Trädfiler i MedView

Olof Torgersson

1 Inledning

I MedView [1] används ett textbaserat datalagringsformat kallat 'trädfiler'. Formatet designades ursprungligen som ett sätt att lagra en stor mängd dokument med gemensam struktur men varierande detaljinnehåll. Varje dokument kan även betraktas som en partiell induktiv definition [3].

I samband med utformandet av trädfilsformatet byggdes även en objektmodell i Objective-C som tillät enklare analys av data utifrån ett definitionellt arbetssätt. Denna låg bl a till grund för 'Kuben' [2]. Även andra arbeten som Gisela [4] och dagens Java-baserade MedView [cite] har fortsatt att använda trädfiler som datalagringsformat. Trots detta saknas en ordentlig beskrivning vilket är orsaken till detta dokument. För att göra detta krävs först en diskussion om de abstraktioner som används.

2 Kliniska undersökningar

I MedView modelleras kliniska undersökningar hierarkiskt som induktiva definitioner. Utifrån grundbegreppet *Undersökning* beskrivs de olika delar en undersökning omfattar. Om en undersökning består av delarna *Patientuppgifter*, *Anamnes*, *Status*, *Diagnos* och *Daganteckning*, ger detta upphov till ekvationerna:

```
Undersökning = Patientuppgifter
Undersökning = Anamnes
Undersökning = Status
Undersökning = Diagnos
Undersökning = Daganteckning
```

Därefter beskrivs de olika delarna varför sig. T ex så kanske `Patientuppgifter` innehåller uppgifter om vem som skickat patienten (`Ref-in`) och varför (`Ref-cause`). Dessutom sparas patientens id-nummer och födelseland. Beskrivningen av detta blir

```
Patientuppgifter = P-code
Patientuppgifter = Ref-in
Patientuppgifter = Ref-cause
Patientuppgifter = Born
```

Beskrivningen så långt är gemensam för alla undersökningar som använder samma 'protokoll'. En specifik instans skapas genom att värden anges för de termer som förekommer i undersökningsbeskrivningen:

```
P-code = G10019221
Ref-in = Tandläkare
Ref-cause = Slemhinneförändring
Born = Sverige
```

Inmatning av data enligt ett visst protokoll sker genom att protokollet läses in i ett därtill anpassat verktyg. Det kan t ex se ut som i figur 1. I princip kan strukturen ha en godtyckligt djup nästningsnivå, även om Java-MedView idag (juli 2004) bara stödjer två nivåer.

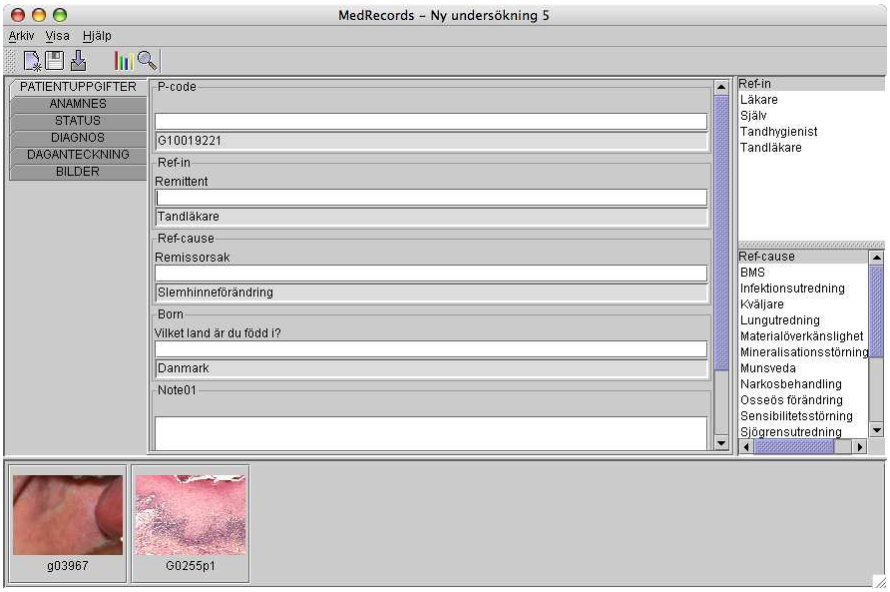

Figure 1: Inmatning av data i MedRecords.

3 Data som träd

Den induktiva definition som skapas då undersökningsdata matas in kan också beskrivas som ett träd med godtyckligt antal grenar för varje nod. I detta fall utgör *Undersökning* trädets rot. I exemplet ovan är `Patientuppgifter`, `Anamnes`, `Status`, `Diagnos` och `Daganteckning`, direkta barn till Undersökning. `P-code`, `Ref-in`, `Ref-cause` och `Ref-cause` barn till `Patientuppgifter` och de undersökningsspecifika värdena (`G10019221`, `Tandläkare`,...) trädets löv.

3.1 Begrepp

I trädfiler används begreppen nod och löv på ett sätt som avviker en aning från standardterminologin för träd.

- **Nod**: Alla noder i trädet som är en del av den gemensamma strukturen för alla undersökningar kallas för noder. I exemplet, allt utom värdena `G10019221`, `Tandläkare`, `Slemhinneförändring`, `Sverige`.

- **Löv**: De specifika värden som matas in vid en undersökning och alltså inte ingår i den generella strukturen.

Det som är speciellt är att en nod inte måste ha några barn. T ex så kan Born sakna värde. Lövet, t ex Sverige, finns då inte men Born är ändå en nod.

Den generella strukturen illustreras i figur 2. Löv är markerade som fyrkanter, noder som ovaler.

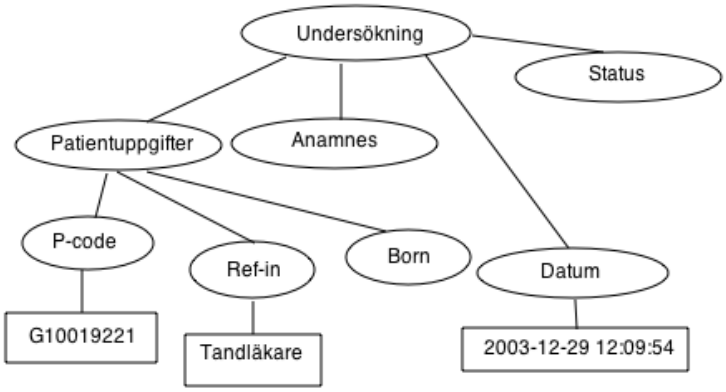

Figure 2: Undersökningsdata som träd.

3.2 Speciella noder

Varje trädfil innehåller två speciella noder `Datum` och `Konkret_identifikation`. Dessa är placerade direkt under roten *Undersökning*. `Datum`, som finns med i figur ref-fig:sampletree, är den tidpunkt då undersökningen gjordes. Formatet ska vara *yyyy-mm-dd hh:mm:ss*. `Konkret_identifikation` finns med av historiska skäl och kan i princip vara vad som helst. Ett lämpligt val är trädfilens namn.

3.3 Namngivning och teckenkodning

En trädfil känns igen på sin extension '.tree'. Trädfiler måste namnges på ett sätt som gör att de får ett unikt namn för att förhindra att data förloras. I MedView används för närvarande *P-code_tidpunkt.tree*, t ex `G10019221_031229120954.tree`. Programvara som läser trädfiler ska kunna läsa data oberoende av filnamnet så länge extensionen är '.tree'. Viss optimering kan dock göras om filnamnet inleds med *P-code*.

Teckenkodning: Trädfiler ska sparas med teckenkodningen **ISO Latin 1**.

3.4 Format

Trädroten *Undersökning* finns inte explicit representerad i en trädfil. Istället inleds varje trädfil med en rad som ursprungligen var tänkt att identifiera undersökningen. Idag har denna förlorat sin betydelse men av kompabilitetsskäl ska varje trädfil inledas med raden *yymmddhhmmss*, t ex

```
031229120954
```

Därefter följer en kodning av det representerade trädet som gör det möjligt att bygga upp det via en genomlöpning av trädfilen.

Den generella formen på innehållet är en följd av enheter på formen $(N|L)Str(\#*)$. Varje enhet representar en nod eller ett löv i trädet. Första tecknet i varje enhet står på början av en ny rad. Noder börjar med 'N' och löv med 'L'. Noder innehåller inga radbrytningar. Löv däremot kan innehålla strängar med nyradstecken.

Sist i varje enhet står 0 eller flera '#'-tecken. Ett löv avslutas alltid med minst ett '#'-tecken. Noder däremot kan avslutas med nyradstecken och sakna avslutande '#'.

Observera att tomma löv inte skrivs ut

3.5 Tolkning

Brädgårdarna som avslutar enheter används som navigationsanvisning då en trädfil läses. Varje '#' betyder att man ska förflytta sig ett steg uppåt från aktuell plats för att hitta rätt ställe att lägga in nästa enhet i trädet. Delträdet för *Patientuppgifter* i figur 2 blir:

```
NPATIENTUPPGIFTER
NP-code
LG10019221##
NRef-in
LTandläkare##
NRef-cause
LSlemhinneförändring#
LSjögrensutredning##
NBorn
LDanmark##
NNote01##
```

4 Exempelkod

Kod för att läsa och skriva trädfiler finns i MedViews CVS. Nedan följer i princip koden från klassen `Tree.java` i paketet `medview.datahandling.examination.tree`. Värdet i noden finns som en sträng i instansvariabeln `value`.

```
public String toString() {
    if (isLeaf())
        return leafToString();
    else
```

```
        return branchToString();
}

private String branchToString() {
    StringBuffer buffer = new StringBuffer();
    buffer.append( value );

    for (java.util.Iterator it = getChildrenIterator(); it.hasNext();) {
         Tree nextChild = (Tree) it.next();
         if (nextChild.isLeaf()) {
             String leafString = nextChild.toString(); // "" or "Value#"
             if (!leafString.trim().equals("")) {        // not ""
                 buffer.append("\nL");                   // Leaf marking
                 buffer.append(leafString); // "Value"
             }
         } else {
            buffer.append("\nN"); // Node marking
            buffer.append( (nextChild.toString()));
        }
    }
    buffer.append("#");
    return buffer.toString();
}

private String leafToString() {
    if (value != null && !value.trim().equals("")) {
        return ( value + "#");
    }
    return "";     // Don't write this leaf out since it is empty.
}
```

Revisionshistoria

- Denna version: 04-08.24
- Första version: 04-01-02
- Ursprunglig författare: Olof Torgersson

Att göra

Det saknas redogörelse för format på P-code, filstrukturen i en '.mvd'-struktur och namngivning av bilder.

References

[1] Y. Ali, G. Falkman, L. Hallnäs, M. Jontell, N. Nazari, and O. Torgersson. MedView: Design and adoption of an interactive system for oral medicine. In A. Hasman, B. Blobel, J. Dudeck, R. Engelbrecht, G. Gell, and H.-U. Prokosch, editors, *Medical Infobahn for Europe. Proceedings of* MIE 2000, volume 77 of *Stud. Health Tech. Inform.*, pages 3–7. IOS Press, 2000.

[2] G. Falkman. Information visualization in clinical odontology: Multidimensional analysis and interactive data exploration. *Artif. Intell. Med.*, 22(2):133–158, 2001.

[3] L. Hallnäs. Partial inductive definitions. *Theor. Comput. Sci.*, 87(1):115–142, 1991.

[4] O. Torgersson. Declarative programming and clinical medicine: On the use of Gisela in the MedView project. In S. Krishnamurthi and C. R. Ramakrishnan, editors, *Practical Aspects of Declarative Languages. Proceedings of PADL 2002*, volume 2257 of *Lect. Notes Comput. Sci.*, pages 64–81. Springer-Verlag, 2002.