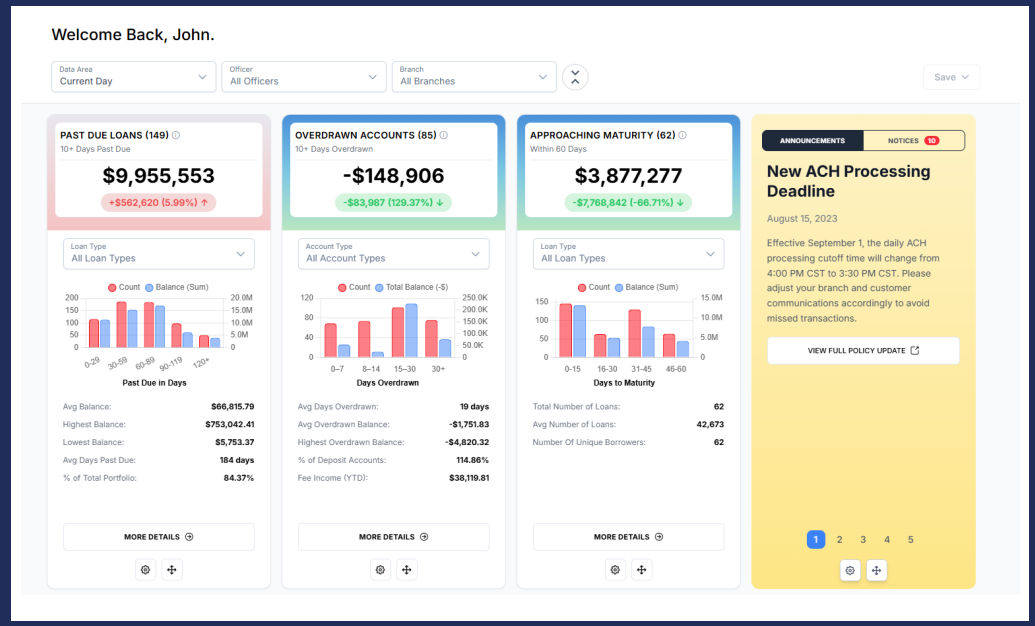
Synergex DevPartner Conference 2026

Dashboards, APIs, and the real path to a modern UI



Matt Canaday
VP, Software Development, ASI
October 28, 2026

Today's journey



01

The vision

Bringing in a UI/UX partner to design a new dashboard experience



02

The foundation

Harmony Core, OData, and the aggregate challenge



03

Making it our own

Taking development in-house and building a repeatable process



04

Results and lessons

General availability, .NET 10, and what comes next

Q&A at the end

Why we set out to modernize

Our customers deserved a modern banking experience, built on the core they already trust.

Modernize from the inside out: new experience on top, proven Synergy core underneath.



A modern look and feel

A clean, current UI that matches what users expect from today's software



Information at a glance

Dashboards that surface the most important core data the moment you log in

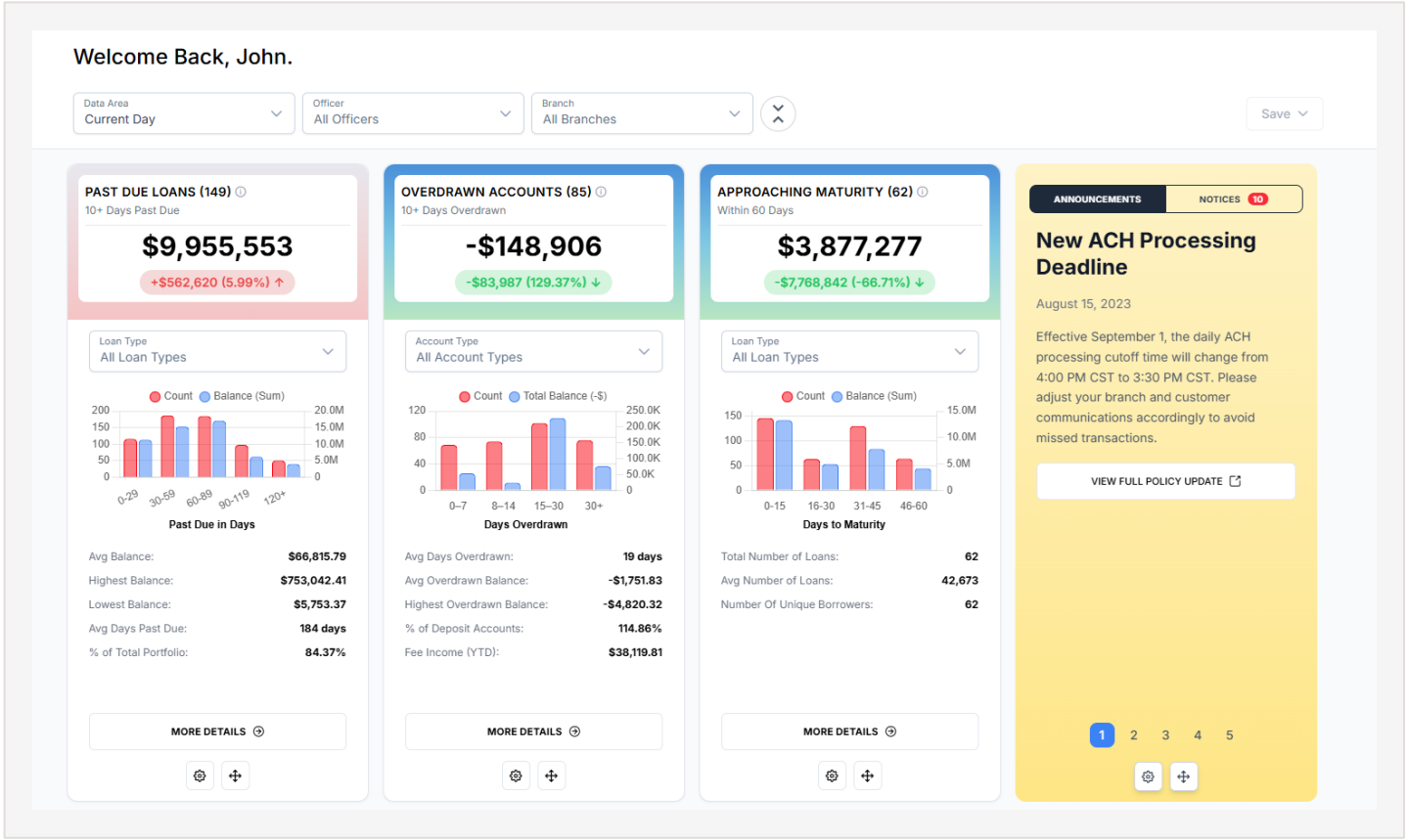


No rip and replace

Keep the core system and its data; build new layers on top of it

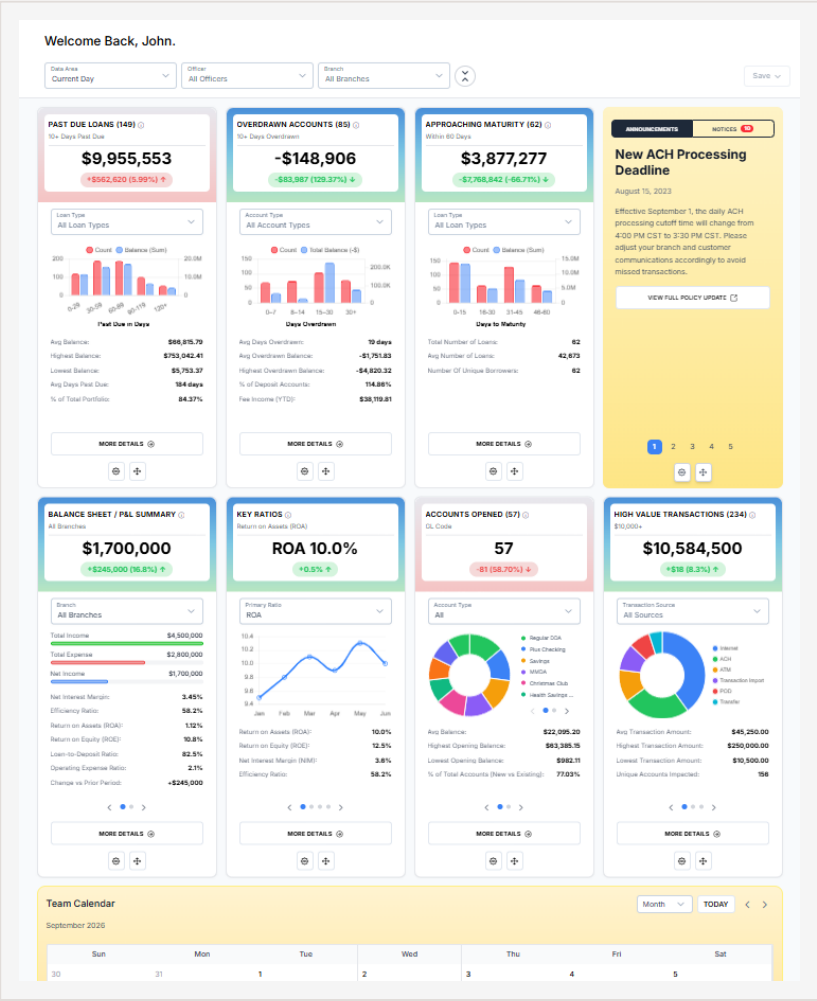
Bringing in a UI/UX partner

- Engaged a third-party UI/UX team to design the new dashboard experience
- Designed around customizable widgets, each with its own filters and details
- Focused on what bankers need first: past due loans, overdrawn accounts, approaching maturities, announcements



Original dashboard design from our UI/UX partner

The full design: a customizable canvas



Widgets in the original design

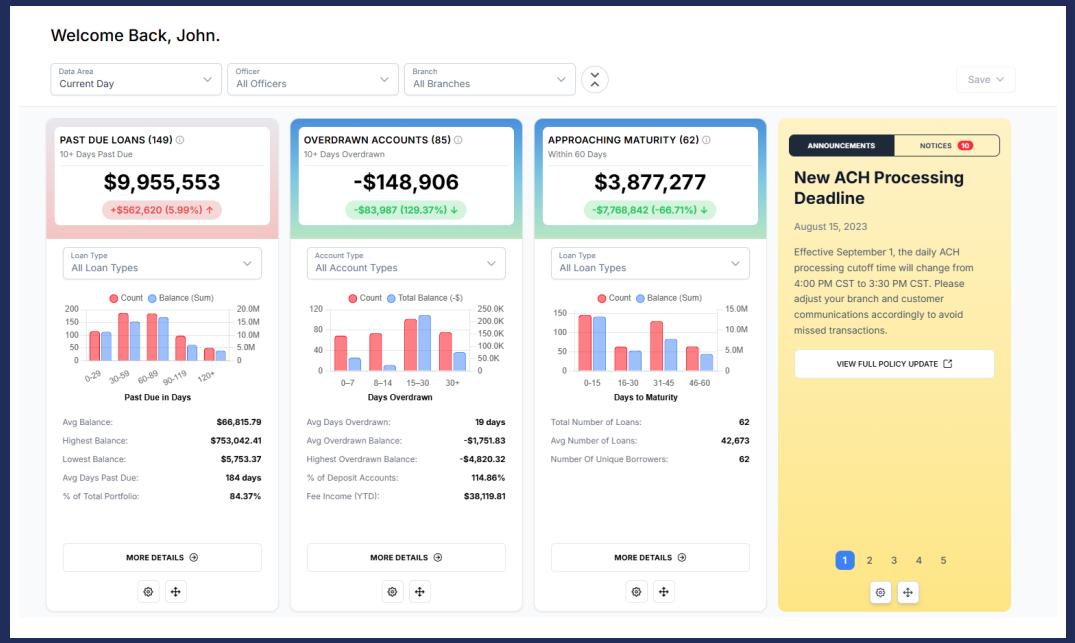
- Past due loans
- Overdrawn accounts
- Approaching maturity
- Announcements and notices
- Balance sheet / P&L summary
- Key ratios
- Accounts opened
- High value transactions
- Team calendar

Every widget filters by data area, officer, and branch, and can be configured and rearranged.

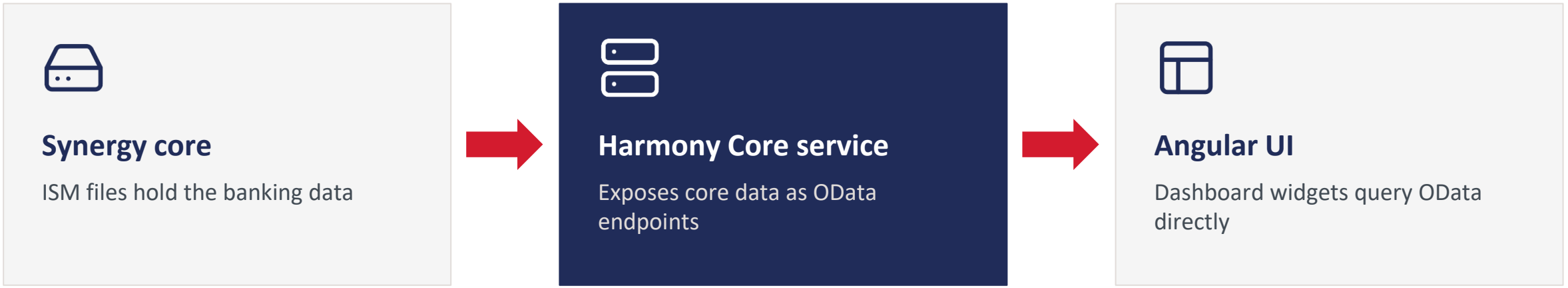
ASI annual conference · October 2025

We showed customers the design. It was very well received.

“What used to take 20+ clicks can now be done in 1-2 clicks"



Starting with Harmony Core and OData



The original ask

OData going straight from Harmony Core to the Angular UI. Standard, queryable, and fast to stand up, with no custom API layer in between.

The challenge: every widget is an aggregate

Dashboards don't show records. They show counts, sums, averages, and buckets.



What a widget needs

- Count of past due loans by aging bucket
- Total balance per bucket
- Average, highest, and lowest balance
- Percent of total portfolio

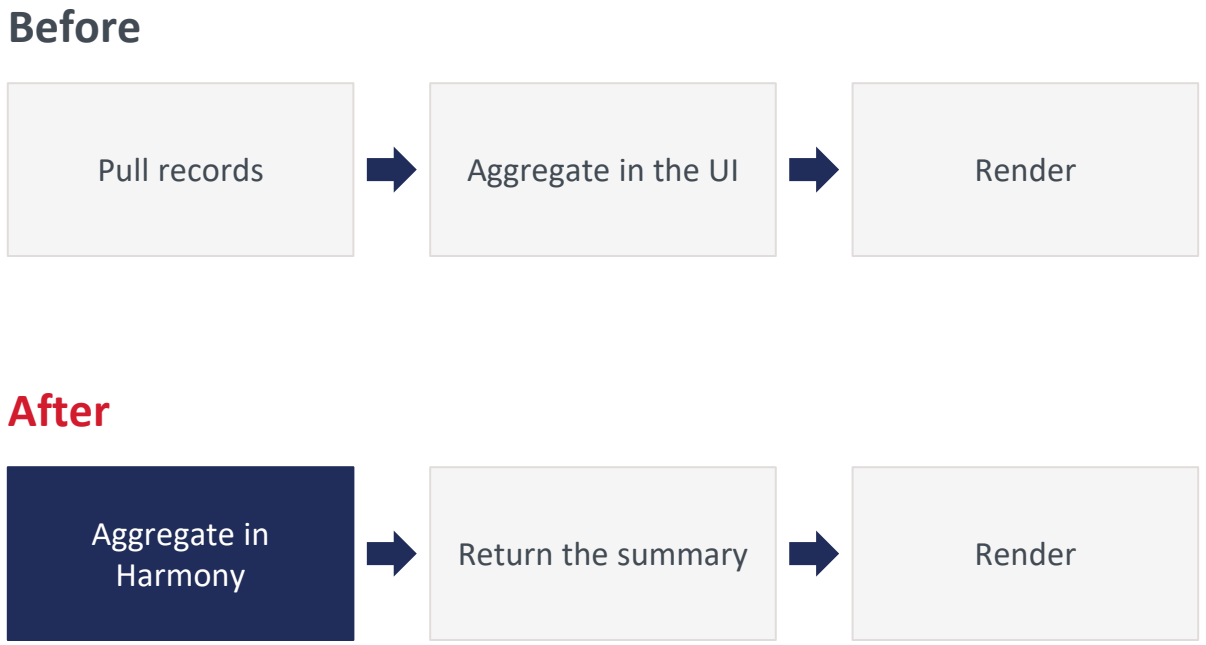


What plain OData gave us

- Individual records, row by row
- Math done in the browser
- Far more data over the wire
- Slower widgets as volumes grow

Partnering with Synergex on aggregates

- Reached out to Jeff and the Synergex team about adding aggregate support to Harmony Core's OData
- In November 2025, Synergex delivered a new, improved Harmony version
- Immediate, positive impact on UI development



The math moved to the server, close to the data. The UI just renders the answer.

The impact on the past due loans widget

Time to gather, calculate, and render one widget

15–20 s

Before: raw records, aggregated in the UI



< 2 s

After: aggregates calculated in Harmony

Each widget now renders in under 2 seconds.

Taking development in-house

After the proof of concept, we took over development from our third-party partner and made it our own.



More core data

Added more ISM files to our Harmony project to feed new widgets



New calculations

Built the calculations behind additional widgets in the Harmony service



A faster UI

Tuned the Angular front end and the API layer for overall speed

A small, focused team



2

UI developers



1

API developer



1

Synergy developer

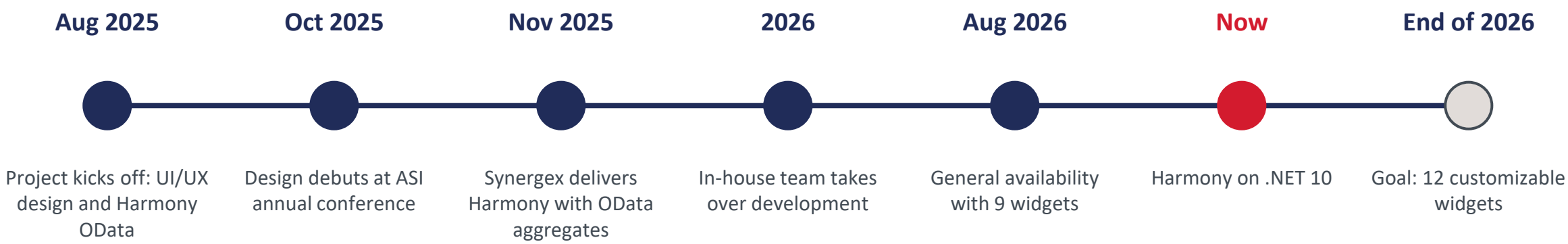


1

Product owner coordinating sprint work

Every layer covered, from the Synergy core to the Angular front end.

The road so far



2026 by the numbers

9

Widgets live at general availability, end of August

12

Customizable widgets targeted by year end

5

Steps in our repeatable widget process

.NET 10

Harmony version now in production

A repeatable process for every widget

1



Identify the data

Determine what data the widget needs from the core

2



Add ISM files

Add any needed ISM files to the Harmony project

3



Harmony endpoints

Code the API endpoints in the Harmony service

4



UI API endpoint

Code the matching endpoint in the UI API

5



Angular UI

Build the widget UI in Angular



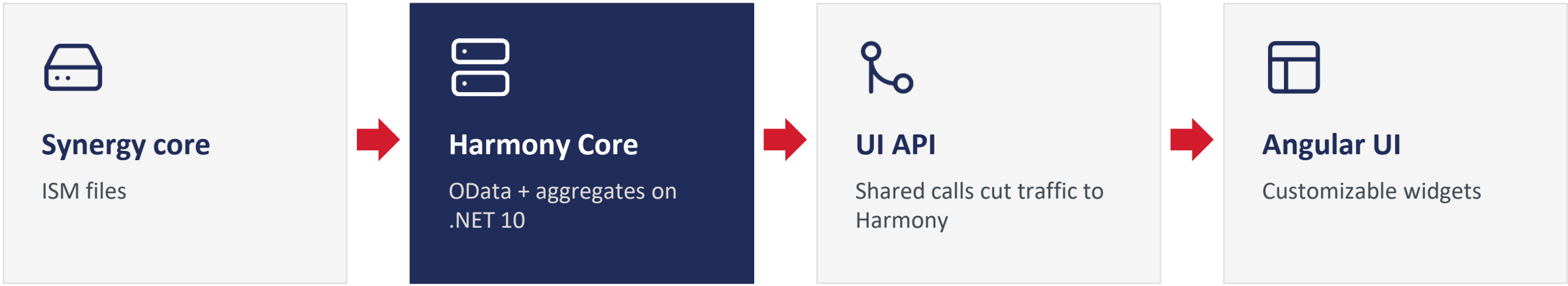
Same steps every time: predictable estimates, easier onboarding, faster delivery.

How the architecture evolved

Then

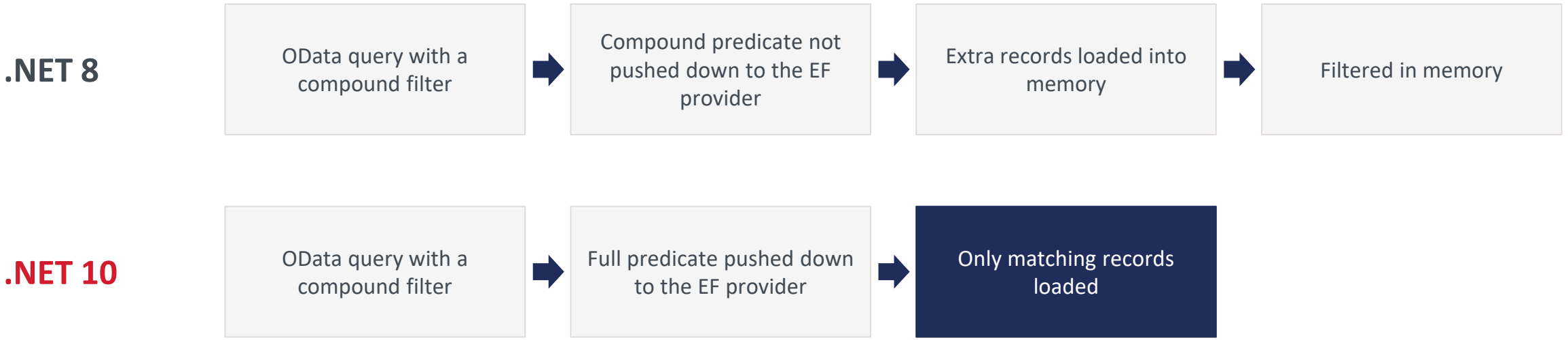
Core → Harmony Core OData → Angular UI

Now



Harmony on .NET 10: more efficient queries

Compound filters, such as days past due and branch together, now reach the database intact.



Filtering happens in the database, not in memory: less data loaded, less work per request.

What we learned

Start with the data

Every widget begins with a question: what does it need from the core?

Partner with your vendor

Asking Synergex for aggregate support changed our trajectory

Borrow expertise, then own it

An outside design team got us moving; bringing it in-house made it ours

Make it repeatable

A five-step process turns each new widget into routine work

What comes next

Reach 12 widgets

Complete the remaining widgets by the end of 2026

Keep tuning performance

Build on the .NET 10 gains across Harmony and the UI API

Continue working with Synergex

Keep making the most efficient use of Harmony Core to reach core data

Real-time managed services widget

Live updates for customers on managed services processing

9 → 12

Customizable widgets by year end



Thank you. Questions?

Matt Canaday
VP, Software Development, ASI
Matt.Canaday@asiweb.com

Special thanks to Jeff and the Synergex team.