Seminar Aspektorientierte Systemprogrammierung

Towards Aspectual Component-Based Real-Time System Development

Matthias Körber

Übersicht

- Motivation
- Eingebettete Systeme & WCET
- Komponentenbasiertes Softwaredesign
- Kriterien für neuartige Designmethoden
- ACCORD und RTCOM
- Zusammenfassung

Motivation

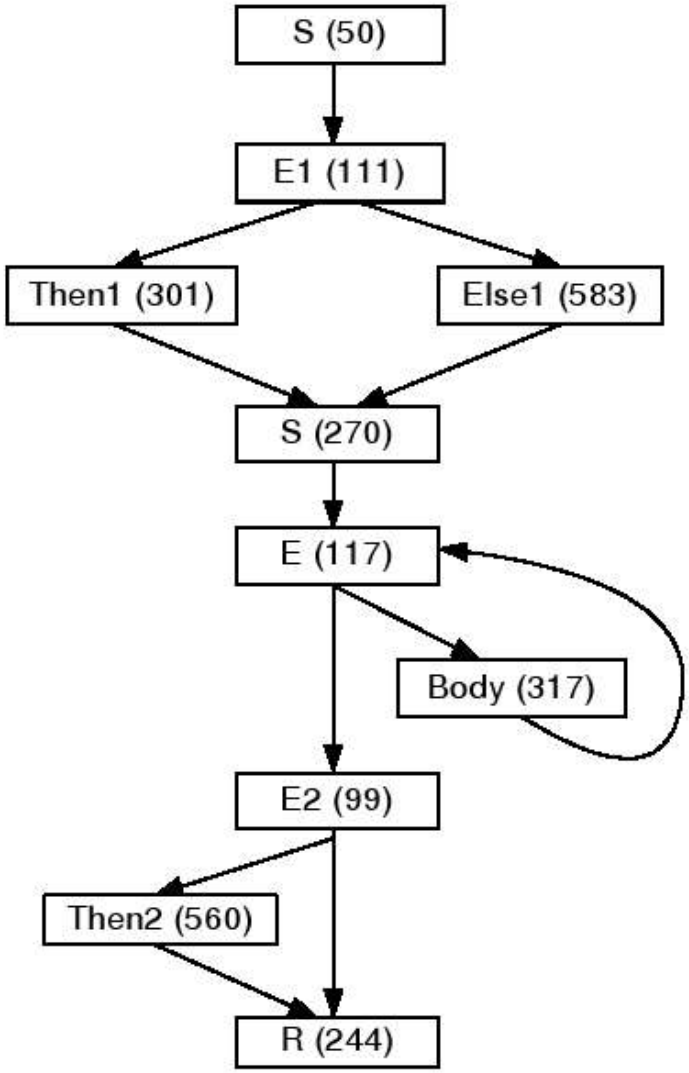
- 98% der weltweiten CPU-Produktion für eingebettete Systeme
- wie bei allen Softwaresystemen:
  - zu lange Entwicklungszeiten
  - zu hohe Entwicklungskosten
  - zu hohe Wartungskosten
- mögliche Lösung: Komponentenbasierte Softwareentwicklung

Eingebettete Echtzeitsysteme

- mehr als nur eingebettete Systeme
- nebenläufige Programme, Tasks
- Unterscheidung zwischen
  - harten Echtzeitsystemen
  - weichen Echtzeitsystemen
- Tasks müssen vorhersehbar sein
- Worst Case Execution Time (WCET)

Worst Case Execution Time - Beispiel

```
1  double power(double fkn, int e) {
2      double res = 0;
3      int times = 0;
4      bool pos = true;
5
6      if e < 0 then
7          times = -e;
8      else
9          pos = false;
10         times = e;
11     end if
12     res = 1.0;
13
14     for(i in 1 ... times loop
15         res = res * fkn;
16     end loop
17
18     if(pos)
19         res = 1 / res;
20
21     return res;
22 end power
```



Worst Case Execution Time

- gesucht ist die worst-case execution time
  für -10 < e < 10
- $WCET_{power} = 50 + 111 + max(301, 583) + 270 + 10(117 + 317) + 99 + max(560, 0) + 244 = 6374$
- manche Pfade schliessen sich gegenseitig aus
- zu pessimistisch

Symbolische Worst Case Execution Time

- Abhängig von Parameter
- dadurch genauere Angabe der Worst Case Execution Time möglich

$$WCET_{power}(e) = 50 + 111 + [e < 0]301 + [e \geqslant 0]583 + 270 + 117 + \sum_{i=1}^{|e|}(117 + 317) + 99 + [e < 0]560 + [e \geqslant 0]0 + 244$$

Symbolische Worst Case Execution Time

- bzw. Vereinfachung zu:

$$WCET_{power} = \begin{cases} 1752 - 434e, e < 0 \\ 1474, e = 0 \\ 1474 + 434e, e > 0 \end{cases}$$

- daraus folgt Maximum bei e = -10
- $WCET_{power}$ = 6092

Komponentenbasierte Softwareentwicklung

- Probleme monolithischer Systeme:
  - hohe Entwicklungskosten
  - schlechte Unterstützung für langlaufende Systeme
  - komplexere Systeme provozieren Qualitätsmängel
- Komponenten ermöglichen:
  - verkürzte Entwicklungszeiten
  - niedrigere Wartungskosten
  - leichtere Erweiterbarkeit von komplexen Systemen

Eigenschaften von Design-methoden für Echtzeitsysteme

- Geheimnisprinzip
- Schnittstellen
- Verschiedene Komponentensichten
- Zeitliche Eigenschaften
- Aufgabenzuweisung
- Aspekt-Unterstützung

Eigenschaften von Design-methoden für Echtzeitsysteme

- Aspektweben
- Mehrere Schnittstellen
- Unterstützung bei der Konfiguration
- Zeitliche Analyse

→ **ACCORD**

ACCORD – Klassifikation von Aspekten in Echtzeitsystemen

- Anwendungsaspekte
- Laufzeitaspekte
- Kompositionsaspekte

ACCORD – Real Time Component Model (RTCOM)

- Echtzeit-Komponentenmodell
- ermöglicht Aspektweben in Komponenten
- besteht aus drei Teilen
  - Funktionaler Teil
  - Laufzeitsystemabhängiger Teil
  - Kompositionsteil

Funktionaler Teil - Definitionen

- Komponenten bestehen aus:
  - Mechanismen
  - Operationen
  - Schnittstellen
- Mechanismus:
  - Funktionen in einer Komponente gekapselt
  - Elementaroperation der Komponente
- Operation:
  - ruft Mechanismen der Komponente auf
  - oder Operationen anderer Komponenten

Funktionaler Teil - Definitionen

- funktionaler Teil einer Komponente besteht aus:
  - Mechanismen
  - Operationen
- Anwendungsaspekt besteht aus:
  - Advice
  - Menge von Pointcuts
- Pointcuts beschreiben Operationen

Funktionaler Teil – Policy Frameworks

- zunächst Entwurf von Mechanismen
- Operationen implementieren mit Hilfe von Mechanismen
- stellt ein erstes Policy Framework dar
- kann durch Aspekte geändert werden

Beispiel – Verkettete Liste

- Komponente enthält Mechanismen
  - createNode()
  - deleteNode()
  - getNextNode()
  - linkNode()
  - unlinkNode()
- Operation listInsert()
  - ruft createNode()
  - und linkNode() auf

Beispiel – Verkettete Liste

- zusätzliche Operationen
  - listRemove()
  - listFindFirst()
- Policy der verketteten Liste ist FIFO
- ändern der Policy durch Aspekt

Beispiel – Verkettete Liste

```
aspect listPriority{
1:
2: pointcut listInsertCall(List_Operands * op)=
3:    call("void listInsert(List_Operands*)")&&args(op);
4:
5: advice listInsertCall(op):
6:    void before(List_Operands * op){
7:      while
8:         the node position is not determined
9:       do
10:        node = getNextNode(node);
11:        /* determine position of op->node based
12:        on its priority and the priority of the
13:        node in the list*/
14:    }
15: }
```

Laufzeitsystem-abhängiger Teil

- WCET von Mechanismen unveränderlich
- Operationen werden durch Aspektweben verändert
  - Anzahl und Art der Mechanismen kann sich ändern
  - beeinflusst zeitliches Verhalten von Operationen

Laufzeitsystem-abhängiger Teil

- Berechnung der WCET trotz Aspektwebens möglich, wenn:
  - WCET aller Mechanismen bekannt
  - WCET einer Operation errechnet sich aus
    - WCET der benutzten Mechanismen
    - interner WCET
  - WCET eines Advices errechnet sich aus
    - WCET der benutzten Mechanismen
    - interner WCET

Beispiel – Verkettete Liste

```
policy(noOfElements){
  operation{
    name listInsert;
    uses{
      createNode 1;
      linkNode   1;
    }
    intWcet 1ms;
  }
  operation{
    name listRemove;
    uses{
      getNextNode noOfElements;
      unlinkNode  1;
      deleteNode   1;
    }
    intWcet 4ms;
  }
....
}
```

```
mechanisms{
  mechanism{
    name createNode;
    wcet  5ms;
  }
  mechanism{
    name deleteNode;
    wcet  4ms;
  }
  mechanism{
    name getNextNode;
    wcet  2ms;
  }

....
}
```

Beispiel – Verkettete Liste

```
aspect listPriority(noOfElements){
  advice{
    name listInsertCall;
    type before;
    changes{
      name listInsert;
        uses{
          getNextNode noOfElements;
        }
    }
    intWcet 4ms+0.4*noOfElements;
  }
....
}
```

Zusammenfassung

- Verknüpfung von Komponentenbasierter und Aspektorientierter Softwareentwicklung
- Unterstützung der Untersuchung von WCETs

Bibliographie

- A. Tešanović, Towards Aspectual Component-Based Real-Time System Development, Linköping University, 2003
- C. Szyperski, Component Software: Beyond Object-Oriented Programming, Addison-Wesley / ACM Press, 1998