Request for Information (RFI)

Subject: AI Tools for Software Refactoring and Rearchitecting

Issuing Organization: PEO STRI

Contact: Mr. Brian Kemper

PEO STRI is seeking information about methods to assess and rapidly refactor and rearchitect existing software systems into modern, open modular, and cloud-ready systems ostensibly utilizing commercially available and emerging AI-powered tools and processes. We are particularly interested in tools that can analyze, assess, and recommend improvements for achieving Modular Open System Approach (MOSA) objectives, embracing innovation, enabling parallel program development, supporting rapid deployment and achieving high composability.

Background:

The Project Lead Enterprise Transformation & Integration (PL ETI) office holds the critical responsibility of enterprise-wide portfolio planning for software development and DevSecOps among other transformation strategies. PL ETI is tasked with establishing a well architected and integrated supersystem of authoritative shared modular software elements while eliminating redundant software elements . A key aspect of this mission involves guiding the modernization of software design practices and ensuring alignment with industry best practices and architectural paradigms like Microservices and Cloud native patterns

The current landscape of software systems within the organization presents both opportunities and challenges. While PEO STRI has been a leader in product line software strategies within the army, there are areas where legacy software and architectural complexities hinder re-use, scalability, and the adoption of modern development practices. To address these challenges and achieve the strategic objectives set forth by PL ETI, a comprehensive modernization effort is required.

This modernization effort hinges on empowering development teams with disruptive tech such as AI. AI-powered tools specifically designed to analyze, refactor, and guide the re-architecting of existing software systems are identified as integral components for achieving enterprise-wide portfolio modernization objectives. These tools hold the potential to significantly accelerate the modernization process, mitigate risks, and ensure alignment with the desired architectural vision.

Target Architecture

Our goal is to achieve a modern, flexible, and robust software architecture that adheres to the following key principles:

- **Parallel Development:** Supports large software environment where concurrent development of multiple projects or capabilities with minimal interdependency.
- **Innovation & Freedom:** Empowers developers to explore and rapidly integrate cutting-edge technologies.
- **Autonomy & Reduced Dependencies:** Minimizes reliance on external resources and third-party dependencies outside the project's control.
- **Simplified Development Workflow:** Reduces the complexity of code merging, minimizing merge conflicts and associated risks.
- **Seamless Data Distribution:** Ensures efficient and reliable distribution of data model updates across the system.
- **Isolated Changes & Minimal Impact:** Isolates the impact of changes within individual projects or components, minimizing ripple effects across the system.
- **Rapid Deployment:** Enables rapid and frequent deployment of new features and capabilities with minimal risk to the operational system, even while the system is running.
- **Portability & Diverse Environments:** Supports easy portability to new and diverse hosting environments (cloud, on-premise, hybrid).
- **High Composability:** Allows for flexible assembly and re-assembly of components to create tailored solutions.
- **Diverse Hosting Solutions:** Supports deployment across a variety of hosting solutions to accommodate emerging technologies and deployment models.

Target Architecture Design Features:

In addition to the above principles, the target architecture should exhibit the following design features, aligning with MOSA principles:

- **Service Autonomy:** Services are designed to operate independently with minimal reliance on other services, promoting fault tolerance and independent scalability. (MOSA: Service Loose Coupling)
- **Service Loose Coupling:** Interactions between services are minimized and standardized, reducing interdependencies and facilitating independent development and deployment. (MOSA: Service Loose Coupling)

- **Independently Deployable:** Services can be deployed, updated, and scaled independently without affecting other parts of the system, enabling rapid iteration and continuous delivery. (MOSA: Service Composability**)**
- **Service Granularity:** Services are decomposed into appropriately sized units of functionality, balancing between cohesiveness and manageability. This promotes agility, reusability, and efficient resource utilization. (MOSA: Service Abstraction)
- **Model-Driven Design:** Leveraging models to represent business logic, data structures, and service interactions, promoting interoperability and platform independence. (MOSA: Model-Driven Architecture)
- **Reusability:** Services are designed for maximum reusability across different applications and contexts, reducing development time and effort. (MOSA: Service Reusability)
- **Discoverability:** Services are easily discoverable through a service registry, allowing for dynamic service composition and integration. (MOSA: Service Discoverability)

Scope of Information Requested

This RFI requests information on AI tools and associated methodologies that address the following aspects of software refactoring and rearchitecting, considering the key principles and design features outlined above:

1. Code Analysis and Understanding:

- **Automated code analysis:** Identify code smells?, technical debt, and potential bugs, particularly those hindering parallel development, rapid deployment, service autonomy, and loose coupling.
- **Architecture visualization:** Generate visual representations of existing architecture and dependencies, highlighting areas of high coupling and potential for decoupling to achieve service autonomy and granularity.
- **Code comprehension assistance:** Help developers understand complex codebases and identify areas for improvement to achieve greater autonomy, reduce dependencies, and design to enable automatic deployment.

2. Refactoring Recommendations and Automation:

- **Intelligent refactoring suggestions:** Provide actionable recommendations for code refactoring based on best practices and design patterns, specifically those that promote the target architecture's principles and design features.

- **Automated refactoring execution:** Automate the process of applying refactoring actions, ensuring code consistency and correctness while minimizing manual intervention and risk.
- **Refactoring impact analysis:** Predict the potential impact of refactoring on code functionality and performance, especially concerning potential conflicts with parallel development efforts and service autonomy.
- **Automated documentation:** Provide accurate documentation with associated refactored code elements in order to support future analysis, discoverability, and modernization efforts.

3. Architectural Modernization:

- **Microservices identification and extraction:** Identify candidates for microservices extraction from monolithic architectures to achieve high composability, rapid deployment, and support for diverse hosting solutions, aligning with the desired service granularity and independence.
- **Cloud migration assessment and planning:** Analyze existing architecture and recommend optimal cloud migration strategies that align with the desired principles of portability, diverse environments, and support for independently deployable services.
- **Technology stack modernization:** Suggest suitable modern technologies and frameworks for re-architecting, considering their ability to support innovation, developer freedom, service autonomy, loose coupling, and integration with cutting-edge technologies.

4. Tool Capabilities and Requirements:

- **Supported programming languages and frameworks:** List the programming languages and frameworks relevant to our current technology stack and future innovation goals.
- **Integration with development environments:** Describe how the tool integrates with popular IDEs and development workflows to ensure seamless adoption and minimize disruption.
- **Scalability and performance:** Provide information on the tool's scalability and performance when analyzing large codebases, as this is crucial for supporting our development scale and parallel development efforts.
- **Deployment options:** Describe the available deployment options for the tool (cloud-based, on-premise, etc.) and their compatibility with our desired hosting solutions.

- **Pricing and licensing model:** Provide details on the tool's pricing and licensing model to assess affordability and return on investment.

5. Case Studies and Success Stories:

- Share relevant case studies or success stories where the tool has been successfully used for software refactoring or rearchitecting projects, demonstrating its ability to achieve the key principles and design features outlined in this RFI.
- Provide quantifiable results and benefits achieved through the use of the tool, particularly in terms of improved development speed, reduced risk, increased innovation, and successful implementation of service-oriented design principles.

6. Submission Instructions

Responses should be submitted electronically to [Email Address] and should include the following:

- Company name and contact information
- Detailed description of the AI-powered tool and its capabilities, specifically addressing how it supports the outlined key principles and design features.
- Relevant case studies and success stories, highlighting quantifiable results and benefits achieved.
- Provide potential contract vehicles available to assess, leverage, procure the AI tool and associated support as well as any available pricing and licensing information.
- **Response Format:** Respond in either MSWord or PDF format.
- **Submission Method:** Respond via email to: _____
- **Contact Person:** Brian Kemper (brian.e.kemper.civ@army.mil).
- **Deadline for Submission: 30 APR**
- **Page Limits:** No more than 5 standard, 8.5 x 11 pages.
- **Confidentiality:** Clearly state in the response if the information is proprietary to your company. No responses will be accepted that requires
- a Non-Disclosure Agreement (NDA).
-

We appreciate your time and look forward to receiving your response.

7. NEXT STEPS

Following the review of RFI responses, we may invite selected vendors to participate in more detail technical discussion and industry collaboration sessions on the topic.

Disclaimer:

This RFI is issued solely for information and planning purposes and does not constitute a solicitation. PEO STRI reserves the right to not proceed with any procurement or engagement based on the information received.