Imagine hiring a Michelin-star chef to run your kitchen

**Your Executive Chef Has to Chop Carrots**

due to poor kitchen layout,

**they have to spend all day chopping vegetables** and cleaning dishes, just to get started.

**Takeaway for Execs:**  
We’re spending expensive, high-talent resources on plumbing work because the system doesn’t abstract or automate basic prep steps.



**Analogy:**  
You run a coffee shop. You have 4,000 orders for black coffee.  
But your machine forces the barista to:

* Grind beans fresh for **each cup**
* Heat water manually
* Wash and reset the machine after every order

You’ll still get great coffee — but the process is **incredibly slow and inefficient**.

**Takeaway for Execs:**  
The system wasn’t designed for speed at scale. Great for point-in-time pricing (a single espresso), not for batch analytics (catering an office of 500).

Each Date Is Like a Sealed Jar in a Pantry

**Tech Concept:** Each date of a curve (e.g., 4,000+ points from 2007 to 2025) is stored as an individual serialized Java object.

**Analogy:**  
Imagine needing to make a smoothie, but instead of grabbing a bag of frozen berries, you have 4,000 **tiny sealed jars**, each holding one berry.  
To make the smoothie (the full time series), you must:

* Find each jar
* Open it one at a time
* Scoop the contents
* Then blend them all together manually

**Takeaway for Execs:**  
MDSOR stores historical data in a way that's great for traceability, but painfully slow for assembling anything long-term. What works well for daily use becomes unmanageable at scale.

**1. We Don’t Just Need More People — We Need the Right Builders**

Hiring generalists won’t solve architectural debt. We need engineers who can rethink how MDSOR handles history, stitching, and scale.

**You Can’t Fix a Broken Kitchen by Adding More Chefs**

Our current system lacks the tools and layout for scale. Throwing more resources at it without redesign will only increase inefficiency.

**“There’s No Thread to Sew the Quilt”**

**Tech Concept:** There’s no native stitching logic — MDSOR doesn't natively connect historical data points across time.

**Analogy:**  
Imagine you’re given 4,000 beautiful quilt squares, one for each day.

But there’s no thread, no sewing machine, and no pattern.  
You’re expected to **manually sew** every piece to build a blanket.

**Takeaway for Execs:**  
The data is there, but putting it together into something usable for historical analysis is slow and fragile without engineering sup

**High-Caliber Developers Prevent the Cycle of Rework**

Right now, senior devs are stuck repeating low-leverage tasks. With the right hires, we build reusable logic instead of patching.

**High-Caliber Developers Prevent the Cycle of Rework**

Right now, senior devs are stuck repeating low-leverage tasks. With the right hires, we build reusable logic instead of patching.

**High-Caliber Developers Prevent the Cycle of Rework**

Right now, senior devs are stuck repeating low-leverage tasks. With the right hires, we build reusable logic instead of patching.

**Angles of the Issue**

**1. The System Wasn’t Built for This**

* MDSOR’s bitemporal design is a Ferrari for pricing, but a tractor for long historical time series.
* You’re trying to run 18 years of calibration on something optimized for yesterday’s curve.
* You don’t have native stitching logic, no high-speed cache for analytics.  
  **Principle:** *You can’t win a race with the wrong vehicle — build the tool for the job.*

**You’re Spending Expensive Talent on Cheap Work**

* Senior devs and SMEs are burning cycles on manual data stitching and object-by-object recon.
* That’s like paying a Michelin chef to peel potatoes all day.  
  **Principle:** *Protect your talent’s time — high leverage work only.*

**3. Ownership Without Enablement Is a Trap**

* FO is being asked to “own” both FO and Risk data without the dedicated resources, tooling, or governance Risk teams had.
* Taking ownership without fixing the system is volunteering to be blamed later.  
  **Principle:** *Don’t own the problem unless you also own the levers to solve it.*

**ital Owner Concerns = Opportunity for Leverage**

* Their surprise over SHRP risk isn’t just panic — it’s leverage to get attention and funding for fixes.
* If you act fast, you can turn a complaint into a budget conversation.  
  **Principle:** *Every complaint is a hidden buying signal for change.*

**Roadmap to Fix It (5 Moves)**

1. **Separate the Problem by Use Case**
   * Build dual-path architecture:
     + Bitemporal for Clean PnL & FO workflows
     + Flattened batch store for historical analytics & TWC calibration
2. **Form a Core Remediation Squad**
   * 3–5 high-caliber devs with write access and system knowledge
   * Dedicated to building stitching logic, caches, and self-serve tools
3. **Automate the Low-Leverage Work**
   * Stop burning SMEs on manual object inspection
   * Build validation scripts, recon dashboards, and one-click gap fixes
4. **Negotiate for Funding Now, Not Later**
   * Use Capital/Model Owner concern as leverage
   * “We can own this — but we need X budget for architecture and Y headcount for remediation”
5. **Publish a Visible TWC-to-Green Tracker**
   * One-page view: what’s blocking, who owns it, when it flips to green
   * Update weekly — builds trust and reduces noise

**Execution Filter for MDSOR/TWC**

Before committing to own TWC remediation, run it through this checklist:

* Do we have the **right people**?
* Do they have the **skills**?
* Do they have the **tools/infrastructure**?
* Do they have the **time**?
* Do they have the **authority**?

Here’s a **matrix-style comparison table** you can give your manager for quick reference:

**EMR vs MARS vs TWC – Roles, Dependencies, and Capital Impact**

| **Feature / Aspect** | **MARS** | **TWC (Time Window Calibration)** | **EMR** |
| --- | --- | --- | --- |
| **Primary Purpose** | Compute VaR for Capital | Decide which historical stress window to use for Capital calculations | Internal management risk metrics |
| **Methodology** | Delta/Gamma approximation using sensitivities and risk factor shifts | Runs same GVAR/SVAR calcs, but across 18 years of history | Various internal metrics |
| **Historical Data Needed** | Current stress window data only | **Full 18 years of history** (2007–present) for every risk factor | Depends on MDSOR data availability |
| **Regulatory Use** | ✅ Yes — Capital models | ✅ Yes — Determines Capital stress window | ❌ No (currently internal use only) |
| **Data Source** | Risk systems feeding Greeks & shifts | Requires historical time series from data platform (MDSOR if FO-owned) | MDSOR |
| **Platform Impact** | Requires accurate current sensitivities and shifts | Heavy load on platform due to massive historical runs | Runs entirely on MDSOR |
| **MDSOR Dependency** | Indirect — for FO risk factor data | **Direct — needs 18 years history per risk factor from MDSOR** | **Direct — already runs on MDSOR** |
| **Current Limitation** | None specific to MDSOR | MDSOR not optimized for large historical backtesting (performance, stitching, caching gaps) | Same limitations as TWC if expanded for Capital use |

**Manager’s takeaway:**

“MARS is already Capital-critical. TWC is equally critical because it sets the stress window, and it’s the heavy historical job. EMR uses MDSOR today — if TWC runs on MDSOR, we must invest in architecture and tooling so the platform can deliver 18 years of clean, fast, and complete history for every risk factor.”

Do you want me to also add a **dependency flow diagram** showing how MDSOR feeds EMR/TWC and how that links to MARS and Capital? That would make this even easier for him to explain visually.

Here’s a clean, **executive-friendly breakdown** you can give your manager so he quickly understands **EMR vs MARS** and the **TWC/MDSOR dependency** without technical overload:

**EMR vs MARS – and Why TWC Matters for MDSOR**

**1. MARS – What It Does Today**

* **Purpose:** Computes VaR (Value at Risk) for Capital purposes.
* **Method:** Delta/Gamma approximation.
* **Inputs Needed:**
  + Greeks/Sensitivities
  + Risk factor shifts
* **Regulatory Dependency:** Yes — Capital models rely on MARS outputs.

**2. TWC – The Capital Stress Window Engine**

* **Purpose:** Decides which historical stress window to use for Capital calculations.
  + Example: Will the stress window remain **GFC (2008)** or switch to **COVID (2020)**?
* **How It Works:**
  + Runs the **same GVAR/SVAR calculations** — but across **18 years of data** (2007–present).
  + Calibrates the entire portfolio against each historical period to see which stress scenario is most severe.
* **Regulatory Dependency:** Yes — Capital Owners depend on TWC to set the stress window.

**3. EMR – Current State**

* **Purpose:** Internal management risk metrics — **not currently used for Capital**.
* **Data Source:** MDSOR.
* **Regulatory Dependency:** No (as of now) — but **if MDSOR must run TWC**, this changes.

**4. The Dependency Problem**

Since **EMR uses MDSOR**, and **TWC requires 18 years of history for each risk factor**,  
→ **MDSOR must be able to support massive historical runs** for Capital compliance.

**Today’s Gap:**

* MDSOR is optimized for **real-time pricing** and daily PnL, not for long-span backtesting.
* Running 18 years of data for all risk factors means:
  + Heavy deserialization of thousands of Java objects.
  + Manual stitching logic.
  + Performance bottlenecks.

**5. The Takeaway for Your Manager**

* MARS → already Capital-critical.
* TWC → decides Capital stress window → runs on 18 years of history.
* EMR → internal now, but **its platform (MDSOR)** will be responsible for TWC data supply.
* **If MDSOR can’t handle 18 years efficiently, Capital deadlines & compliance are at risk.**

**Soundbite:**

“MARS runs the Capital VaR. TWC decides which stress window to use — and needs 18 years of data. EMR uses MDSOR, so if MDSOR can’t feed TWC at scale, Capital calculations are at risk.”

If you want, I can also create a **simple side-by-side visual** showing MARS, TWC, EMR, and MDSOR with arrows and dependencies so your manager can explain it in 30 seconds.  
Do you want me to do that?

Here’s a **clear matrix + short narrative** so your manager can quickly explain the **current state vs. target state** for Capital calculations and why TWC/MDSOR matters.

**Capital Calculation Dependency – Current vs. Future**

| **Aspect** | **Current State – MARS** | **Target State – EMR + TWC** |
| --- | --- | --- |
| **Capital Calculation Method** | Greeks / Delta-Gamma approximation | Full Revaluation methodology |
| **Source of VaR** | MARS outputs | EMR outputs |
| **Regulatory Dependence** | ✅ Yes – all Capital models use MARS results | 🚀 Yes – goal is to move Capital models to use EMR results |
| **TWC Dependency** | Only runs on MARS history | Must run TWC using EMR’s 18 years of history |
| **Historical Data Needed** | Current stress window history | **Full 18 years** of history (2007–present) for every risk factor |
| **Platform Dependency** | Risk systems + MARS data feeds | **MDSOR must supply 18 years of data to EMR** for TWC run |
| **Current Gap** | None specific to Capital | MDSOR not optimized for long-span backtesting (performance, stitching, caching gaps) |

**Quick Narrative for Manager**

* **Today:** All Capital calculations come from **MARS**, which uses a **Greeks/Delta-Gamma approximation**.
* **Future Goal:** Move Capital to use **EMR**, which calculates VaR using **Full Revaluation** for greater accuracy.
* **The Catch:** For EMR to feed Capital, it must also support **TWC** runs — and **TWC requires 18 years of clean history for every risk factor**.
* **Why This Matters:** MDSOR, which powers EMR, is excellent for daily pricing but **not yet architected to handle 18 years of data at scale**. Without upgrading MDSOR’s architecture and tooling, EMR cannot be reliably used for Capital.

If you want, I can **merge this with the EMR vs MARS vs TWC table** I gave earlier so your manager has a **single-page cheat sheet** that explains both the **current/future state** and the **technical gap** in one view.  
Do you want me to do that?



