Le moyen le plus sûr de supprimer Mainframe Assembler

Jusqu'à la fin des années 1980, les assurances, les instituts financiers, les autorités gouvernementales et les compagnies aériennes n'avaient d'autre choix que de développer leurs applications de base en Mainframe Assembler. Modifiées et étendues à l'infini, les connaissances de la logique interne de ces systèmes ont disparu. Alors que les programmeurs de la première heure ont pris leur retraite, des milliers de ces systèmes subsistent - difficiles à entretenir et bloquant les mouvements stratégiques.

Les promesses

Combien de fois avez-vous entendu parler de projets Assembler qui ont échoué parce qu'ils ont cru à l'idée du "nous allons convertir 70% automatiquement" ou "nos experts Assembler offshore vont résoudre le problème".

Honnêtement : La transformation sans erreur d'un code Assembler en un "vrai" langage informatique est l'une des tâches les plus difficiles en informatique.

Nous le savons, car nous l'avons réalisée: **la dernière extension de notre produit Njema offre la solution**.

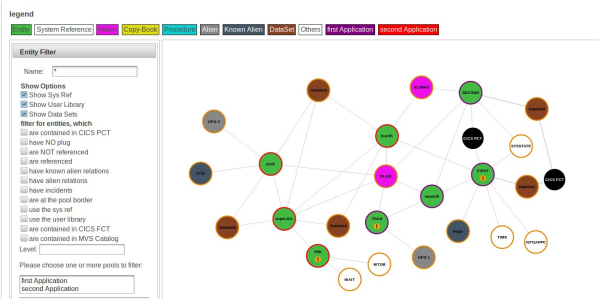
Automatisation complète dans l'Assembler-Transformation

- **apporte de la sérénité** au projet
- **réduit** considérablement **les risques**
- rend **les coûts prévisibles**
- **réduit** considérablement **la durée du projet**
- **libère** le projet de la **dépendance** à l'égard de la maintenance régulière du programme
- **élimine les erreurs humaines**
- permet de **répéter** la transformation sans problème **aussi souvent que souhaité**
- vous permet **d'être opérationnel en quelques heures ou quelques jours**

Méthodologie

La réussite du projet dépend de la connaissance préalable de tous les détails de chaque déclaration d'Assembler dans l'ensemble de l'application.

- Njema examine donc les macros du système et de l'utilisateur, les conventions de codage, ainsi que les astuces et les habitudes des programmeurs. Il analyse l'adressage, l'utilisation des registres, trouve du code auto-modifiant, etc. et stocke tout cela dans le *Référentiel de Métadonnées de Njema*.
- Njema crée également un inventaire complet de tous les champs de données, de leurs propriétés d'utilisation et - très important - de leurs interrelations.
- Cette quantité massive de données permet d'appliquer des **algorithmes d'IA** pour une transformation intelligente et une **ré-ingénierie** finale au-delà des frontières du programme. Ainsi, les règles de transformation font leur travail en tenant compte de tous les autres états de l'ensemble de l'application.
- Sur la base de ces connaissances, un moteur de transformation Njema spécifique au projet est construit en choisissant des centaines de règles de transformation dans le *Référentiel de Règles Njema*.
- Ce moteur est capable de transformer toutes les entités d'Assembler en un seul cycle de traitement. Chaque cycle révèle de nouvelles possibilités d'automation, qui sont ensuite ajoutées au référentiel de règles.
- Les constructions de code uniques qui peuvent apparaître rarement ou une seule fois sont couvertes par des règles particulières.
- Au final, le moteur de transformation a appris à convertir l'application Assembler intégrale en une incarnation Cobol fonctionnellement identique : 100% automatiquement.



La Récompense

Cette méthodologie fournit un flux à faible risque et planifiable du projet de transformation.

- Votre moteur Assembler-Transformation dédié est construit et testé alors que le développement régulier et la maintenance de l'application ne sont pas du tout affectés.

- *Continuous Code Monitoring* permet de s'assurer que le code à transformer et à tester est régulièrement mis à jour par rapport aux versions productives du programme.
- Les erreurs de transformation découvertes lors d'un cycle de test ne sont jamais corrigées dans le code Cobol. Elles sont corrigées dans les règles de transformation.
- L'interconnexion entre les programmes, macros, champs, structures de données et fichiers peut être présentée graphiquement à partir des métadonnées stockées (m. s.).
- Une fois le test final réussi, un Test d'Intégration prouvera que le nouveau code Cobol peut être mis en service, car il correspond exactement à la fonctionnalité des anciens programmes Assembler.

Plateformes supportées

Actuellement, Njema supporte les transformations Assembler pour z/OS, TPF et VSE.

Comment tout cela peut-il se faire en l'absence de documentation ?

En fait, **il existe une documentation fiable** : c'est le code hexadécimal appelé "Object Code" (ou Code Objet) que Njema utilise comme directive ultime.

Le Code Objet est ce qui commande la logique du programme Assembler. Ainsi, être capable de l'utiliser conduira toujours à générer l'équivalent Cobol correct.

- Le code Cobol résultant est bien structuré et facile à lire, de sorte que la maintenance régulière des programmes peut se poursuivre immédiatement avec les programmes Cobol.

Conditions d'utilisation

La transformation est fournie en tant que service.

Alternativement, votre *Modernization Engine* individuel est déployé dans un cloud comme un *Continuous-Build Package*. Cela signifie que vous utiliserez le *Modernization Engine* et qu'ITM en assurera l'extension et la maintenance.