Die passenden Klassen und Schnittstellen für den PCMAdapter herauszufinden scheint die Suche nach der Nadel im Heuhaufen zu sein. Gestern habe ich einen sehr kleinen Fortschritt gemacht, der allerdings sehr vielversprechend aussieht. Das Herzstück zum Auslesen ist die Repository-Klasse. Der Zugriff auf deren Inhalte (*seffbranches, loops ...)* stellt sich bislang allerdings als nicht-trivial heraus.

Kurz zum Auslesen der XMI Files (ich habe bislang keine Erfahrungen mit XMI Files gemacht. Das nachfolgende könnte für manche deswegen mehr als selbstverständlich klingen):
Der Aufbau der XMI Files ist ein Baum, dessen Knoten die Bausteine des PCMs darstellen (*components, seffs, branches, interfaces...).* Dabei kann ein und derselbe Knoten-Baustein unterschiedliche Typen besitzen (zB sind die Kinder eines *branch_behaviours step_behaviours*.
Die step_behaviours können u.A. die Typen "*StartAction*", "*ExternalAction*", "*BranchAction*" und "*StopAction*" annehmen.)

Das erste Problem:
Ich weiß nicht, wie ich über diesen Baum sinnvoll traversieren soll. Zunächst habe ich ihn im Breitensuchschema manuell abgeklappert aber mit fehlender Rekursion. Dann ist mir aufgefallen, dass es einen *TreeIterator* gibt der scheinbar mit der next()-Funktion wirklich ALLE Elemente des ganzen Baumes auslesen kann. Vorschläge?

Das zweite Problem:
Alle Objekte, die aus dem Repository als Liste oder Tree herausgegeben werden, sind ziemlich allgemein (EObject). Um damit zu arbeiten, muss ich diese casten. Allerdings ist mir bislang nicht klar, welche dieser Objekte für uns relevant sind (ich müsste diese ja vor dem cast abfangen und nach Klassenzugehörigkeit fragen -> Problem 3). Auf jeden Fall sind die ganzen Seff-Klassen wichtig. Da wir aber einen Baum haben, spalten sich die *Seff*-Knoten weiter auf zu *stepBehaviours*, welche sich zu *Branches* aufspalten, welche sich wieder zu *stepBehavious* aufspalten. Mithilfe von getContents() und der Typisierung/casts kann man an die Konkreten Zweige gelangen, die sich dem analysierten Programmstück entsprechen.
Nun das konkrete Problem:
Ich habe das Gefühl, dass, wenn es keine bessere Möglichkeit zum Auslesen gibt, dieser Part noch extrem aufwändig wird. Vorallem:
- Feststellen welche Klassen für uns relevant sind
- Umsetzung
Ich denke nicht, dass dieser Aufwand in unserer momentanen Phase gerechtfertigt ist. Was meint ihr?

Das dritte Problem (Stilfrage):
Ist es in Ordnung, Klassen von Objekten abzufragen und diese anschließend in spezifischere Objekte zu casten?

Das vierte Problem:
Wie ihr oben seht, habe ich mich so sehr mit Suchen und Problemen befasst, dass die eigentliche Zielsetzung (einen funktionsfähigen Adapter zu kreieren) sich etwas verloren hat. Ich bin mir momentan gar nicht mehr sicher, welche Interfaces ich bereitstellen soll, vielleicht habt ihr Ideen (welche dann wiederum mein wirres Suchen nach Lösungen gezielter machen).

Außerdem:
Wollen wir die wichtigen PCM-Komponenten in einer eigenen Klasse nochmals neu definieren um deren Zugriff zu vereinfachen? (Bei Branches zum Beispiel ein eigenes Interface für Wahrscheinlichkeitswert, Identifier und Mapping zum Code)