La forma más segura de sustituir Mainframe Assembler

Hasta finales de la década de 1980, las compañías de seguros, los bancos, las agencias gubernamentales y las aerolíneas no tenían más remedio que realizar sus aplicaciones principales utilizando Mainframe Assembler. Constantemente modificada y ampliada, el conocimiento de su lógica interna se fue perdiendo.

Los antiguos programadores desaparecieron, pero aún se utilizan miles de estos sistemas, difíciles de mantener y un obstáculo para el desarrollo estratégico posterior.

La promesa

¿Cuántas veces ha oído hablar de proyectos Assembler que han fracasado porque la gente se ha creído afirmaciones como "convertiremos el 70% automáticamente" o "nuestros expertos en Assembler en el extranjero resolverán el problema"?

Seamos justos: convertir el código Assembler a un lenguaje informático "real" sin errores es una de las tareas más difíciles de la informática.

Lo sabemos porque lo hemos hecho: la última ampliación de nuestro producto Njema ofrece la solución.

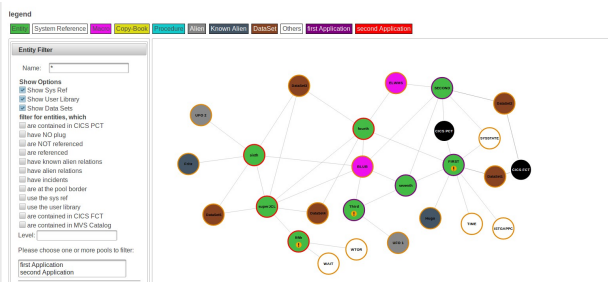
Automatización completa de la transformación Assembler

- **aporta tranquilidad** al proyecto
- **reduce** drásticamente **el riesgo**
- hace que **los costes** sean **predecibles**
- **acorta** significativamente **la duración**
- **hace** que el proyecto sea **independiente** del mantenimiento regular del programa
- **elimina el error humano**
- permite **repetir** la transformación **tantas veces como quiera** sin esfuerzo
- puede **salir a la luz en unas horas o días**

La metodología

Para el éxito del proyecto es esencial conocer de antemano todos los detalles de cada línea del código de la aplicación

- Por ello, Njema examina las macros del sistema y del usuario, las convenciones de codificación y los trucos y hábitos de los programadores. Analiza el uso de los registros, detecta el código automodificable, y guarda todo ello en el en el *repositorio de metadatos Njema*
- Njema también crea un inventario completo de todos los campos de datos, sus propiedades y su relación entre ellos.
- La gran cantidad de datos generados permite utilizar **algoritmos de IA** para **la transformación y el Re-engineering final entre programas**. Esto significa que las reglas de transformación siempre hacen su trabajo teniendo en cuenta todas las demás sentencias de toda la aplicación.
- A partir de este conocimiento, se construye un *Transformation Engine* específico para el proyecto seleccionando cientos de reglas de transformación del *repositorio de reglas Njema.*
- Este Engine transforma todos los programas Assembler en una sola ejecución. Cada ejecución identifica nuevas oportunidades para aumentar el nivel de automatización, que luego se añaden al repositorio de reglas.
- Las situaciones que ocurren raramente o sólo una vez se manejan con reglas singulares.
- Después de la última ejecución, el Transformation Engine ha aprendido a convertir toda la aplicación Assembler en Cobol funcionalmente idéntico: de forma 100% automática.



Las Ventajas

Esta metodología garantiza un riesgo muy bajo y una excelente previsibilidad del proyecto.

- El desarrollo de su Transformation Engine personalizado no interfiere en absoluto con el desarrollo y el mantenimiento habituales.
- La red de relaciones entre programas, macros, campos, estructuras de datos y ficherospuede visualizarse directamente a partir de los metadatos (véase más arriba).

- *Continuous Code Monitoring* garantiza que el código que se transforma esté siempre actualizado.
- Los errores de conversión nunca se corrigen en el código Cobol, sino sólo en las reglas de transformación.

¿Cómo es posible todo eso si no hay documentación?

Bueno, sí existe, la documentación fiable del programa: es el llamado *Object Code* hexadecimal que Njema utiliza como información última.

Sólo el Object Code determina la lógica y el flujo de un programa. Por lo tanto, si se sabe utilizarlo, el programa Cobol generado será siempre la imagen lógica exacta del programa Assembler.

- Una vez finalizadas las pruebas, una prueba de integración garantiza que el código Cobol está listo para la producción.
- El Cobol generado está bien estructurado y es fácil de leer. Así, se puede pasar inmediatamente al mantenimiento regular del programa.

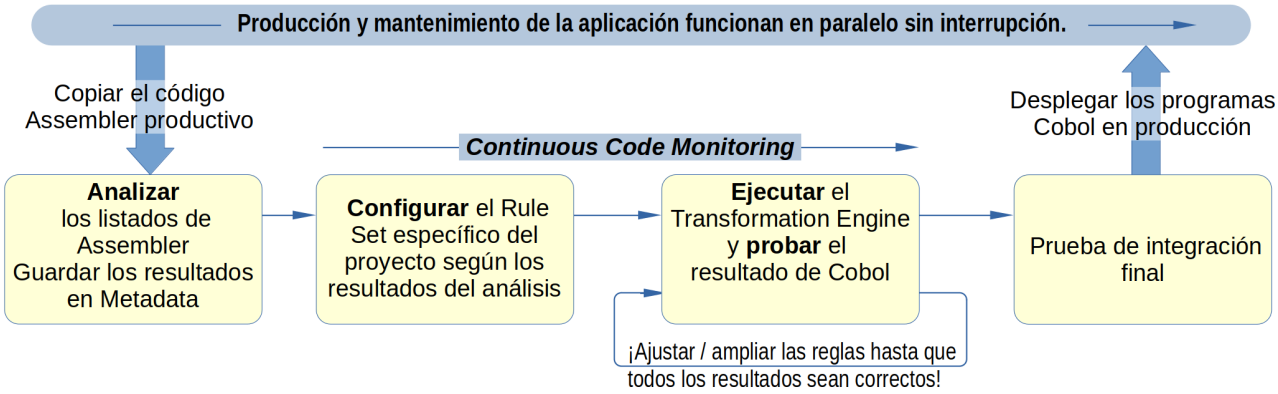
Plataformas compatibles

Actualmente, Njema Assembler Transformation está disponible para z/OS, TPF y VSE.

Condiciones del contrato

ITM ofrece la transformación como un servicio.

Alternativamente, ponemos su Transformation Engine personalizado en la Cloud como un *Continuous-Build-Package.* Esto significa que usted utiliza nuestro producto e ITM lo desarrolla y despliega en la Cloud.



Puede comprobarlo fácilmente, sin compromiso, por qué esta es la forma más segura de reemplazar el Assembler.

El ITM ofrece gratuitamente:

1. transformación de algunos miles de líneas de código fuente

2. análisis de una aplicación completa de Assembler para estimar los costes y la duración del proyecto de conversión