Der sicherste Weg, Mainframe Assembler abzulösen

Bis Ende der 1980er Jahre hatten Versicherungen, Banken, Behörden und Fluglinien keine andere Wahl als ihre Kernanwendungen mit dem Mainframe Assembler zu realisieren. Ständig verändert und erweitert, ging das Wissen über ihre innere Logik mehr und mehr verloren.

Die alten Programmierer verschwanden, aber Tausende dieser Systeme sind immer noch im Einsatz – schwierig zu pflegen und ein Hemmnis für die strategische Weiterentwicklung.

Versprechen

Wie oft haben Sie von Assembler Projekten gehört, die gescheitert sind, weil man Behauptungen geglaubt hat, wie z. B. “wir *werden 70% automatisch konvertieren*” oder “*unsere Offshore Assembler Experten werden das Problem lösen*”.

Seien wir fair: Die fehlerfreie Konvertierung von Assembler Code in eine “richtige” Computersprache ist eine der schwierigsten Aufgaben in der IT.

Wir wissen das, weil wir es gemacht haben: die neueste Erweiterung unseres Produktes Njema bietet die Lösung.

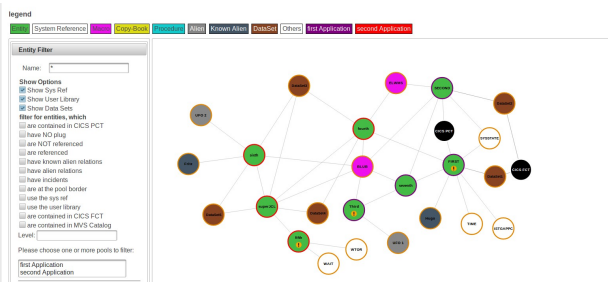
Vollautomation in der Assembler-Transformation

- **bringt Ruhe** in das Projekt
- **verringert das Risiko** dramatisch
- macht die **Kosten vorhersehbar**
- **verkürzt die Dauer** signifikant
- **macht** das Projekt **unabhängig** von der regulären Programmwartung
- **eliminiert menschliche Fehler**
- erlaubt es, die Transformation ohne Aufwand **beliebig oft** zu **wiederholen**
- ermöglicht die **Produktivsetzung in** wenigen **Stunden oder Tagen**

Methodik

Für den Projekterfolg ist es unerlässlich, im Voraus alle Details jeder einzelnen Zeile des Anwendungscodes zu kennen.

- Deshalb untersucht Njema aufgerufene System- und Benutzermakros, Codierkonventionen, sowie Tricks und Angewohnheiten der Programmierer. Es analysiert die Registernutzung, erkennt selbst-modifizierenden Code und speichert alles in das *Njema Metadata Repository*.
- Njema erstellt auch ein vollständiges Inventar aller Datenfelder, ihrer Eigenschaften und ihrer Beziehung zueinander
- Die generierte große Datenmenge ermöglicht es, **KI Algorithmen** für die Transformation und das abschließende **programmübergreifende Re-engineering** einzusetzen**.** Das bedeutet, dass die Transformationsregeln ihre Arbeit stets mit Blick auf alle anderen Statements der gesamten Anwendung verrichten.
- Auf diesem Wissen basierend wird eine projektspezifische *Transformation Engine* aufgebaut, indem Hunderte von Transformationsregeln aus dem *Njema Rule Repository* ausgewählt werden.
- Diese Engine transformiert alle Assembler Programme in einem einzigen Durchlauf. In jedem Lauf werden neue Möglichkeiten zur Erhöhung des Automationsgrades aufgezeigt, die hernach zum Rule Repository hinzugefügt werden.
- Situationen, die selten oder nur ein einziges Mal auftreten, werden durch singuläre Regeln behandelt.
- Nach dem letzten Lauf hat die Transformation Engine gelernt, wie sie die gesamte Assembleranwendung in funktional identisches Cobol umwandeln kann: 100% automatisch.



Vorteile

Diese Methodik gewährleistet ein sehr niedriges Risiko und eine ausgezeichnete Planbarkeit des Projekts.

- Die Entwicklung Ihrer maßgeschneiderten Transformation Engine berührt die reguläre Entwicklung und Wartung in keiner Weise.
- Das Beziehungsgeflecht zwischen Programmen, Makros, Feldern, Datenstrukturen und Dateien kann direkt aus den Metadaten visualisiert werden (s. o.).

- *Continuous Code Monitoring* stellt sicher, dass der zu transformierende Code stets auf dem neuesten Stand ist.
- Konvertierungsfehler werden niemals im Cobol Code korrigiert, sondern nur in den Transformationsregeln.

Wie ist das alles möglich, wenn es keine Dokumentation gibt?

Nun, es gibt sie doch, die **verlässliche Programmdokumentation**: es ist der hexadezimale sogenannte *Objektcode*, den Njema als ultimative Information nutzt.

Nur der Objektcode bestimmt die Logik und den Ablauf eines Programms. Wenn man also in der Lage ist, ihn zu nutzen, wird das erzeugte Cobol Programm immer das genaue logische Abbild des Assemblerprogramms sein.

- Sobald die Tests abgeschlossen sind, stellt ein Integrationstest sicher, dass der Cobol Code produktionstauglich ist.
- Das erzeugte Cobol ist gut strukturiert und leicht lesbar. Somit kann es sofort an die reguläre Programmwartung übergeben werden.

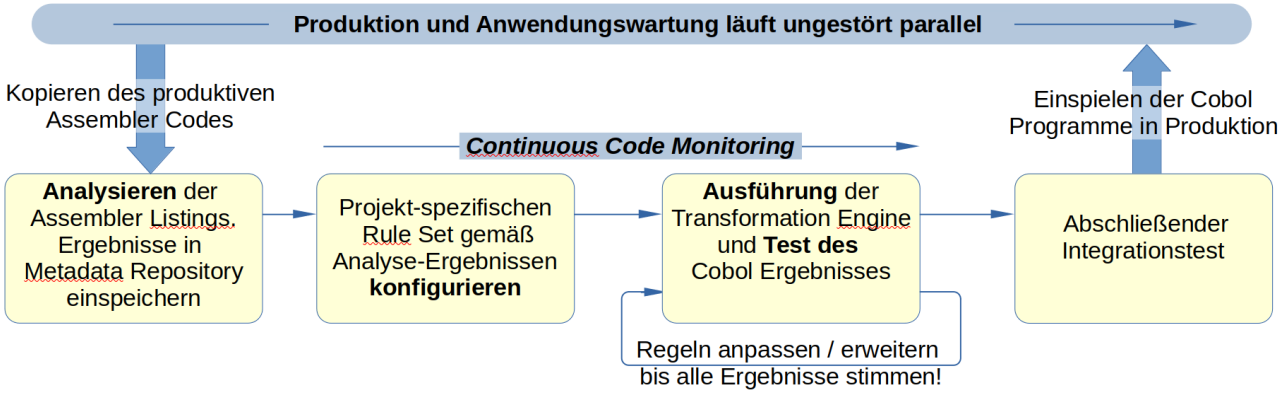
Unterstützte Plattformen

Derzeit ist die Njema Assembler Transformation für z/OS, TPF, und VSE verfügbar.

Vertragsgestaltung

ITM bietet die Transformation als Service an.

Alternativ stellen wir Ihre individuelle Transformation Engine in der Cloud als *Continuous-Build Package* bereit. Das bedeutet, dass Sie diese Engine nutzen und ITM sie in der Cloud erweitert und pflegt.



Sie können einfach und unverbindlich überprüfen, warum dies der sicherste Weg ist, den Assembler abzulösen

ITM bietet kostenfrei:

1. Die Transformation einiger Tausend Quellcodezeilen

2. Analyse einer gesamten Assembler Anwendung
zur Schätzung von Kosten und Dauer des Umstellungsprojekts