greyspark partners | valantic

Low-code/ no-code in the Financial Services Industry

Criteria to consider when selecting low-code/no-code technology

Contents

Low-code/no-code in the Financial Services Industry

Criteria to consider when selecting low-code/no-code technology

Less than one year after his return to Apple Computer Inc., the company’s then-Interim CEO Steve Jobs explained his overarching rule for software design in a 1998 Business Week interview:

Almost two decades on, and Jobs’ thinking can be associated with the influence behind not only numerous tech start-ups, but also the approach that users of those companies’ services demand when consuming the products that they create. In other words, the arrival of the App Economy. And just like any viable economic model, the App Economy is fueled by a labor force of software development professionals for whom churning out miles of code in a specific programming language is no longer the essence of the role.

Rather, working as a professional developer in 2025 means having the ability to expertly delve into multiple aspects of a technology stack, various platforms, data sources, databases, network layers, APIs, security mechanisms and procedures that are used to build software solutions – and anything that can be done faster is a ‘win’.

> “Simple can be harder than complex … But it’s worth it in the end because once you get there, you can move mountains”

Hence, the rise of so-called low-code/no-code (LCNC) technologies which provide frameworks that allow for the rapid development of microservices and even entire applications, all while enabling cost savings and – in some cases– empowering non-software developers to customize or create their microservices independently.

This article, produced in partnership with GreySpark Partners, explores the characteristics that corporate and investment banking institutions should look for when selecting a LCNC framework for transaction automation.

The drivers, benefits & risks of low-code/no-code solutions use by Financial Services

The rapid rise in the popularity of the LCNC approach to microservices design has been witnessed since 2020 as part of the wider Buy & Build trend that began to be talked about in 2015. The degree of maturity of LCNC technology, offered by companies such as Genesis Global, valantic FSA, and Xceptor, means that the possible benefits associated with adoption are well understood.

In 2025, many large financial services firms no longer have the budget or resources to develop an entire technology stack in-house (‘Build’), but at the same time they have operational requirements that are too sophisticated to rely solely on off-the-shelf (‘Buy’) solutions. As a result, there are now three models for building capital markets systems, specifically:

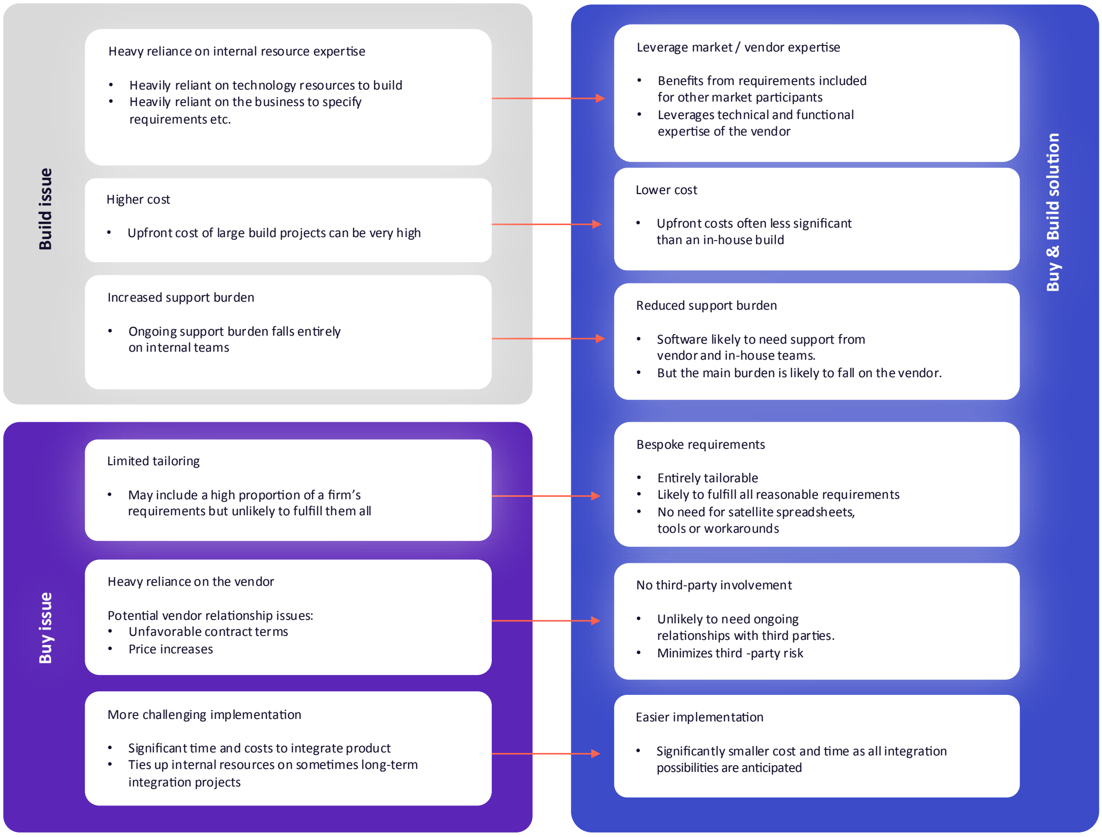
- **Buy:** Off-the-shelf packaged software
- **Build:** Bare-bones development frameworks
- **Buy & Build:** Frameworks which provide an accelerator of some sort.

A common saying is that necessity is the mother of invention and Buy & Build solutions were invented to overcome the issues with Buy, and with Build.

Buy & Build frameworks were created to solve some debilitating problems for developers, as shown in **Figure 1**. In capital markets, standalone 'buy' and 'build' solutions exposed limitations of both the bank and the third-party vendor. Relying entirely on an off-the-shelf solution would subject the bank to unfavorable contract terms, unsuitable upgrade schedules and limited flexibility, ultimately impacting the bank's operational efficiency and competitiveness. On the other hand, relying solely on in-house development make for lengthy, complex and expensive transformation projects, often leading to a strain on resources that would divert attention from core revenue generation activities. These challenges led to the development of Buy & Build, providing an antidote to the challenges faced by standalone 'buy' and 'build' solutions. Specifically, the premise underlying the approach is that users can rapidly develop and deploy a fully functional offering while retaining the ability to expand, remove or customize the software in the manner needed to meet new challenges in an evolving operational landscape, thus creating competitive differentiation. Ultimately, Buy & Build delivers a best-of-both, high functioning platform from the start, without creating dependency on the system's core trading functionalities or on software development teams, whether they be in-house, vendor or certified third-party developers.

Figure 1: Buy Issues, Build Issues and Buy & Build solutions to software development in Financial Services

Source: GreySpark analysis



Build issue

Heavy reliance on internal resource expertise

- Heavily reliant on technology resources to build
- Heavily reliant on the business to specify requirements etc.

Higher cost

- Upfront cost of large build projects can be very high

Increased support burden

- Ongoing support burden falls entirely on internal teams

Buy issue

Limited tailoring

- May include a high proportion of a firm's requirements but unlikely to fulfill them all

Heavy reliance on the vendor

Potential vendor relationship issues:

- Unfavorable contract terms
- Price increases

More challenging implementation

- Significant time and costs to integrate product
- Ties up internal resources on sometimes long-term integration projects

Buy & Build solution

Leverage market / vendor expertise

- Benefits from requirements included for other market participants
- Leverages technical and functional expertise of the vendor

Lower cost

- Upfront costs often less significant than an in-house build

Reduced support burden

- Software likely to need support from vendor and in-house teams.
- But the main burden is likely to fall on the vendor.

Bespoke requirements

- Entirely tailorable
- Likely to fulfill all reasonable requirements
- No need for satellite spreadsheets, tools or workarounds

No third-party involvement

- Unlikely to need ongoing relationships with third parties.
- Minimizes third -party risk

Easier implementation

- Significantly smaller cost and time as all integration possibilities are anticipated

Buy or Build?
That is the question.

Figure 2 shows the maturation of Buy & Build frameworks, which can be categorized into three sections - extensible packages, development frameworks and LCNC platforms. As highlighted below, valantic FSA's x-gen solution is pioneering the LCNC trend in the capital markets industry.

Figure 2: The different types of the Capital Markets industry Buy & Build solutions Source: valantic FSA and GreySpark analysis

Extensible packages	• Calypso • FlexTrade • SmartTrade LMS	A complete package with a set of APIs allowing users to expand or replace the functionality by injecting their own code.
Development framework based low-code/no-code	• Adaptive Hydra • Beacon • Itiviti Tbricks • Numerix CrossAsset	Software development kits combined with pre-existing services, workflow engines and application accelerators.
Platform centric low-code/no-code	• Genesis Global • Velox • Xceptor • x-gen (by valantic FSA)	A collection of technical and functional services that can be combined to rapidly create business applications.

Features / Functionality of Buy & Build Solutions

The Buy & Build approach served as an umbrella of inspiration for technology vendors developing LCNC approaches to microservices design.
The growth of the Buy & Build approach as the favored method of technology architecture for financial services companies is fueled by:

- Strong growth in spend towards risk management and middle-office solutions;
- Growing rates of trading technology equipment adoption among buyside firms; and, for technology, vendors
- Recent growth in the addressable population of asset managers and corporate and investment banks to sell to in-growth markets such as Asia-Pacific.

With those Buy & Build growth drivers in mind, key factors fueling LCNC solutions adoption include:

- Increasing digitalization across the capital markets industry
- A general shortage of developers and development expertise for the volume of work required within firms or institutions
- The requirement for fast-to-market applications in response to significant changes across the industry – for example, increased remote working, increased use of Cloud and SaaS technology, increased electronification of markets and increased regulatory compliance burden
- Sector-wide acceptance at management-level that technology of this kind can be trusted
- The ongoing push to eliminate uncontrolled and insecure end user-developed spreadsheets, databases and applications that create an inadequate audit trail or lead to lax management oversight

Once adopted though, the main benefits of platform-centric LCNC solutions and development framework-focused LCNC solutions for banks and buy-side include:

- Shorter times to production for applications and the microservices that they entail
- Resource optimization for in-house developers, wherein the mundane or one-off development work is done within the solution, thus creating time that can be spent on higher-value tasks

Non-developers – that is, business users – can also be looped into application development using LCNC solutions, allowing for an increase in the productivity of the IT team. This is less easy for development framework-based LCNC solutions, which require some understanding of coding and software design.

The downsides or risks for banks and buy-side firms of adopting LCNC solutions is the opaqueness of the source code. This can lead to issues with application development such as bugs or other types of security flaws. Non-developers may not have sufficient understanding of how the technology works, thus restricting their ability to create robust, durable and secure apps.

Therefore, it is important to distinguish between development framework-based LCNC solutions versus those that are more 'holistic' and non-developer user friendly. While all LCNC systems are typically built on robust technology stacks, such that they are as modular and scalable as possible, development framework-based LCNC solutions are typically more like runtime environments in that:

- Applications must be developed within the system's integration development environment (IDE)
- Applications must be run on the solution's own servers
- Custom code can be injected into the solution in the form of either scripts interpreted by the system's server or as linked libraries callable from its runtime
- APIs and connectors are available for systems integration.

Development framework-based LCNC solutions are often unsuitable for a large range of applications, such as customer journeys, data analysis and visualization, risk and pricing, and rich user interfaces. Barriers to adoption are that the development framework-based LCNC solutions are vast and complex to master, requiring substantial training for users to learn how to operate them and become sufficiently proficient to fully benefit from the productivity gains that they offer.

Development framework-based LCNC solutions are, however, specifically designed and optimized for one type of enterprise application: the automated handling, normalization and processing of streams of data and events according to pre-defined workflows.

In contrast, platform-centric LCNC solutions that exist as platform offerings are systems comprised of extensive ecosystems of software built around a baseline of core architecture. These ecosystems and the offerings that they can contain typically include:

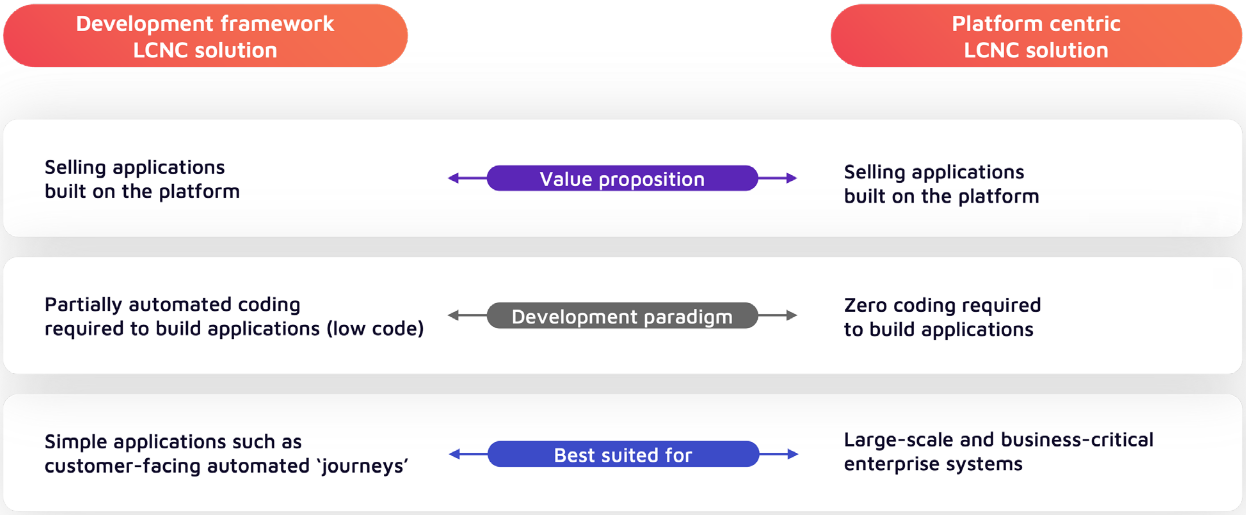
- Vast libraries of pre-built adaptors to common financial applications and markets' infrastructure
- Network of systems integrators that are acting both as prescriptors and implementation partners
- Community asset stores combining thousands of reusable components that are either written by the vendor or are shared by other clients and users

The network effect created by these non-functional benefits of platform-centric LCNC solutions means that the systems offer a productivity suite that development framework offerings struggle to compete with when addressing banks and buyside firms' use cases. The functional benefits of platform-centric LCNC offerings typically include:

- Cloud-based IDEs and productivity tools
- High-spec 'what-you-see-is-what-you-get' / zero-code frontend / GUI design tools, allowing for more complex interactions and visualization than grid / data panel layouts
- Integrated continuous integration / continuous development pipelines covering all steps including:
  - Requirements management
  - Source code management
  - Build
  - Automated testing for both backend and frontend
  - Deployment onto Cloud and on-premises.

Figure 3: Low-code/no-code platform differentiation

Source: GreySpark analysis



Development framework LCNC solution		Platform centric LCNC solution
Selling applications built on the platform	← Value proposition →	Selling applications built on the platform
Partially automated coding required to build applications (low code)	← Development paradigm →	Zero coding required to build applications
Simple applications such as customer-facing automated ‘journeys’	← Best suited for →	Large-scale and business-critical enterprise systems

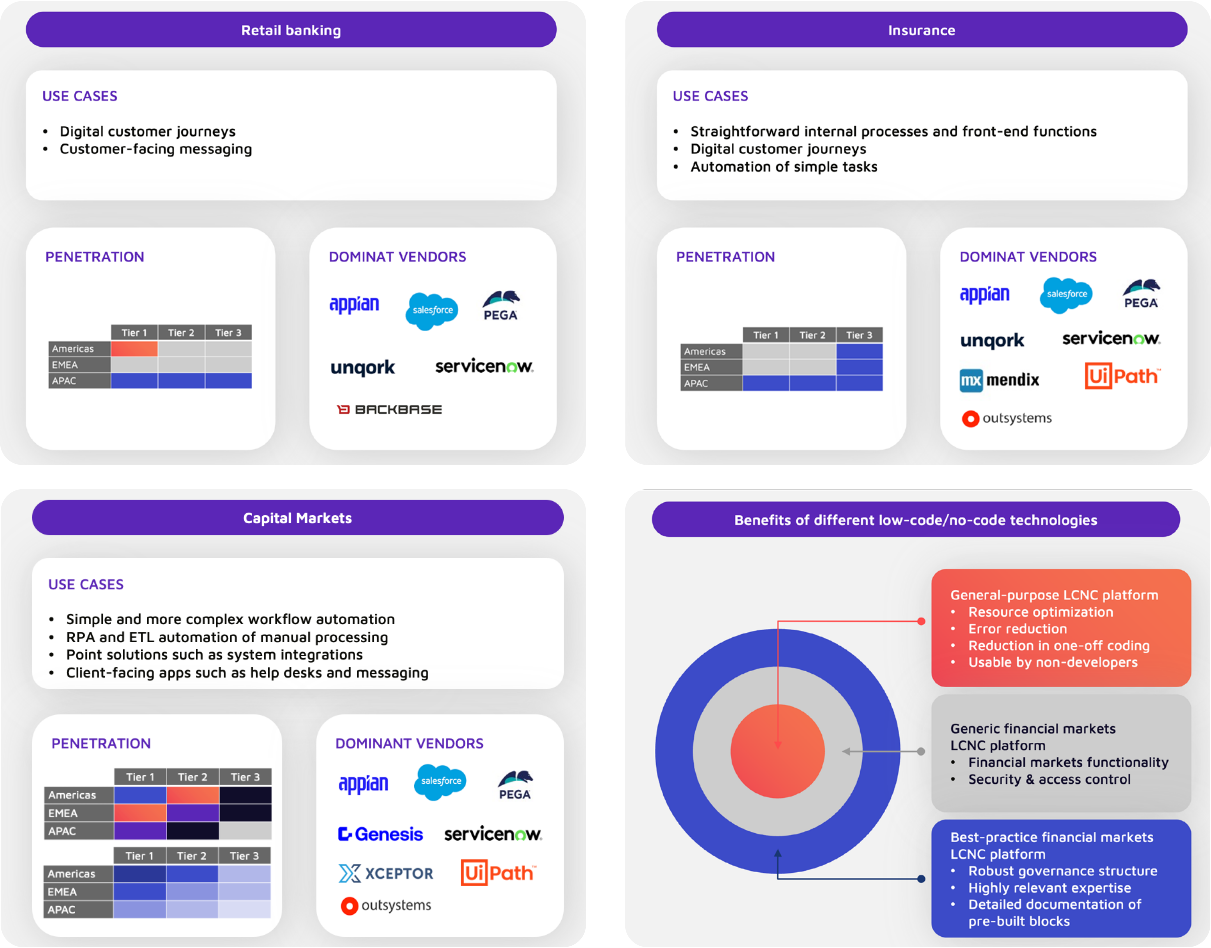
Figure 3 shows how platform-centric LCNC solutions can be differentiated based on their positioning across value proposition, development paradigm and the types of applications that they are best suited for.
GreySpark believes that many of the development framework-based LCNC solutions could evolve into more advanced platform-centric LCNC solutions, however, those capabilities are not yet organized, packaged and run as part of a coherent offering.

LCNC use cases in Financial Services

There are many possible use cases for LCNC technology in the financial services industry, from the relatively small-scale automation of manual processes and the replacement of risky end user-designed spreadsheets via innovative applications that address new business opportunities, through to the replacement of large legacy core systems **(see Figure 3)**.

Figure 4: The Use Cases of development framework-based LCNC and platform-centric LCNC in Financial Services

Source:GreySpark analysis



Retail banking

USE CASES

- Digital customer journeys
- Customer-facing messaging

PENETRATION

DOMINAT VENDORS

appian, salesforce, PEGA, unqork, servicenow, BACKBASE

Insurance

USE CASES

- Straightforward internal processes and front-end functions
- Digital customer journeys
- Automation of simple tasks

PENETRATION

DOMINAT VENDORS

appian, salesforce, PEGA, unqork, servicenow, mendix, UiPath, outsystems

Capital Markets

USE CASES

- Simple and more complex workflow automation
- RPA and ETL automation of manual processing
- Point solutions such as system integrations
- Client-facing apps such as help desks and messaging

PENETRATION

DOMINANT VENDORS

appian, salesforce, PEGA, Genesis, servicenow, XCEPTOR, UiPath, outsystems

Benefits of different low-code/no-code technologies

General-purpose LCNC platform
- Resource optimization
- Error reduction
- Reduction in one-off coding
- Usable by non-developers

Generic financial markets LCNC platform
- Financial markets functionality
- Security & access control

Best-practice financial markets LCNC platform
- Robust governance structure
- Highly relevant expertise
- Detailed documentation of pre-built blocks

Financial services firms typically have highly specific requirements, needing technology that is scalable, compliant and agile. As such, there are specialized LCNC providers that combine technological innovation with industry-specific knowledge and expertise, and whose LC building blocks are specifically designed to address requirements particular to financial or capital markets.

With this wide range of possible use cases in mind, banks and buy-side firms are typically observed as struggling to select the correct or 'right' LCNC product for the range of broad development tasks that constitute a specific use case. This struggle then leads technology buyers to misdiagnose the problem statements that a team or the institution overall is trying to resolve and – in turn – mismanage the vendor selection process that can result in, for example, selecting a packaged, platform-centric LCNC offering when, in fact, an open-ended development framework product presents a more applicable solution.

Figure 4 presents a framework that banks and buy-side firms can use to assess any LCNC vendor in the running for implementation inside of a specific use case, taking account of:

- ✓ The typical value proposition that providers of different ilk will typically present
- ✓ The ways in which they seek to monetize the technology they are seeking to sell
- ✓ The development paradigm they claim their offering can resolve
- ✓ The target users within an institution that they believe their solution(s) are most suitable for

Figure 5: A Taxonomy to compare low-code/no-code vendors' offerings

Source: GreySpark analysis

VALUE PROPOSITION

The type of problem that an LCNC vendor is primarily trying to address determines how and to whom the technology is sold

Solutions – the LCNC vendor aims to solve a specific business problem, the LCNC technology offered is a means to an end to rapidly develop a system covering a specific use case, sold to business stakeholders.

App accelerator – the LCNC vendor offers a range of reusable functional assets that can be leveraged by the client to rapidly build applications across a defined range of related use cases. Sold to business stakeholders.

Platform – the LCNC vendor offers a versatile technology stack, complete with a suite of reusable non-functional components as well as generic functional ones. The LCNC vendor is not promising to solve any business problem, rather it gives the means to the client to solve any (relevant) business problem by themselves. Sold to senior technology stakeholders.

TECHNOLOGY MONETIZATION

Part of the technology stack that an LCNC vendor generates the bulk of its revenue from:

Dev tools – licensing of the development suite and framework (the IDE), code generators, component and apps library, quality assurance and testing tools, static code analysis, combines practices of continuous integration and continuous delivery. Typically based on the number of developers.

Runtime – licensing of run time environment irrespective of the deployment methods (on premises, hosted, private or public cloud). Typically based on the number of use cases, of cores / central processing unit and of end users.

SaaS – licensing of the run-time deployed in a closed environment managed by the vendor, typically based on the number of applications and of end users.

DEVELOPMENT PARADIGM

Approaches offered by the LNCN vendor for application development.

Code generation – of loading a significant part of a developer's burden by automatically generating executable code for non-functional requirements and code stubs for the functional ones. Functional requirements then need to be implemented using an Object-Oriented development language (typically Java or C#).

Scripting & rules – business logic is implemented using scripting languages, expressions, formulas or rules definitions. The scripting language is proprietary to the LCNC vendor.

Zero code – applications are developed using graphical tools, assistants and what-you-see-is-what-you-get editors. No writing of code is required to build a fully functioning application.

TARGET USERS

Primary users of the technology offered by an LCNC vendor are:

Core developers – aimed at experienced developers with a solid background in lower-level programming , as well as extensive knowledge of the software development life cycle (SDLC).

RAD developers – aimed at RAD developers with an understanding of the SDLC and with high level development skills.

Citizen developers – aimed at IT literate business users with a strong functional background, a sound understanding of technology, but with limited or no coding skills.

SaaSification and its role

In 2025, the overarching theme of LCNC solution development continues to be 'SaaSification', where cloud-focused software as service models are developed with little to no coding. In fact, roughly 70% of business applications now use LCNC functionality. The core premise of LCNC SaaS is the ability to develop and deploy applications quickly. With pre-built modules and templates available, developers can focus on creating customizations and modifications to the existing code instead of writing it from scratch.

When offered in a packaged, platform-centric manner, LCNC solutions can provide users with drag-and-drop 'studios' to design processes for the applications to execute, sometimes with pre-built blocks that can be used to quickly create common application components. Examples of what those building blocks might look like for financial services use cases include popular system integrations; for example, with middle- or back-office systems, market and static data interfaces, and a range of APIs including FIX builds.

x-gen – Setting the benchmark

valantic FSA's long-standing and trusted LCNC platform, x-gen, provides the foundation layer for functionally rich business automation solutions across payments, securities and post-trade and crypto settlement domains.

x-gen was subject to a benchmarking exercise by GreySpark Partners, with the solution weighted against 'Use Case' KPCs and 'Development Paradigm' KPCs, across three components.

Figure 6: valantic FSA's x-gen Benchmarking Exercise, by Key Purchasing Criteria Source: GreySpark analysis

	USE CASE KPCS			DEVELOPMENT PARADIGM KPCS		
Vendor	Digital Journeys	Business Workflows	Enterprise Systems	Code Generation	Scripting & Rules	Zero Code
Actico	🟦	🟦	▒		▒	🟦
Aurachain	🟦	🟦	🟦		🟦	
Bizagi|	🟦	🟦	🟧	🟧	🟦	▒
Druid	🟦	🟧			🟦	🟧
Duco		▒	🟦			🟦
FintechOS	🟦				🟦	▒
Fintus	🟦	🟦	🟧		🟦	🟧
FlowX.AI	🟦	🟦		🟦	▒	🟦
Fortia Financial Solutions		🟦	▒		🟦	🟦
Genesis Global		🟦	🟦	🟦		
Genus		🟥	🟦	🟧	🟦	🟦
Innoveo	▒	🟦	▒		🟦	🟦
Instanda	🟦	🟦	▒			🟦
Oriana		🟦	🟦	▒	🟦	🟦
Synerise	🟦	🟦	▒		🟦	▒
Ultimus		🟦	▒		🟦	
valantic FSA		🟦			🟦	
Vermeg	▒	🟦	▒	🟦	🟦	🟧
WebRatio			🟦	🟦		
Xceptor		🟦	🟦		🟧	🟦

Legend: Assessed level of KPC fulfillment LOW/MEDIUM/HIGH

LOW: 🟥 MEDIUM: ▒ HIGH: 🟦

As shown in Figure 8, Use Case KPCs were assessed for three use cases:

1. **Digital Journeys** – LCNC software that enables a business to build digital customer journeys and customer-facing messaging or digital helpdesk apps and solutions
2. **Business Workflows** – LCNC software that enables a business to achieve the automatic ingestion, transformation and input of data into an independent flow of tasks, documents and information across a variety of trading processes
3. **Enterprise Systems** – LCNC software that enables a business to build trading systems or other, large, complex offerings with several business workflows synthesized to perform critical functions that need support from various data sources and departments as well as high security.

Development Paradigm KPCs, on the other hand, were assessed against three criteria:

1. **Code generation** – The offloading of a significant part of a developer's burden by automatically generating executable code for non-functional requirements and code stubs for the functional ones. Functional requirements can then be implemented using an object orientated development language – typically Java or C#
2. **Scripting & rules** – Business logic implemented using scripting languages, expressions, formulas or rules definitions. The scripting and rules definition languages are typically proprietary to the LCNC vendor
3. **Zero code** – No writing of code is required to build a fully functioning application. Applications are developed using graphical tools, assistants and what-you-see-is what-you-get editors

x-gen's high score across 'business workflows' and 'scripting & rules' is reflective of x-gen's ability to leverage automation across a variety of trading processes. valantic FSA's proprietary, LCNC technology can be seamlessly integrated and modularized to meet the client's requirements.

x-gen: Low-code/no-code platform of choice for workflow automation

The solution can reduce manual intervention among trading workflows by 95%. The x-gen platform has several key differentiators from industry-standard LCNC platforms, including:

- ✓ Out-of-the-box modular deployment to the client's requirements
- ✓ Low-code/no-code visual programming that allows the user to be self-sufficient and take control
- ✓ Incorporates out-of-the-box support for business standards e.g. Swift FIN & Swift ISO 20022 (support for all message categories), SEPA, Target2, FIX, etc.
- ✓ Capable of high-volume, low latency transaction throughput via a highly scalable and dynamic run-time architecture
- ✓ "Plug-and-pay" approach enabling seamless integration with existing integration, both legacy and modern

More specifically, x-gen provides the following capabilities for the payments, securities post-trade and crypto-settlement domains:

- **Payments:** Payments systems integration, straight-through processing, and exceptions investigations across different payment scheme standards.
- **Securities post-trade:** Securities post-trade systems integration and transaction orchestration, including trade order vs. trade confirmation matching and reconciliation.
- **Crypto settlement:** Cryptocurrency wallet integration and settlement services.

Coupled with valantic FSA's deep experience of implementing and adapting solutions to precisely fit the customer's needs, while deriving greater levels of standardization and speed-to-market, x-gen provides an opportunity to quickly transform business processes in a world of ever-increasing complexity.

valantic

About us

valantic FSA automates the trading and transaction workflows at more than 100 firms in the Financial Services industry.

Our mission is to digitize, augment and evolve the value streams within our clients. This delivers new levels of efficiency, insight, and agility so that our clients can position themselves for maximum impact today and in the future.

Our deep industry expertise is used to assemble these systems from a broad range of proven components and next-generation technologies.

valantic FSA

Smarter automation driving the financial industry.

www.valantic.com/fsa
www.linkedin.com/company/valantic-fsa