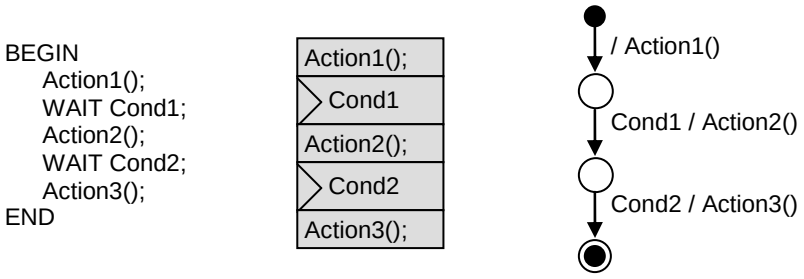
Automatische Transformation von Programmen in Statecharts

Am Christian Doppler Labor für Automated Software Engineering wurde zusammen mit der Firma KEBA eine Domänenspezifische Sprache für die Automatisierungstechnik entwickelt (Monaco - **Mo**deling **N**otation for **A**utomation **Co**ntrol). Monaco erlaubt es, die Hardware-Komponenten einer Maschine durch Software-Komponenten zu modellieren, die jeweils auf Kommandos reagieren sowie Daten liefern und Ereignisse auslösen können. Damit lassen sich Maschinensteuerungen auf natürliche Weise in Software abbilden.

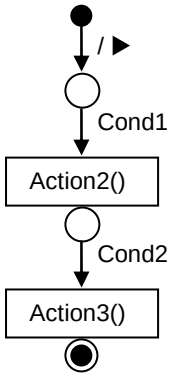
Ein Monaco-Programm ist eine textuelle Darstellung eines Statecharts [HKKR05, Har87]. Es besteht aus Arbeitsschritten (z.B. Berechnungen und Zuweisungen) und WAIT-Anweisungen, in denen es auf ein Ereignis oder eine boolesche Bedingung wartet. Im Zuge des Monaco-Projekts wurde auch eine auf Eclipse basierende visuelle Entwicklungsumgebung (IDE) geschaffen, in der man Monaco-Programme in textueller und grafischer Form editieren kann, wobei die beiden Darstellungen jederzeit umgeschaltet werden können. Als grafische Form wurde eine Notation verwendet, die Ähnlichkeiten zu Struktogrammen hat. Monaco-Programme, Monaco-Struktogramme und Statecharts sind ineinander transformierbar, wie die folgende Abbildung zeigt:



Monaco-Programm *Monaco-Struktogramm* *State Chart*

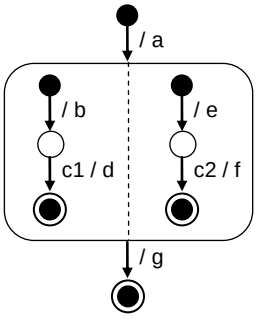
Aufgabe dieses Projekts ist es, die Monaco-IDE so zu erweitern, dass neben Struktogrammen auch Statecharts angezeigt (allerdings nicht editiert) werden können. Dazu sind folgende Aufgaben zu lösen:

1. Statecharts haben das Problem, dass die mit Zustandsübergängen verbundenen Aktionen oft aus mehreren Anweisungen bestehen, die im Diagramm neben den Bedingungen kaum Platz finden. Als Abhilfe könnte man die Aktionen durch ein spezielles Symbol (z.B. ▶) ersetzen, das man anklicken kann, worauf der Text der Aktionen in einem Popup-Fenster erscheint. Durch einen Rechtsklick könnte man das Symbol expandieren, so dass die Aktionen vor dem nächsten Zustand eingeblendet werden:

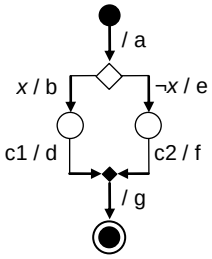


2. Überlegen Sie sich für alle Monaco-Sprachkonstrukte äquivalente Statechart-Notationen, z.B.:

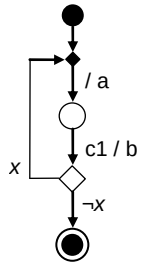
```
a
PARALLEL
    b
    WAIT c1
    d
||
    e
    WAIT c2
    f
END;
g
```




```
a
IF x THEN
    b
    WAIT c1
    d
ELSE
    e
    WAIT c2
    f
END;
g
```

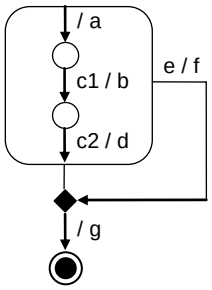



```
LOOP
    a
    WAIT c1
    b
WHILE x
```



3. Monaco-Programme können Blöcke mit ON-Handlern aufweisen. Wenn sich das Programm in einem WAIT-Zustand innerhalb dieses Blocks befindet und die im ON-Handler spezifizierte Bedingung eintritt, so wird der Block verlassen und der ON-Handler wird ausgeführt. Anschließend geht es nach dem Block weiter. Überlegen Sie sich eine adäquate Statechart-Notation dazu, z.B.:

```
BEGIN
    a
    WAIT c1
    b
    WAIT c2
    d
ON e
    f
END;
g
```



4. Studieren Sie die Implementierung der Monaco-IDE und implementieren Sie ein Eclipse-Plugin, das einzelne Monaco-Routinen aus der internen Zwischensprache der IDE in Statecharts transformiert und anzeigt. Achten Sie auf gefälliges und gut lesbares Layout.

Die Arbeit ist in regelmäßigen Abständen mit dem Betreuer zu besprechen. Achten Sie bei der Implementierung Ihres Werkzeugs auf guten Programmierstil und ausführliche Kommentierung, damit es später auch von anderen Personen gewartet werden kann.

Betreuer: Dr. Herbert Prähofer
Beginn: März 2007

Literatur

[HKKR05] Hitz, M., Kappel, G., Kapsammer, E., Retschitzegger, W.: UML@Work. 3. Auflage, dpunkt.verlag, 2005

[Har87] Harel, D.: Statecharts: A Visual Formalism for Complex Systems. Sci. Comput. Prog. 8, 231-274, 1987