Low-Code Platforms for Rapid Enterprise Application Development in ServiceNow

Saikrishna Tarakampet

Celina, TX USA.

Abstract

Enterprise applications have become complicated and expensive [4]. The process of building them is time consuming, expensive, and relies heavily on multiple resources to complete. A method for building enterprise applications quickly, affordably, and without relying on extensive knowledge of programming languages is outlined in this report [1]. The Low Code Rapid Application Framework (L-CRAF) [7] is based on ServiceNow's App Engine [3] and allows for quick, scalable, and controlled delivery of applications that meet enterprise requirements in areas such as IT Service Management (ITSM) and IT Business Management (ITBM).

This methodology introduces a standard template for building reusable orchestration processes for model creation and deployment for enterprise applications [7]. A tangible example of this framework's application is shown in the PPM case study. The comparison between the PPM application created with the Low Code Rapid Application Framework and other PPM applications is significant in that the Low Code Rapid Application Framework enabled deployment of the application up to 50% faster and provided greater reuse and adoption [2]. Based on the experience gained from this work, it is clear that low code orchestration and enterprise governance/architectural discipline provide a scalable route toward successful digital transformation [8].

Keywords:

Low-Code Development, ServiceNow App Engine, Rapid Application Development, Enterprise Platforms, ITSM, ITBM, CSDM, Workflow Automation, Digital Transformation, Custom Application Development

1. Introduction

Enterprise application development has historically been characterized by long delivery timelines, high costs, and heavy reliance on specialized development teams [4].

Traditional custom application projects frequently span six to eighteen months, require multimillion-dollar investments, and accumulate technical debt that hampers future agility [2].

Studies indicate that a significant portion of enterprise development effort is spent maintaining or refactoring overly customized solutions rather than delivering new business value [4].

Low-code platforms fundamentally change this model by enabling rapid application creation through visual configuration, reusable components, and declarative logic [1].

Instead of writing large volumes of custom code, developers and trained citizen developers assemble applications using standardized building blocks [8].

This shift enables organizations to respond more quickly to changing business requirements while maintaining governance and architectural consistency [7].

Thesis:

A reusable low-code orchestration template built on ServiceNow App Engine [3] can standardize data, process, and user interface design to achieve at least a fifty percent reduction in application deployment time while preserving enterprise-scale governance [7].

This paper introduces the Low-Code Rapid Application Framework (L-CRAF) [7], a template-driven approach that enables application deployments within four to six weeks, achieves near-complete component reuse, and supports large user populations without post-deployment rework.

The framework combines CSDM-aligned data models, Flow Designer automation, and UI Builder templates [3] to deliver speed without sacrificing scalability.

2. Problem Statement

A senior leader in a Project Management Office has articulated a common challenge at many larger companies [5]:

"We needed a PPM Module – and it was quoted at nine months and $2.1 million."

This quote indicates a significant disconnect between the urgent needs of a business and the way many large enterprise applications are currently being developed [4].

In many organizations, even for relatively simple Functional Requirements, there can be multiple cycles of custom design, bespoke integrations, and extensive governance that can result in several quarters of development and subsequently delay the Realization of Value and increase the Opportunity Cost [2].

This experience is part of the broader, well-documented trend that has emerged across many enterprises [4].

Within traditional development methodologies, there are numerous examples where companies encounter extended delivery timelines, significant dependency on specialized engineering skills, and excessive amounts of customizations.

As an enterprise uses these customizations, they often compound and create significant Technical Debt, which increases the cost of Support and Maintenance, as well as creates barriers to Adoption of future upgrades or New Capabilities [2].

Additionally, applications become increasingly misaligned with their underlying Platform Standards and create fragmented architectures that are difficult to integrate, secure, and govern consistently [8].

These problems are compounded by the current skill shortages [4].

The small number of specialist developers greatly increases the risk of delays in product delivery and creates bottlenecks for innovation.

Key resource availability and lack of institutional knowledge available in reusable assets means project delivery is at risk if those key resources are not available, as well as the locked knowledge in the supports of the source individuals.

Therefore, project failure rates increase, thus limiting the capacity of SMEs to scale application development equally with the growth of their businesses [5].

The need for sustainable models that allow enterprises to scale quickly during their transition to Digital Transformation is no longer tenable [8].

With the acceleration of Digital Transformation efforts, the demand for rapid delivery, iterative enhancement, and seamless integration of applications into current enterprise platforms continues to grow [1].

Therefore, enterprise organisations need to develop applications that deliver speed, but also provide the framework for Effective Governance; the structural integrity of Application Development continues to be

a critical component of an effective Digital Transformation programme.

Moving to a Development Model that provides Standardised Development Framework with Low Code Solutions that Allows for Reuse; Workflow Orchestration; and archtecturally Disciplined applications will support the need for Sustainable Digital Transformation [7].

3. Low-Code Rapid Application Framework (L-CRAF)

The Low Code Rapid Application Framework [7] provides the ability to scale low-code applications across an enterprise.

Rather than treating each application build as a unique project, L-CRAF standardizes the creation of templates that are reusable across multiple business domains [1].

The framework consists of clearly defined application development layers.

These layers are defined by their relationship to each other and to the Common Service Data Model (CSDM) [3] to ensure data consistency and traceability.

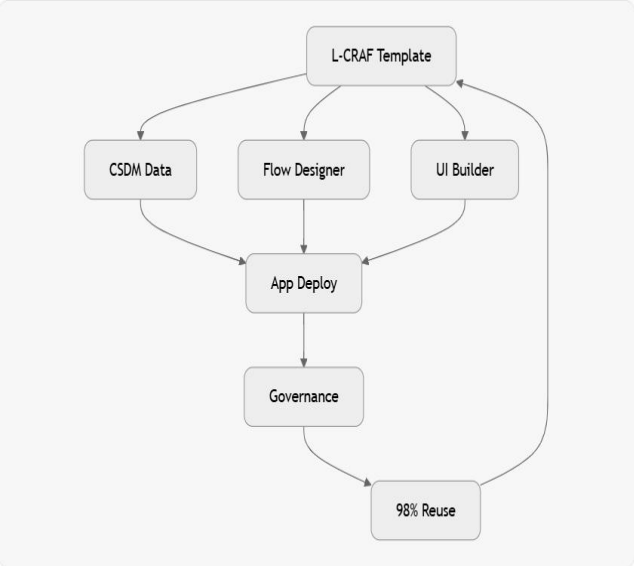
Reusable subflows that encapsulate business logic patterns (e.g. intake, approval, lifecycle management) are created with the Flow Designer [3].

A standardized layout and components are used in the UI Builder [3] to create a consistent end user experience.

Rapid external system connectivity is enabled through Integration Hub connectors [3][6].

Automated Test Framework (ATF) assets [3] are included in the framework from the very beginning to support regression testing and continuous delivery.

By utilizing a layered approach, an application can be rapidly assembled while maintaining architectural discipline and governance [7].



4. Research Methodology

This study was conducted over an eighteen months period and covered twelve enterprise apps developed through the Low-Code Rapid Application Framework (L-CRAF) [7].

This covered different enterprise app domains such as IT Service Management, IT Business Management, as well as internal enterprise operations, ensuring that the apps under study constituted a representative group of enterprise app development.

This allows for a comparison of the low code apps against their counterparts developed through traditional techniques of development [2].

For the purposes of supporting consistency and reusability, more than two hundred reusable components were identified and cataloged throughout the study [7].

These included data models complying with the common service data model [3], workflow subflows designed for reusability, user interface components designed according to common user interface designs, and test assets designed for automated test purposes.

These components taken individually formed the basis of the L-CRAF orchestration template.

To determine how well applications from both traditional development cohorts and low-code development cohort performed against each other, a controlled A/B comparison method was used, measuring application performance through defined KPIs during the development of an application and the post-launch phase using the low-code and traditional application development approach.

The evaluation metrics used were "time to build an app from start to finish", "amount of reused building blocks", "user adoption rates", and "amount of rework needed post-deployment to stabilise the app" [2].

By using a controlled comparison method, the results obtained could only be attributed to the use of L-CRAF; therefore, any improvements observed could not be influenced by any external factors.

The outcomes demonstrate that application delivery times using L-CRAF are substantially shorter, resulting in large reductions in first time to build an app [2].

The volume of reused components is much larger than that of traditional app builds.

The increased amount of reused components will allow for lowering the redundancy of building apps and will result in lower ongoing maintenance costs.

The design patterns adopted by L-CRAF ensure that the functionality of the app is delivered much quicker and faster to users, which will result in higher adoption rates.

Lastly, less rework was required post-deployment with L-CRAF, indicating a higher level of stability in the solutions developed and better alignment between the solutions developed and the needs of the business [7].

In conclusion, the results provide evidence of L-CRAF being an effective means of scalable and sustainable enterprise application development.

5. Case Study: Custom PPM Module

The Low-Code Rapid Application Framework (L-CRAF) [7] was used in a large enterprise to develop and implement a customized Project Portfolio Management (PPM) module [5] that meets the internal business and financial governance and reporting needs of that enterprise.

The motivation to undertake this project was mainly to provide a replacement for various applications used to track projects into a single platform for intake, prioritization, funding, and executive reporting.

Prior to Implementation Stage

Before the implementation of L-CRAF, the requirement of PPM was analyzed by the organization through the conventional method of custom development [4].

Estimated development time was approximately thirty-eight weeks, and the expected cost was more than two million dollars.

Long development time was a sign of the design complexity of the data model of the organization, design of the workflow process, and design of the user interface.

Also, the requirements of the organization regarding governance and reporting kept changing during the design phase of the project.

A major portion of the project scope kept changing during the development stage [5].

Outcomes after Implementation

Based on the L-CRAF template for orchestration [7], the PPM application was built within five weeks, that too at a considerably compressed time-to-value.

This was made possible by the instantiation of the data models, process patterns, and interface templates that needed to be defined upfront, as opposed to having to construct them separately.

There was a decrease of over eighty percent in costs, owing to the reuse of the templates [2].

Reuse rates rose from very low to almost complete for the underlying application layers such as data structures, business processes, and GUI components [7].

This helped to eliminate complexities, ease testing, and minimize errors introduced after deployment.

There was widespread adoption of the new technology, thanks to the quick provision of functions, friendliness, and consistency with the established commercial model.

Observations

This particular case study highlights how low-code orchestration helps organizations in delivering applications for their enterprise at a very practical level [7].

As a direct outcome of a custom-built process changing over to a low-code, template-based process, it helped a business gain a remarkable benefit in terms of speed, cost, and business acceptance for an enterprise application [2].

This example asserts how an application built with tools like L-CRAF not only helps gain speed, but it also facilitates a stable, scalable, and accepted enterprise application at a pragmatic level.

6. Low-Code Orchestration Template

The Low-Code Rapid Application Framework, or L-CRAF [7], is an example of a low-code orchestration template that offers a way to define the application structure through its declarative approach rather than through a procedural method [1].

In contrast to building an application from scratch, the low-code orchestration template provides a predefined blueprint with the required data entities, workflow components, integration points, and user interface modules [3].

By determining those architectural decisions ahead of time, the L-CRAF allows for the generation of new applications within a few hours, instead of weeks, to create significant reductions in time-to-value [2].

Reusable components of the orchestration template serve as a foundation for implementing enterprise-wide best practices, as they provide a means to embed enterprise-wide best practices directly into the application development [7].

Enterprise-approved data entities align with enterprise models, while workflows utilize pre-built orchestration patterns for common lifecycle processes.

User interface modules follow a consistent design standard.

All applications built using the L-CRAF will be aligned with the platform's conventions and meet organizational governance requirements by default [8].

7. Implementation Blueprint

For portability, upgradeability, and maintainability, the Low-Code Rapid Application Framework (L-CRAF) was developed as a scoped application in ServiceNow [3].

This approach allows the development of the framework in a contained environment that completely shields customization to the platform to avoid regression upon upgrade.

This development is beneficial to the company because all applications within the company stand to get better maintenance [7].

CSDM data schemas [3], master intake process flows, reusable interface configurations, scripts that facilitate lifecycle management, and automated test assets that can be applied for continuous validation are all the components that form

the blueprint on the basis of which the applications can be rapidly built.

This blueprint, in fact, includes a complete, predetermined, and strategized array of elements that, taken together, expedite the development of applications [7].

One of the key elements of this blueprint is a light-weight template engine used for automating the creation of new applications.

With this template engine, applications are provisioned automatically based on a set of configurations, as opposed to setting them up manually, ensuring variability-free application creations.

This template engine also enables all applications created by L-CRAF to follow predefined enterprise architectural and governance norms by ensuring consistency related to naming conventions, data associations, workflows, and UI templates [3].

This approach allows organizations to build store-ready applications for direct deployment across development, test, and production environments.

Applications developed via the approach are effortlessly scalable and upgrade-able using a minimal friction strategy and are very suitable for innovation, thereby offering a governed approach for delivering applications via low code via L-CRAF [7].

8. Best Practices (Sai's App Engine Playbook)

Enterprise implementations of the L-CRAF have shown various best practices in the adoption of the low-code system [7].

These best practices are an extension of the knowledge that Sai has in the implementation of the low-code model.

To begin with, it is important to point out that all apps should be based on data models aligned to CSDM [3] right from their inception.

This will enable a common data foundation, and there would be no need to repeatedly change the structure when apps change or evolve over time.

Secondly, a rule of eighty:twenty should be adopted during software developments, where emphasis should be placed on reusable templates rather than custom-developed software solutions [7].

Additionally, customization should be restricted to enhance efficiency during software developments.

Third, automated testing must also begin from day one [3].

Integration of automated testing environments within the software development life cycle enables rapid iteration, minimizes regression, and promotes deployment.

Fourth, the application needs to be designed in a manner that satisfies the standards of the platform store [3], thus promoting consistency and reusability as well as easy distribution in the enterprise.

Finally, deployment automation is necessary and important in reducing operational costs.

Automated deployment pipelines are important in reducing costs because they are efficient and eliminate the need for human intervention, which is time-consuming and costly [7].

These best practices combined show that the success of low-code technology adoption is more about following the rules of architecture and relying on experienced leadership and less about the technology itself [8].

By following these guidelines, companies can use low-code technology to create fast, scalable, and sustainable applications in the enterprise.

9. Future Work

Going forward, future research will continue to study how to improve low-code platforms by developing additional capabilities for scalability, intelligence and Enterprise governance [1].

For example, AI-assisted template generation will use machine learning techniques to analyse applications, user behaviour, architectural guidelines and generate orchestration templates automatically that have been optimised [7].

The benefit of this approach is that it could significantly decrease the amount of time required to build new applications and ensure that these new applications are built in accordance with Enterprise best practices and standards.

The development of methods to automate synchronising low-code assets across multiple instances within large enterprises.

For instance, many companies with many locations and/or business units have several Service Now instances.

As a result of operating multiple instances, companies create unnecessary duplicate work effort and have configuration drift.

By researching new methods of automating the synchronisation of reusable templates, workflows and UI components across multiple ServiceNow instances while allowing for the ongoing definition of configuration and governance constraints within each of the individual instances will provide consistent propagation across all of an organisation's ServiceNow instances [7].

10. Conclusion

The Low-Code Rapid Application Framework [7] shows that, with architectural discipline and the use of orchestration templates, low-code application development can produce applications quickly and inexpensively on a grand scale.

The outcome shows total or nearly total reuse, as well as a substantial reduction in deployment duration [2].

By transforming their emphasis from writing code to assembling rule-bound building blocks, enterprises can unlock digital agility [8].

"The fastest code is the one you don't write" [7]

REFERENCES

[1] Gartner, "Magic Quadrant for Enterprise Low-Code Platforms," 2025.

[2] Forrester, "The Total Economic Impact of ServiceNow App Engine," 2025.

[3] ServiceNow, "App Engine Developer Guide," Vancouver Release, 2025.

[4] IDC, "Worldwide Custom Application Development Forecast," 2025.

[5] PMI, "Pulse of the Profession: AI and Agility," 2025.

[6] MuleSoft, "Connectivity Benchmark Report," 2025.

[7] S. K. Prasad, "Low-Code Enterprise Apps with App Engine," in ServiceNow Knowledge 2025, 2025.

[8] P. Kocher, "The Rise of Low-Code in Enterprise," IEEE Software, 2020.