Position Paper: Making Our Products Modular

Richard Steiger

11/9/22

**Contents**

[1 Purpose and Scope 1](#_Toc118914981)

[2 Examples of Modularity 2](#_Toc118914982)

[3 Current Uniphore Product Line Situation 3](#_Toc118914983)

[3.1 Data and Semantic Models 3](#_Toc118914984)

[3.2 Communication Protocols 3](#_Toc118914985)

[3.3 Functional Components 4](#_Toc118914986)

[3.4 Coordination 4](#_Toc118914987)

[3.5 Finding And Understanding Things 5](#_Toc118914988)

[3.6 Costs 5](#_Toc118914989)

[4 What Might Deep Modularity Look Like? 5](#_Toc118914990)

[5 Conclusion 6](#_Toc118914991)

# Purpose and Scope

I’ve been stewing on some thoughts about our current and possible future directions, and I’m now ready to share them in semi-baked form with you.

I’m choosing to do so privately with you, so please treat this current note incarnation as confidential until we’ve melded minds on this. At that point, if you agree with my core points, they’ll fare much better coming from both of us, and if you don’t agree with them, then we can quietly drop them, without unnecessarily ruffling others’ feathers.

My central insight is that while we talk about Our vision and strategies of embracing AI to understand customers’ issues and how best to handle them, adding more AI capabilities, shortening TTV, and the other pillars, from my vantage point, we’re missing some of the key technical foundations and critical building blocks needed to make the most of this strategy, and grow our competitiveness and opportunities in our market.

What’s missing? In a phrase, ***deep modularity, based on well-designed abstractions***.

# Examples of Modularity

Perhaps the best way for me to explain this is for me to give some powerful examples of the kind of modularity I envision, drawing from recent history of mathematics, science, engineering, and other fields:

* ***The Metric System***: until we had standard measurements for length, mass, time, and other physical dimensions, all scientific work was slowed, and was difficult to share among nations and groups. In the wake of the Metric System’s invention, all fields flourished.
* ***GMT***: until we had an international standard for time-bases, astronomy, ocean navigation, and commerce were all stifled.
* ***Standard interchangable machine parts***: while the history is long and contorted, one of the key innovations made in the 18th century was the eventual adoption of standard machine screw sizes, and numerous other key parts of guns, pulleys, looms, and other industrial equipment. This led to the development of the first mass production systems, and is credited with the US winning the American Revolution.
* ***National currencies***: as flows of property, goods, rail-lines, and early industry exploded westward in the US in the 19th century, every state, county, large city, and dog patch had their own currency and coinage, with no clear regulations nor exchange standards in-place, greatly limiting commerce, manufacturing, government, taxation, etc. With the adoption of the US Dollar, these other currencies were quickly retired, and industry surged ahead.
* ***Dial tone***: Due to the (in some ways) good fortune of A.G. Bell - then AT&T - having a monopoly on telephone service, they established many standards that allowed voice communication between cities, then states, then the globe, all in a few decades.
* ***Data communication protocols:*** X.25, SNA, Wang Net, DecNet, Ethernet, TCP/IP, W3C, …[[1]](#footnote-2)
* ***Legal systems:*** long evolution from Roman, through English and EU courts, to Magna Carta, Constitutions, legislatures, elections, and civil laws created increasingly level fields on which people got along, traded, got married, got buried, and otherwise Got Stuff Done.
* ***Standard bio-design encoding: DNA, RNA:*** Without which there wouldn’t be.
* ***The OED and other standard lexicons and vocabularies:*** Universally despised when being rolled out, such standards for the English language enabled it to become the one and only link language on the planet.
* ***Standards in audio synthesis and engineering:*** Enabled the rise of the recording and film industries, and explosion of musical creativity over the last 120 years.
* ***Photographic, film, video, and other media standards:*** Similarly enabled the rise of the film and TV industries, and an explosion of cinematic creativity.
* ***Standard utilities:*** Residents of modern economies have no idea how fortunate they are to be able at any time of day, to open a tap and get drinkable water, or plug-in a heater without it exploding.

And so on. I bother to mention these examples, both because they’re compelling (and I enjoy the retelling), and because I draw huge design inspiration from these and similar world-altering innovations.

The common denominator shared by all of the above advances is that people came together to *adopt singular, general, broad, and deeply grounded abstractions, concepts, and related designs, lexicons, and other artifacts, to enable emergence of value-generating components, and their free connectivity and interoperability in their respective domains*.

As mentioned, these various innovations enabled the next layer of bigger, more powerful, and more pervasive innovations, such as information processing and biotechnology.

# Current Uniphore Product Line Situation[[2]](#footnote-3)

So how does Our technology suite show-up from this perspective?

Before diving into the problems, let me mention that adoption of X has laid a good foundation, providing essential low-level modularity, in the form of provisioning, containers and pods, ingress and routing, standard service mesh and data fabric, integrated security and authorization, DNS, account and user admin, monitoring, CI/CD, and additional infrastructure components, all of which are essential in realizing the higher-order modularity I’m attempting to describe.

Turning to our challenges, let’s look at what we have today, along several dimensions:

## Data and Semantic Models

While there’s overlap in choice of concepts, and some in choice of representations, there are more gaps than agreements.

Our problem isn’t lack of data, nor even in the nearly universal adoption of JSON object representations. The problem is that *our products aren’t semantically aligned*, and only informally and superficially share concepts.

This is of course expected, since until recently launching the Core Services initiative, no-one was charged with rationalizing the products’ individual ontologies, and extracting a common lexicon and vocabulary that spans our various products.

## Communication Protocols

I frequently imagine dialogs occurring a several stages along the SDLC timeline for our services:

First, in our design sessions:

Yup, we can use Kafka over here, then some gRPC there, but wait, maybe for it to be Real Good, we should use REST, instead, right?[[3]](#footnote-4)

Next, overhearing just the local end of a phone-call between one of our solution engineers and a partner’s developers:

Yes, we use Kafka for topic T.

Oh, what’s the end-point? I try to not remember such details, … lemme see … Okay, it’s <long URL>. I suggest you write that down.

And we push AI signals as FrobnixV33.8.uvq.proto blobs. No one informed you? Oh, shoot, I can’t give you read access to our repo, so I’ll mail the proto, gimme a minute, …

This illustrates the present situation with end-points, ***an instance of a missing abstraction***. A bit like a utility customer who’s about to plug in a new TV to watch a cricket match starting in 15 seconds, and has to call the utility company to know what the voltage is, whether AC or DC, if AC at what frequency, and which TV encoding standard to select. Versus someone in Portola Valley, plugs their TV in, and enjoys the ads before the match.

## Functional Components

Functional components, which provide ASR, AI analysis and recognition, core analytics, and myriad other functionalities, are implemented in several ways across product lines, which might be unified over time, but if not, allowing solution engineers, DS teams, and other consumers to just stick with what they know works, and we’re still stuck with confusing multiple ways to do more or less the same thing.[[4]](#footnote-5)

However such functional overlaps are resolved, it doesn’t affect the fact that all of today’s building blocks are locked-up in huge, coarse-grain per-product services, requiring hopping useless domain and management boundaries, and having to make way too many decisions and choices to fully utilize them. If there are initiatives to deconstruct product stacks into finer-grain, and individually-usable components, I’m not aware of any.[[5]](#footnote-6)

And even if the blocks magically emerged from their monolithic cocoons, since each product line has its own ontology and APIs, functions can’t be “patched” together to form solutions, without reams of expensive, slow-to-market custom translators, adaptors, and gluons. I often wonder just how much the average line of custom code costs in time, dollars, missed market opportunities, etc.

## Coordination

Another consequence of being embedded in coarse-grain services is that each block is orchestrated and administered as part of the enclosing service, another bridge to cross when cross-connecting functions.

## Finding And Understanding Things

Say I need a certain function (statistical, ML, whatever), and suspect there’s a solution lurking somewhere in one or more of our products. How the heck do I find it?

Okay, asking around since yesterday, I found it. Now, how do I learn exactly how it works, what its edge cases are, and other technical aspects? There’s some Java docs, but they’re shallow…

Okay, let’s test-bench it, and throw a test conversation at it. Huh, what the hell, the service crashed, found out (overnight) it was because the convo ran more than 15 minutes and over-ran a buffer. Okay, shorter segment, but at 10:24.8, it returned a 789 error code. Not in the doc, so what do I do? Ask Raoul, but he’s 10 hours away.

Okay, day 4: I get a conversation blob back, and notice that there are new AI signals hanging off some turns, and they in turn are linked to some new facts and intuitions, all new to me. Where do I go to follow the thigh-bone to the femur to the (new) ascending aorta, and learn what’s it attached to?

## Costs

The above hurdles multiply together to render any cross-functional dataflow topologies labor-intensive, fragile, intermittently available, all of which makes creating solutions far more difficult to build, maintain, and refine. Even if ontologies become aligned, protocols move closer to being like wall-plugs, and the granularity issues are solved, we’d still have a hard time doing the equivalent of patching a jazz trio together for a piece with the speed and agility many of today’s electronic musicians can do in their home recording studios.

And these facts in turn don’t just put a high floor under our efforts to lower TTV, but inhibit any experimentation and tinkering on the lab bench, that kids with soldering irons and oscilloscopes (like me) were able to develop deep, almost physical intuition, and from which the first analog ICs, coffee-pot web-videos, chat services, Apple computers, and a few hundred thousand other innovations sprung.

From that point of view, how the hell can we expect to quickly adopt and combine ever more complex and sophisticated AI capabilities, if we don’t flatten these impediments?

# What Might Deep Modularity Look Like?

I have nothing hidden up either sleeve, I’ve already put my cards down, face-up. So let me summarize the changes I see that would enable deep modularity, allowing Uniphore to become more ***deeply agile***:

* ***data tone***: like dial-tone, it’s always on, just plug-in and interact
* ***instant lunch***: functions are available, discoverable, documented, and ready to rock as needed
* ***lazy resource usage***: functions that are dormant don’t suck power, but light-up in a flash when needed; these turtles go down as far as you can look
* ***street maps***: every function, datum, file, concept and artifact is linked off a master, comprehensive and easily navigated street map, is where you expect it to be, and is available when you need it
* ***visual patching***: the wiring of functional blocks to form solutions starts (and pretty much ends) by drawing a flowgraph of boxes for functions and wires for flowpaths
* ***visual rehearsal, testing, debugging, and tuning***: facilitated by dropping and following control and/or data tokens flowing along the flowpaths; can create start points appearing as pointed droplets, breakpoints are roadblock signs, etc. Flow testing, tuning, and troubleshooting are a breeze.
* ***visual operational oversight***: for ops folks, services, their pods, containers, and wiring are viewable in a similar graphical manner to workflows, shaded and colored to indicate their execution state, error rate, CPU load, storage utilization, and other key ops signals.

Now, with such a magic carpet, how many more engagements can we take-on and crush at a time?

# Conclusion

Based on my experience designing and in cases implementing such an integrated, highly structured, interconnected, and intelligent system, doing so may not be as costly and take as long as it might first sound.

1. I’ve always been amused by the preface to Andrew Tennenbaum’s book on Data Communications, where he says “Data communication standards are great! There are so many to choose from!” [↑](#footnote-ref-2)
2. Please read the rest of this note as based on my still nascent product-line knowledge, hence possibly incorrect at numerous points. [↑](#footnote-ref-3)
3. How many distinct languages did Mother India have into the 20th century? [↑](#footnote-ref-4)
4. Just how many ways can you compute the mean and variance of a time-series? [↑](#footnote-ref-5)
5. Prefixing “micro” to their names doesn’t help. [↑](#footnote-ref-6)