Challenges

in Legacy Conversion

Security Vulnerabilities

40% of organizations have reported security breaches linked to outdated legacy systems.

Escalating Maintenance Costs

Legacy systems consume up to 60% of IT budgets, limiting resources for innovation.

TRANSONS

| Standard | Standa

Integration Complexity

62% of IT leaders identify integration with modern applications as a top challenge.

Insufficient Documentation

Over 70% of legacy codebases lack adequate documentation, complicating maintenance efforts.

Opensource.com

ARTICLES RESOURCES © DOWNLOADS FREQUENTLY ASKED QUESTIONS

How I avoid breaking functionality when modifying legacy code

Extract methods give the biggest bang for the buck when it comes to modifying legacy code while avoiding the risk of breaking the functionality.

By Alex Banardic (Alume)

July 5, 2021 | O.Comments. | 7 min read

Performance Degradation

Organizations face performance issues with legacy applications, leading to up to 50% degradation in efficiency.

Talent Shortage in Legacy Technologies

An estimated 40% of COBOL programmers are expected to retire in the next decade, exacerbating the talent gap.

Challenging to implement a new testure or fix a bug wethout introducing further complications.

Leck of Documentation

A common issue with legacy codebases is inadequate or outstated documentation. Over time, the original developers may leave the organization, and the remaining documentation may not reflect the current state of the system. This lack of documentation may not reflect the current state of the system. This lack of documentation makes it difficult for new developers to understand the code hope, dependencies, and functionality. This is where documentation ecroes into play, and today, using tools like swimms this can be done automatically and easily.

Onvious hoesingle in the header system with making a documentation. New developers toaked with adding a new repurring feature would struggle to identify where and how to integrate their changes without causing diruptions, as they lack insight into the system's existing structure and data flows. This scenario infen leads to increased onlosseding time and a higher risk of introducing errors.

Tightly Coupled Architecture

Legacy systems often have tightly coupled components, meaning changes in season of the seatom can also allossed. This scenario infendent, most others. This sketch consider.

@ Go to source

Go to source

Go to source

Features

Gen Al-Driven Code Analysis
 Optimizes legacy code using advanced generative Al

Al-Powered Documentation

Automatically generates accurate documentation

Flow Diagram Generation

Creates visual flow diagrams for system processes

Automated Test Case Creation
Develops deployable test cases for functionality validation

Cloud-Native Code Conversion

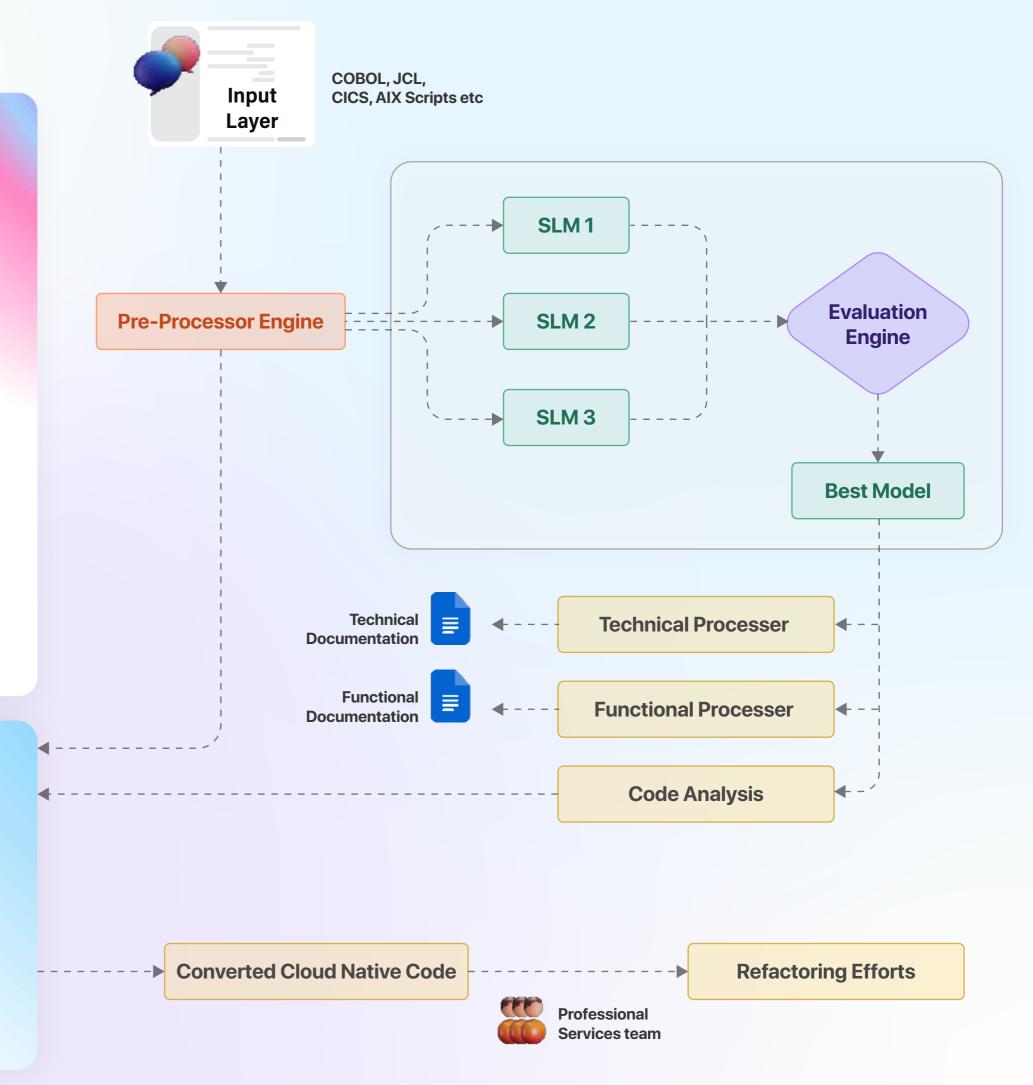
Transforms legacy code to modern cloud-native architecture

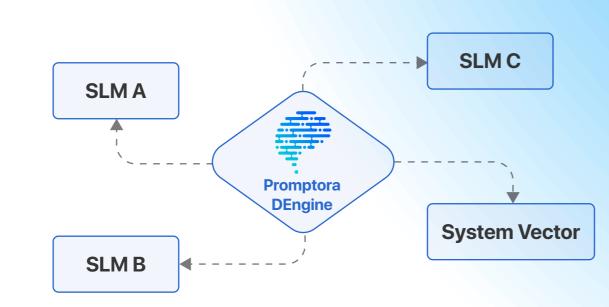
Secure Al Models

Utilizes secure, containerized Gen Al for data protection

Intuitive User Interface
Simplifies the code upload and transformation process

Integrates easily with existing systems and applications





Benefits

AI-Powered Documentation

lower pricing compared to traditional market tools, reducing overall modernization expenses

Faster go-to-market time

by automating code modernization processes.



Improve documentation quality and speed

ensuring comprehensive and accurate project records

Feature	Our IP	Traditional Market Tools
Documentation Generation	Generates comprehensive documentation automatically, ensuring accuracy and reducing manual effort.	Documentation is often outdated or incomplete, requiring significant manual input from developers
Code Optimization	Automatically refactors and optimizes legacy code using generative AI, enhancing performance and maintainability	Manual refactoring is often required, leading to higher chances of errors and inefficiencies
Adaptability	Adapts to diverse datasets and learns patterns for flexibility in code generation and modernization	Rigid and rule-based, requiring frequent updates to accommodate new requirements or technologies
Testing Automation	Automates test case generation and maintenance, ensuring reliability with less manual intervention	Testing processes are typically manual and time-consuming, increasing the risk of undetected issues
Integration Capabilities	Seamlessly integrates with existing systems, facilitating smoother transitions to modern architectures	Integration can be cumbersome and often requires extensive manual coding efforts
Security Enhancements	Utilizes AI to identify vulnerabilities and suggest secure coding practices proactivel	Security measures are often reactive, addressing issues only after they arise due to outdated practices
Cost Efficiency	Reduces overall modernization costs by automating processes and minimizing the need for extensive developer resources	Higher costs associated with manual processes and prolonged project timelines due to inefficiencies