No Big Rubber Shoes: A Jaded Practitioner's Guide to Building on Jube

Author: Richard Churchman
Version: 1.0 FINAL
June 2026

I am Richard, the developer of Jube, which is open-source software for real-time Anti Money Laundering and Fraud Detection. I like value chains [a lot] for the distribution of Jube, which means that I spend my time slinging code or scaffolding infrastructure, while working with System Integrators (SIs) to hold out value propositions to their clients, with whom they have deep trusted relationships. This paper provides some reflections on my engagements to help SIs embark on a successful Jube implementation, and value-added service of their own.

Taking a step back, it is worth being clear about the objectives of the SI: Create a service offering for their clients to meet regulatory requirements in Anti Money Laundering (AML) and Fraud Detection. This is the whole offer, the whole promise, and the overarching guiding principle. This framing is practical, as regulations have become strikingly technologically sophisticated, and maintaining regulations as being the authority, has the effect of giving a free product management function for the SI. While Jube is highly capable and wholly configurable, it is important to resist the urge to offer anything that does not come back to the first principle of regulatory satisfaction; An SI will be busy enough delivering the promise, and the consequences of doing it wrong are material.

There is a line from the film Lock Stock and Two Smoking Barrels that says something better than I ever could, and although perhaps running contrary to our own objective in narrative, illustrates the point: “We rob Post Offices, Steal Cars, what the [redacted] do we know about antiques”. Keep this bizarre, and slightly inappropriate, scene in the back of your mind from the outset, do one thing well, with that one thing being the software infrastructure and accompanying analytics to the satisfaction of regulations for AML and Fraud Detection, no more, no less.

SIs do not by their nature know much about AML and Fraud Detection regulations by default, which based on the direction above, could be a cause

for self-deselection. It follows that lacking depth of understanding of regulations is something that does need to be resolved. Fortunately, and this is especially true in non-western markets, the regulations are nicely written, approachable, highly prescriptive, and when taken together with Jube's guidance notes, allow for a path to acquire this know-how to an adequate standard. The reality is that the client will direct exacting requirements, and while it does not absolve the SI from doing the study - and this is especially true of those responsible for the sales and technical pre-sales – the client will specify, test and sign off the specifics. Put differently, the consumer of this value proposition does not expect deep expertise in the regulations, they just don't want total ignorance.

Jube is a capable platform, and these days, there is little to nothing that can't be satisfied in the context of the stated aim above, which is both a blessing and a curse, as in the early stages of the project, the temptation is to say yes to everything. When the client is speaking to a problem, it is almost always masking fear of regulator fines, or in response to current material losses, and clients will expect urgency in response. Know this: Big Bang projects (i.e. everything at once) mean exactly that, things will explode. From the outset, plan hard, "peel the onion" in an iterative approach, and focus the project on systems integration, through to a smattering of passive rules, thereafter the same rules in real-time, to analytics, and only at that point, bring something small in every week. The iterative approach will, without question, result in a project that executes far faster, and avoid coming across to the client as an amateur. While clients might grumble at the initial pace, on reflection, they will value the wisdom and maturity over speed of execution. On the subject of wisdom, it is worth a side note that clients are extraordinarily disillusioned with junior-heavy teams regurgitating Chat GPT for material challenges, and while we slow-minded elders can't generally rival our Gen Z counterparts in speed of information assimilation, clients want to see the grey hair. Make sure demonstrable systems experience is front and centre in the value proposition, while deferring to the juniors for the cognitive heavy lifting, inside strict guardrails and circuit breakers.

Jube has two means of integration, being JSON over HTTP or AMQP. Jube makes no assumption about metadata, and via no-code / low-code configuration, can consume varied client payloads in a matter of minutes. Notwithstanding, make no assumption about integration, and workshop hard with the client, to arrive mutually at a protocol specification. The reality on the ground is that the client may not have native means of supporting an integration payload quickly, nor in real-time, and this is nearly always cause for project delay. In the most ideal of circumstances, developers are presented with a protocol specification, and they drop in the integration in a logical place in the real-time flow, but there are always a plethora of other considerations; Is there any middleware involved (probably, and likely a separate team)? Is the source system a walled garden (e.g. Temenos), and if so, what "out of the box" event flows are available, and are they real-time? There are nearly always integration solutions, even if it is falling back to hideous database polling patterns as a passive stopgap, but having an early understanding of constraints will reduce the, aforementioned, risk of appearing amateurish. Furthermore, don't underestimate union dynamics, and take special measures not to alienate the client's technical teams from the outset (imagine the client's own Software Architects on a "go slow" given their perception of a clown show tripping over big rubber shoes to

comic effect). If the software is going into client infrastructure, rather than being hosted at the SI, engage with Enterprise Architecture and Security on a similar basis, as they will exhibit a similar dynamic, but don't just do it, embrace it, as this is free product management also.

Jube is thought of as a no-code / low-code platform, but that framing is somewhat an admission of failure, as rules should only exist for as long as it takes to create sufficient amounts of data to bring about the proper application of analytical methodology. Be very cautious with selling a battery of "best practice" template rules, as this stores up an unacceptable operational impact (think angry call centre manager dealing with thousands of angry customers who can't make payments). Only hold out template rules with a strict caveat that they are not tested against real data, representing the client's actual portfolio behaviour, and that they exist only as a short-term measure, pending proper analytical methodology being applied. Once data is flowing, no rule should ever be considered unless there is a comprehensive impact assessment, and, ideally, an understanding of the value Detection Rate (this being the sum total of transaction value, not the count of transactions). Value Detection Rate will be all but impossible to obtain in the early days, as it is highly unlikely that anything like enough known fraud data labelling will be available from the client, and even on initial integration of Jube, and with elegant case management workflows, chances are the end users can't be bothered to label fraud data. Educating users to label fraud data speaks to the importance of operational processes not being an afterthought. Operational teams tend to exhibit more enthusiasm than IT counterparts, but better to drill the operation early, with processes that are sleek, and respectful of their time.

Picture for a moment a DJ on the decks, spinning the discs, mixing the tracks. This is the polar opposite of what needs to be projected to the client at any point in their engagement with the SI. The consequences of anything going wrong, be that in pre-sales through to production, are highly material, and this will happen if there is any freestyling whatsoever. Anecdotally, nobody wants to be responsible for all customer transactions declining in real-time because Chat GPT dropped in some nonsense (this is genuinely deal-breaking, as invariably initial contracts will be short while the SI shows chops). Run Books are absolutely vital, a strict sequence of steps for processes, tested, proven, and for all interactions with the project (not just server tinkering). Test the Run Books hard, perform chaos testing (which is to say, kill a service under load, and recover from it). Drill the project team in the Run Books, and when they are drilled well, do it again. If it is not in the Run Book, it does not exist in all but the most extreme of unforeseen outages. Under unforeseen extreme outage pressure, without a Run Book, any urgent freestyling must be on the basis of a strict four-eyes review, and immediately compiled into a Run Book, ratified at the earliest opportunity. It goes without saying, don't just copy commands from Chat GPT, the world will spin off its axis, and I am wholly confident in stating that this will be the singular cause of your most embarrassing and commercially irrecoverable outages. Monitoring all architectural components and their logs is essential — if the client sees a problem before you, credibility is long lost. To another ubiquitous film reference, the closing scene of Burn After Reading is where we don't want to be across from the client.

Jube is open-source and is distributed under AGPLv3, which means that any modifications made to the software must accompany source code that is available to the end client. This is just the way it is, and even if I wanted to, that horse has bolted, and the AGPLv3 code is well in the wild (some 80 unique cloners every two weeks at the time of writing). The question for SIs is therefore, how do they create proprietary Intellectual Property (IP) and niche value, given that they need to expose any code changes made in the Jube core as part of AGPLv3 licence compliance. Stepping back, and looking to other well-used AGPLv3 software, such as Redis, the answer is hiding in plain sight; configurations and extensions developed inside the Jube extensibility framework are IP that are retained by the SI, and are a practice well-documented in the open-source community, and is frankly fairly obvious. There is ample opportunity for an SI to develop their own IP, without stepping on any AGPLv3 landmines. The client takes licence compliance extremely seriously, and it is well worth getting ahead of those conversations, as any hint of obfuscation will cripple credibility in the wider relationship, and is entirely unnecessary given Jube’s extensibility framework.

More generally, open-source is an uncomfortable initial proposition for a client, but objections rapidly dissipate by shining a spotlight on total transparency, the client’s own security scanning on the code providing for unparalleled security assurance, with data sovereignty always maintained, the opportunity to maintain total business continuity given total access to source code and of course total circumvention of vendor lock-in and the ubiquity of price gouging.

To conclude, none of what is written above is particularly complicated, but very little of it is instinctive either. The iterative approach feels slow until it doesn't. The Run Books feel bureaucratic until the phone rings at midnight. The regulatory grounding feels academic until the client asks a question and someone in the room actually knows the answer. The discipline compounds quietly, and the SI that internalises that early is the one that builds a practice rather than survives a project. The regulatory environment is tightening globally and will continue to do so — the market available to a well-prepared SI is not a window, it is a door that is getting wider. I operate in the part of this value chain where I add most value and, frankly, where I am happiest. That means I have every reason to want the SI to succeed, and no reason whatsoever to see them fail. That alignment is, I think, worth something.