Projektauftrag: Ein Framework zur graphenbasierten Modellierung

SS 2015 Prof. Dr. K. Hoffmann

Problemstellung

Hintergrund

Modellierung mit Hilfe von Graphen
Das mathematische Konzepts des Graphen bildet die gemeinsame Grundlage für die Modellierung bei einer Vielfalt von Problemen, etwa - um nur ein paar Beispiele zu nennen – in folgenden Bereichen:

- Beschreibung von Verkehrsnetzen, wie etwa U-Bahn-/S-Bahn-/Bus-Netz einer Stadt oder das Straßennetz einer Gemeinde,
- Kommunikationsnetze, Rechnernetze
- Beschreibung des dynamischen Verhaltens eines zustandsbehafteten SW/HW-Systems (EC-Automat, Steuergerät für den Scheibenwischer in einem Auto, Aufzugsteuerung, etc.): hier verwendet man gerne Zustandsdiagramme oder sog. Petri-Netze.
- Weitere Beispiele aus der Software-Technik liefern der Kontrollflussgraph oder der mit einem Use-Case verbundene Graph (jeder UC-Schritt stellt hier einen Knoten dar)

Mögliche Arten von Graphen
In all diesen Fällen liegt das Konzept eines (endlichen) Graphen zu Grunde: Ein endlicher Graph besteht aus einer endlichen Menge von Knoten und einer ebenfalls endlichen Menge von Kanten; die Kanten werden als Verbindungen zwischen Knoten interpretiert. Man unterscheidet:

- gerichtete Graphen (d.h. jede Kante ist gerichtet): diese Art von Graph spielt etwa bei Zustandsdiagrammen und Kontrollflussgraphen eine Rolle.
- ungerichtete Graphen (keine der Kanten ist gerichtet)
- Mischformen davon (d.h. es dürfen sowohl gerichtete als auch ungerichtete Kanten vorkommen): z.B. bei einem Straßennetz, wo es Einbahnstraßen und „normale“ Straßen gibt).

Darüber hinaus kann es auch sein, dass man in einem Graphen verschiedene Arten von Knoten unterscheiden möchte; bei einem Petri-Netz etwa gibt es zwei Arten von Knoten: sog. „Stellen“ und sog. „Transitionen“. Mathematisch bedeutet das, dass die Menge der Knoten in zwei disjunkte Teilmengen zerlegt („partitioniert“) ist und eine Kante niemals zwei Knoten aus der gleichen Teilmenge verbinden darf (Stichwort „bipartiter Graph“). Die Knoten eines Graphen, der mit einem Use-Case verknüpft ist, sind ebenfalls in verschiedene Mengen eingeteilt: ein Teil der Knoten gehört zum „normalen Ablauf“, andere Knoten gehören zu

verschiedenen Varianten (allerdings hat man hier keine „k-partiten Graphen“ im Sinne der Graphentheorie vor sich – warum?)

Weiterhin können Knoten und Kanten zusätzliche Attribute tragen: einer Kante kann z.B. ein Zahlenwert zugeordnet sein, der meist mit dem Durchlaufen dieser Kante verbundene Kosten bedeutet (im Falle des Straßennetzes könnte etwa die Wegstrecke ein sinnvoller Kostenwert sein). Auch einem Knoten können Attribute zugeordnet sein: bei einem Petri-Netz etwa trägt jeder Knoten, der eine Stelle ist, eine gewisse Anzahl von sog. „Marken“ (engl. „Tokens“). Diese Beispiele sind Sonderfälle sog. Kanten- bzw. Knotenbewertungen . Auch sog. „Färbungen“ für Knoten bzw. Kanten sind Beispiele (ganzzahliger) Attribute. Als Wertebereiche für Attribute kommen aber nicht nur Zahlen in Frage: auch Bezeichnungen, Erläuterungen, etc. sollen z.B. mit Knoten bzw. Kanten verknüpft werden können. Welche Attribute wirklich gebraucht werden, hängt von der Anwendungssituation für den jeweiligen Graphen ab: der Graph für einen Use-Case braucht sicherlich andere Attribute als der Graph für ein Zustandsdiagramm bzw. ein Petri-Netz.

Mögliche Anwendungen der Modellierung mit Hilfe von Graphen
Eine häufige Form der Anwendung besteht in der Lösung von Optimierungsaufgaben; stellvertretend für solch eine Optimierungsaufgabe sei hier die Bestimmung einer sog. optimalen „Briefträgertour“ vorgestellt:

Eine Briefträgertour durchläuft abwechselnd Knoten und Kanten und kehrt an ihren Ausgangsknoten zurück; dabei muss jede Kante mindestens einmal durchlaufen werden. Jeder Kante ist außerdem ein Kostenwert zugewiesen; es gilt eine Briefträgertour mit minimalen Gesamtkosten zu finden[1]. Briefträgertouren können verwendet werden, um z.B. für einen Briefträger eine möglichst kurze Tour durch sein Zustellgebiet zu finden (oder entsprechend für einen Schneepflug eine möglichst kurze Tour durch das Straßennetz einer Gemeinde) oder im Falle eines zustandsbasierten Software-Tests eine möglichst kurze Folge von Zustandsübergängen, bei der jeder Zustandsübergang mindestens einmal durchlaufen wird (Minimierung des Testaufwands unter Einhaltung gewisser Testabdeckungskriterien).

Systemüberblick

Framework: Es soll ein Framework entwickelt werden, das flexibel und allgemeingültig genug ist, um die zuvor erwähnte Vielfalt von Graphen abbilden zu können. Die Tragfähigkeit der Framework-Konzeption soll in der Dokumentation für verschiedene Anwendungsbereiche (Use-Case-Modellierung, Modellierung mit Petri-Netzen, UML-Statecharts) überzeugend dargelegt sein.

Beispielanwendung auf Grundlage des Frameworks:
Auf Grundlage dieses Frameworks soll ein Software-Werkzeug entwickelt werden, mit dem eine in Textform gegebene Use-Case-Beschreibung (z.B. aus einem Word-Dokument[2])

[1] Natürlich muss der Graph gewisse (hier nicht genannte!) Voraussetzungen erfüllen, damit das überhaupt möglich ist.
[2] Hier sollte man vielleicht Word-XML als Dokumentenformat zugrunde legen: man soll mit Word wie gewohnt (und ohne allzu große Rücksicht auf das zu realisierende Software-Werkzeug) Lastenhefte und Use-Case-Beschreibungen verfassen können; durch die Verwendung von XML kann aber das Software-Werkzeug den relevanten Teil des Inhalts des Word-Dokuments „verstehen“.

eingelesen und in den zugehörigen Graphen überführt werden kann. Das Werkzeug bietet zumindest folgende grundlegende Funktionen:

- Visualisierung des Graphen
- Unterstützung bei der Erstellung der Szenario-Matrix: das Werkzeug könnte hier automatisch eine Szenario-Matrix erzeugen, die bei minimalem Testaufwand sicherstellt, dass jedes Szenario abgedeckt ist. Diese Szenario-Matrix kann in (XML-basiertem?) Dateiformat ausgegeben und möglichst mit Word ggf. noch manuell erweitert werden.

Abzuliefernde Arbeitsergebnisse (Produktumfang)

Siehe das Kapitel „Lieferumfang“ im Anforderungsdokument.