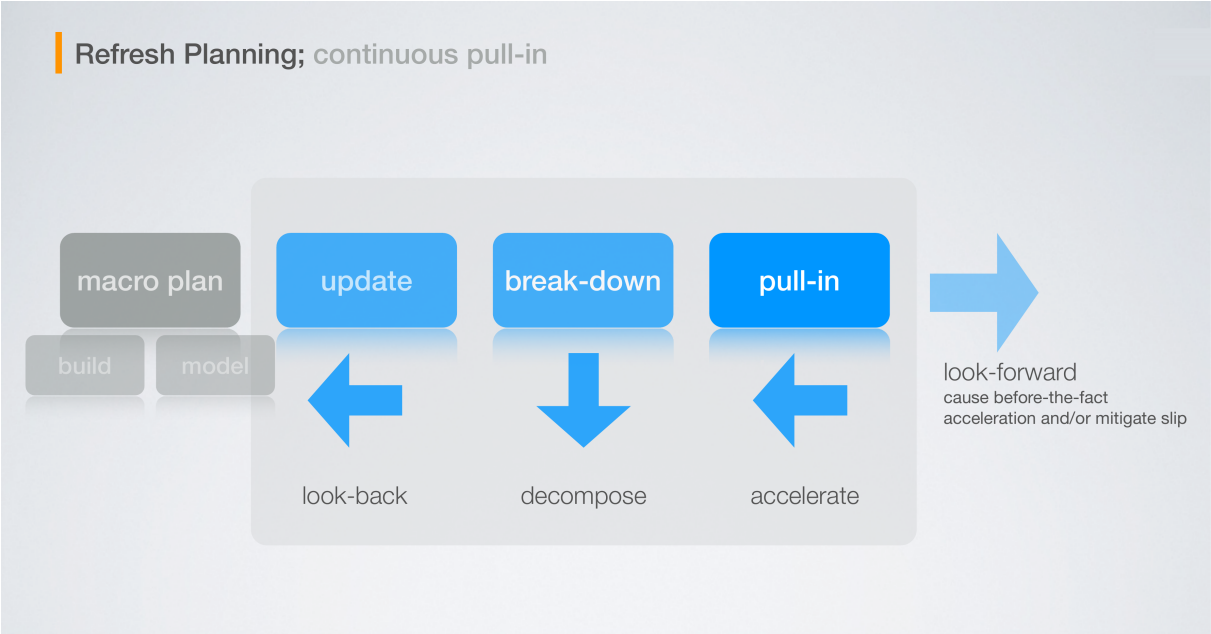


Figure 6. The weekly refresh cycle: build and model the macro plan, look back and update, decompose and break down the near-term work, then pull in and accelerate. Repeated every week, it makes acceleration the team's default rather than a crisis response.

The cadence mattered as much as the structure. Each week the team looked back to update actuals, decomposed the near-term work, and looked forward to pull the schedule in, causing problems to surface before the fact rather than after. Over time this became the team's default mindset: schedule slips were rare and not accepted, and pulling the pain forward was normal and rewarded [18].

The result is set up, not demanded

The through-line of all four practices is a single principle: fast results are engineered by how the program is provisioned, not extracted by pressure after the fact. A host that provides a dedicated, co-located, empowered team with early product definition and a cooperative partner relationship gets speed. A host that leaves resources shared and siloed, defines the product late, and treats partners transactionally gets a slow schedule, and then blames the team. Everest was deliberately set up on the left-hand side of that diagram.

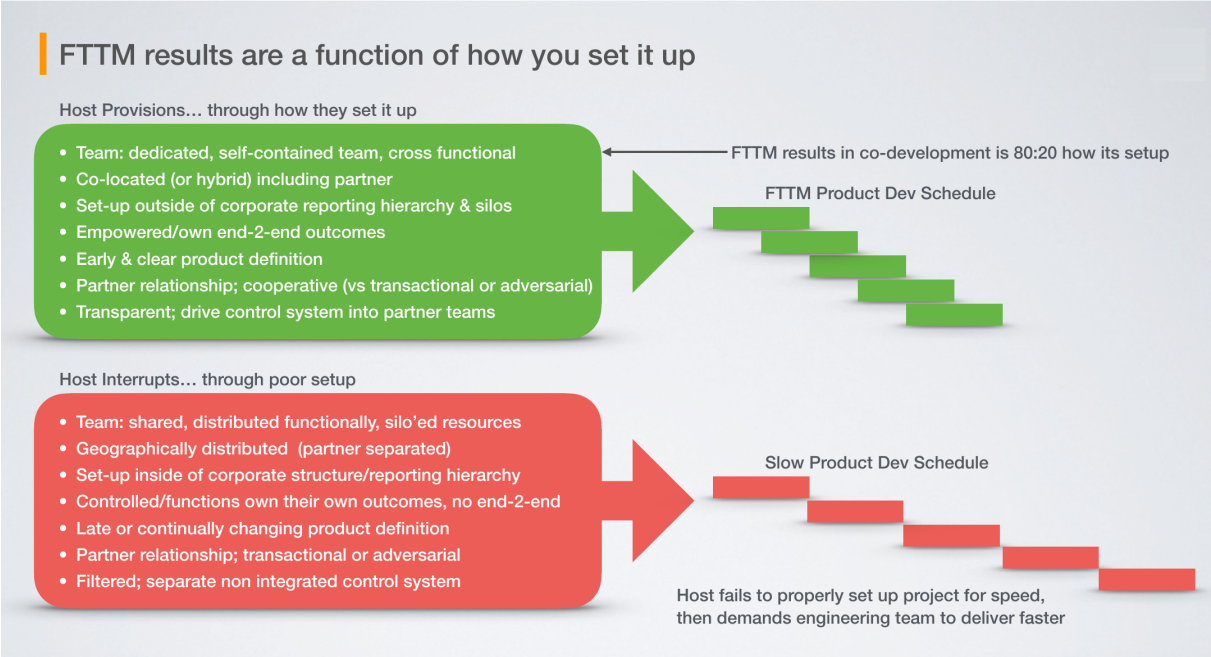


Figure 7. Why setup determines speed: the same team delivers a fast or a slow schedule depending on how the host provisions it. Everest was configured for the conditions on the left. Adapted from the lateralworks FTTM framework.

06

Section six

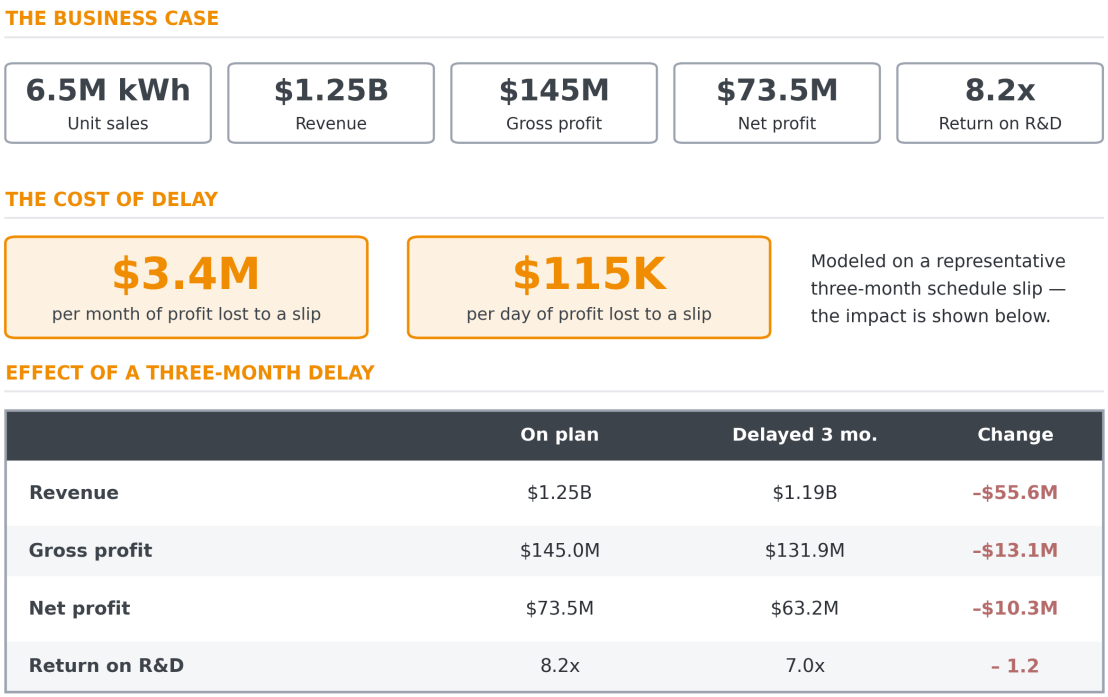
The cost of delay

If the schedule is the instrument, cost of delay is the tuning fork. It converts time into money, and it tells a team exactly how much a day is worth, which changes how the team behaves.

lateralworks built a cost-of-delay business case for Everest, modeling unit sales, pricing, and margin across the product's life against the effect of schedule slip. The idea is not new to the discipline; Reinertsen made quantified cost of delay central to product-development economics, and House and Price's break-even-time work made the same argument decades earlier [13][14]. What is uncommon is putting the number in front of a working team, every week.

About $100K a day

The model put the cost of delay for Everest at about $3.4 million a month, roughly $115K in lost profit for every day the product shipped late, a figure the team carried as approximately $100K a day in executive summaries [18]. The full business case sat behind that headline: unit sales, average selling price, and cost of goods across the product's life, netting to a program worth roughly $1.25 billion in revenue and about $73.5 million in profit at an 8.2x return on R&D.



	On plan	Delayed 3 mo.	Change
Revenue	$1.25B	$1.19B	**–$55.6M**
Gross profit	$145.0M	$131.9M	**–$13.1M**
Net profit	$73.5M	$63.2M	**–$10.3M**
Return on R&D	8.2x	7.0x	**– 1.2**

Figure 8. The Everest cost-of-delay business case: the program economics (top), the profit lost per month and per day to a slip (center), and the effect of a representative three-month delay on revenue, profit, and ROI (bottom). Based on the client cost-of-delay model; figures reflect the program's own planning assumptions.

Run the delay through that case and the abstraction disappears. A three-month slip did not read as "one quarter." On the model it read as more than $10 million of lost profit and a full point of program ROI, from 8.2x down to 7.0x [18]. When the number is that concrete, the conversation about whether to spend a week accelerating answers itself.

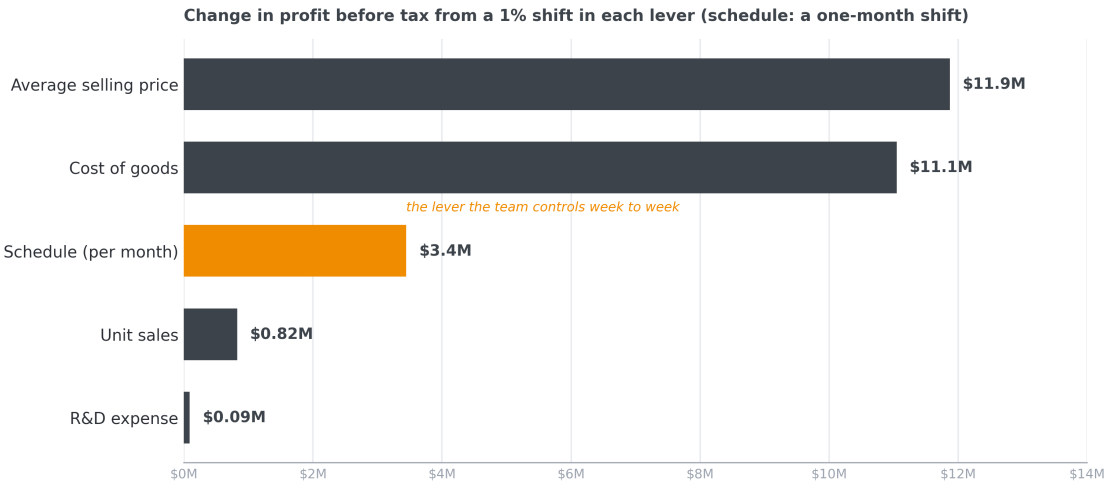


Figure 9. How much program profit moves when each lever changes by one percent (one month for schedule). Price and cost of goods move it most, but those are set by the market and the commodity supply chain. Schedule is the largest lever the team itself controls. Based on the client cost-of-delay model.

That distinction is the whole point. Price and cost of goods carry the largest sensitivity, but a Western integrator has little say over either: the market sets the price and the commodity supply chain sets the cost. Schedule is different. It is the biggest lever the team can actually move, week to week, and it is worth far more than R&D spend. That is the analytical justification for everything the core team did operationally: if a day is worth $115K, a weekly cadence built to find and remove days of slip is not process overhead. It is the highest-return activity on the program.

07

Section seven

Project Maui: autonomy with partners

Everest proved the model on one product. Project Maui was the template for scaling it: a blueprint for autonomous teams working with external development partners as the standard way to build, not the exception.

Maui framed the next-generation development question directly. Facing internal resourcing and execution constraints and a market that would not wait, the company needed new ways to develop product fast. Rather than design and engineer every component in-house, Maui evaluated co-developed and fully engineered third-party paths: partner closely to draw on outside design and engineering, or integrate an externally engineered solution, and add the client's own software value on top [18].

The pattern Everest validated

This is the same pattern Everest ran, generalized. A dedicated core team, empowered and set up outside the hierarchy, takes end-to-end ownership of a co-developed product. Hardware comes from partners chosen for scale and cost. The client concentrates its own engineering on the firmware, the software stack, and the integration that carry the margin. The team runs on one integrated schedule with a weekly acceleration cadence, and the partners are pulled inside that control system rather than managed at arm's length across a contract boundary.

Maui's importance is that it turns a single success into a repeatable operating model. Everest showed the approach works under deadline. Maui defines how the company runs every next-generation program that follows, so that autonomy, partnership, and speed become the norm rather than a one-time result [18].

Nine months, voice of customer to market-ready concept

The ambition Maui set is the clearest signal of how far the operating model can be pushed. The plan was to take a next-generation product from a blank sheet to a fully prototyped, market-ready concept in a single year: a fully specified product by March, a marketable concept by May, and a prototyped product ready for release by the end of the third quarter. From the voice-of-customer council to concept released, that is roughly nine months for a product class that had taken the company years [18].

Figure 10. The Project Maui development path: a voice-of-customer council defines what customers value, requirements align every function, and design, prototyping, and release follow in fast succession — a market-ready concept in roughly nine months.

What makes that pace credible is discipline, not shortcuts. Maui runs a rigorous, cross-functional process directed through a dedicated projects office so the program does not repeat the mistakes of the legacy platform, where scope expanded unchecked and testing was cut. It starts from the voice of the customer rather than internal assumption. And it uses third-party design and manufacturing to accelerate delivery of a high-quality, reliable product on time, exactly the division of labor that Everest proved: partners build the hardware, the core team owns the definition, the software, and the integration. Everest showed the model works under deadline; Maui is the standard that carries it forward [18].

The outcome

From losing race to working model

Stop competing on the commodity. Compete on integration and data, and prove it on one product, on time.

lateralworks program assessment

Project Everest, 2023–2025

08

Section eight
The result

The engagement began with a late, failing product and no replacement in sight. It ended with a working operating model and products trending to on-time delivery. The table below is the before-and-after in the company’s own terms [18].

Where we started, late 2023	Where it stood, early 2025
Late, failing new product with no replacement in sight	Two replacement products trending to on-time delivery, ahead of the sales pipeline
Flagship customer project at risk of significant liquidated damages	Flagship redesigned, improved, and certified with no liquidated-damage charges
Three-year-plus development cycles	New-product programs on schedule, with a management system in place
No voice-of-customer process; product group cut off from engineering	Voice-of-customer study and ongoing process informing cross-functional decisions
No cross-functional structure, weak control over the Asia team	Heavyweight technical leaders and cross-functional core teams; Asia team integrated
Internal-only development, no partner model	Partner-collaborative development integrated into one team and one result

Everest itself held a positive schedule trend through the uncertain early phase and moved toward release inside its twelve-month target. Just as important, the program left behind a framework: a dedicated core-team structure, an integrated master schedule with a weekly refresh, a working cost-of-delay model, and a group of the company's own people trained to carry the methodology forward [18].

Everest versus the generation before it

The sharpest measure of what changed is the contrast between Everest and the product that came before it. The previous generation took three years to develop and still shipped with technical deficiencies, persistent manufacturing problems, and very low margins. Everest was delivered in under twelve months as a fully certified product, ready for sale. It came in early, at a substantially higher margin, and it met every key customer requirement, requirements the team had established up front through an extensive voice-of-customer process rather than assumed from inside engineering [18].

The previous generation	Everest
Three years to develop	Under twelve months to a released product
Shipped with technical deficiencies	Fully certified and ready for sale
Persistent manufacturing problems	Met every key customer requirement
Very low margins	Substantially higher margin
Requirements assumed internally	Requirements set by an extensive VOC process
Delivered late	Delivered early

The compounding win. Speed and quality were not a trade-off. By starting from the voice of the customer, provisioning a dedicated team, and running one accelerated schedule, Everest reached market in a third of the previous generation’s time, as a certified product, at a higher margin, and hit the requirements customers actually valued. That is the difference between building fast and building the right thing fast.

09

Section nine

The practices behind the result

Everest is a clean illustration of the lateralworks FTTM framework because the program was, in effect, the framework applied end to end. It is worth naming the practices explicitly, because they are what made the result repeatable rather than lucky.

Fix the host, then the team gets fast

The assessment showed a host that was interrupting its own teams: shared resources, deep silos, top-down schedules, and late product definition. The first move was not to push the team harder but to change how it was provisioned. A dedicated team, set up outside the hierarchy with a clear owner, is a host decision, and it is the one that unlocks everything else [15][16].

Put a heavyweight in charge

A leader with genuine hardware, firmware, and software depth could own the integration boundary and make cross-domain trade-offs at speed. The heavyweight structure is not a title; it is the concentration of technical authority and end-to-end accountability in one person who can read the whole system [12].

Make the schedule the single source of truth

One integrated master schedule, with the firmware and software micro-teams rolling up weekly onto a single critical path, gave the program the lateral thread the company had never had. The same FTTM planning system across every team meant the roll-up was real, not a slide. The schedule became the instrument the team used every day to decide what to work on [17][18].

Price the time, and pull it forward

A cost-of-delay model turned time into money and gave the weekly refresh its purpose. Acceleration was continuous and before-the-fact: find the slip early, simulate the fix, validate it with the affected stakeholders, and pull the schedule in, week after week. Reinertsen and the earlier break-even-time work make the economic case; Everest is the field evidence [13][14][18].

Then make it repeatable

Finally, the win was institutionalized. Project Maui generalized the co-development pattern, and the client's own people were trained to run it. The measure of the engagement is not one product shipped on time. It is that the company can now do it again [18].

> **The practice behind the result.** Teams deliver products; the host creates the conditions under which they deliver fast. On Everest, lateralworks changed the conditions, provisioned a dedicated heavyweight team, unified the work on one schedule, and priced the cost of delay, and speed became the default. That is the whole of it.

Sources

References

[1] International Energy Agency. *Batteries and Secure Energy Transitions: Outlook for Battery Demand and Supply.* IEA, 2024. https://www.iea.org/reports/batteries-and-secure-energy-transitions

[2] Grand View Research. "Grid-Scale Battery Storage Market Size and Share Report, 2024-2030." 2024. https://www.grandviewresearch.com/industry-analysis/grid-scale-battery-storage-market

[3] BloombergNEF. "Lithium-Ion Battery Pack Prices Fall to $108 per Kilowatt-Hour." December 2025. https://about.bnef.com/insights/clean-transport/lithium-ion-battery-pack-prices-fall-to-108-per-kilowatt-hour-despite-rising-metal-prices-bloombergnef/

[4] Energy-Storage.news. "US Deployed 11.9 GW of BESS in 2024." 2025. https://www.energy-storage.news/us-deployed-11-9gw-of-bess-in-2024-18-2gw-of-grid-scale-additions-expected-in-2025/

[5] Benchmark Mineral Intelligence. "Who Were the Top Cell Suppliers in the BESS Market in H1 2025." 2025. https://source.benchmarkminerals.com/article/who-were-the-top-cell-suppliers-in-the-bess-market-in-h1-2025

[6] Energy-Storage.news. "Vertically Integrated Cell-BESS Suppliers Lose Market Share as Competition Increases." 2025. https://www.energy-storage.news/vertically-integrated-cell-bess-suppliers-lose-market-share-as-competition-increases/

[7] MarketsandMarkets. "BESS Software Market: Global Forecast to 2032." 2025. https://www.marketsandmarkets.com/Market-Reports/bess-software-market-37264532.html

[8] UL Solutions. "UL 9540A Test Method for Evaluating Thermal Runaway Fire Propagation in Battery Energy Storage Systems." ANSI/CAN/UL 9540A:2025. https://www.ul.com/services/ul-9540a-test-method

[9] Ember. "How Cheap Is Battery Storage?" 2025. https://ember-energy.org/latest-insights/how-cheap-is-battery-storage/

[10] Visual Capitalist. "China’s Dominance in Battery Manufacturing." 2025. https://elements.visualcapitalist.com/chinas-dominance-in-battery-manufacturing/

[11] National Renewable Energy Laboratory. *Cost Projections for Utility-Scale Battery Storage: 2025 Update.* NREL, 2025. https://docs.nrel.gov/docs/fy25osti/93281.pdf

[12] Wheelwright, S. C., and Clark, K. B. *Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality.* Free Press, 1992.

[13] Reinertsen, D. G. *The Principles of Product Development Flow: Second Generation Lean Product Development.* Celeritas Publishing, 2009.

[14] House, C. H., and Price, R. L. "The Return Map: Tracking Product Teams." *Harvard Business Review*, January-February 1991, pp. 92-100.

[15] lateralworks. "The Integrated Core Team." Ideas library. https://lateralworks.com/ideas/integrated-core-team

[16] lateralworks. "10 Best Practices of Fast Teams." lateralworks Papers. https://lateralworks.com/papers/10-best-practices-of-fast-teams.pdf

[17] lateralworks. "FTTM Methodology: Refresh Planning, Macro-Micro Roll-up, and Cost-of-Delay Modeling." https://lateralworks.com/methodology

[18] lateralworks. "Internal engagement assessment and program database." Project Everest engagement, 2023-2025.