Cognitive Assistance

Planungsassistenten für Menschen mit kognitiven Einschränkungen

19.01.2007

Gliederung

Erster Teil:

Kognitive Unterstützungssysteme

- Motivation
- Ziele
- Typen

Zweiter Teil:

- Autominder: Beispiel eines Planungsassistenten

Motivation (1)

- Weltbevölkerung altert in erheblichem Maße
- demographischer Wandel
- kognitive Einschränkungen korrelieren mit zunehmendem Alter
- erhöhte Lebensqualität im eigenen Zuhause

Motivation (2)

> Age 60	2000	2050
World	10.0%	21.4%
Belarus	19.3%	37.6%
Germany	23.2%	34.5%
Italy	24.1%	40.6%
Netherlands	18.2%	30.7%
Slovenia	19.2%	41.5%
United States	16.1%	25.5%
Mexico	6.9%	26.2%
Brazil	7.8%	25.9%
Columbia	6.9%	22.7%

China	6.9%	22.7%
India	7.5%	20.1%
Japan	23.3%	42.4%
Myanmar	6.8%	20.5%
Australia	16.4%	29.9%
Fiji	5.7%	22.7%
Egypt	6.8%	18.7%
Iran	6.4%	24.8%
Jordan	4.6%	19.0%
Botswana	4.2%	6.0%
Ethiopia	4.6%	7.7%
Mali	3.9%	5.3%

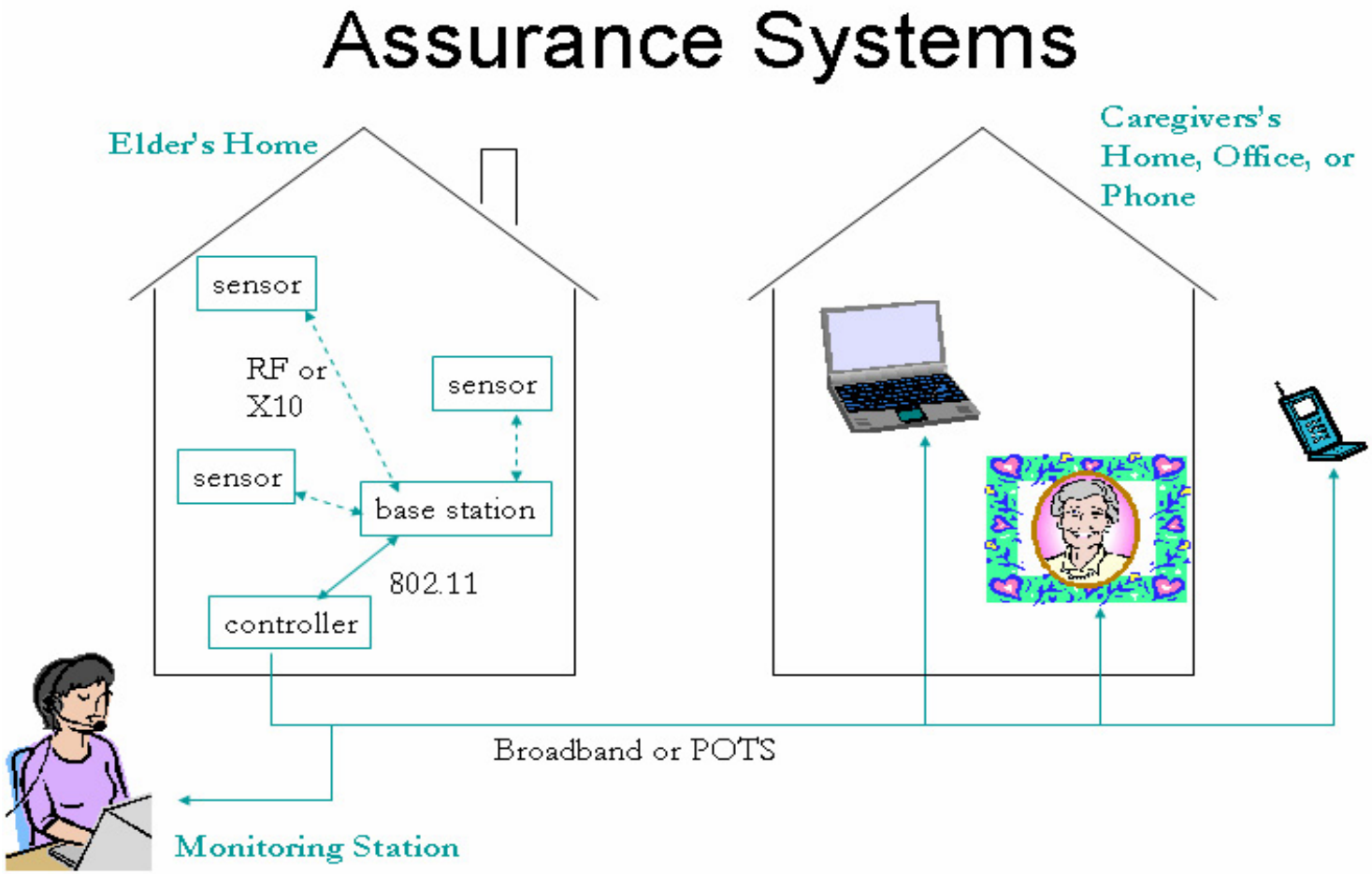
Ziele kognitiver Unterstützungssysteme

- Menschen mit kognitiven Einschränkungen unterstützen
- ihre Sicherheit gewährleisten
- Aufrechterhaltung der Durchführung der Aktivitäten des täglichen Lebens
- Abschätzung des kognitiven Status des Patienten
- Benachrichtigung eines Pflegers im Notfall

Ausprägungen

- Sicherheitssysteme (Assurance Systems)
- Kompensationssysteme (Compensation Systems)
  - Navigationsunterstützung
  - Terminmanagement
  - Aktivitätslenkung
- Beurteilungssysteme (Assessment Systems)

Sicherheitssysteme



Kompensationssysteme

Kompensierung der Einschränkungen die aufgrund von kognitiver Beeinträchtigungen entstehen

Unterstützung im Rahmen von:

- Navigation im Haus des Patienten
- Terminmanagement
- Bewältigung von Aufgaben mit mehreren Schritten
- Gesichtserkennung
- Lokalisierung von Objekten

Navigationsunterstützung

- hilft Menschen sich im häuslichen Umfeld zurecht zu finden
- bei Problemen mit visueller Wahrnehmung und Mobilität
- Hauptziel: Vermeidung von Hindernissen

- System: IMP (Intelligent Mobility Platform)
  - ausgestattet mit einer intelligenten Navigationssoftware
  - nach einer Initialisierungsphase plant das System einen Weg zu einem bestimmten Raum und umgeht dabei Hindernisse

Terminmanagement

- unterstützt die Bewältigung der Aktivitäten des täglichen Lebens durch den Einsatz von Erinnerungen
  - z.B. Medizin einnehmen
  - Essen zubereiten …
- erste Systeme nutzten Kalender oder Wecker für die Erinnerungen, heute auch kabellose Geräte wie Palmtops
- Problem: Menschen führen ihr Leben nicht nach unveränderlichen Terminplänen
- viele frühere, wie auch die meisten kommerziellen Systeme heute, sind nichts anderes als glorifizierte Wecker; dadurch zu unflexibel
- -> Autominder

Aktivitätslenkung

- unterstützt Menschen mit schwerwiegenderen Demenzen
- Lenkung einer Aktivität mit mehreren Schritten
- zum Beispiel: Händewaschen
  - Wasser anstellen
  - Seife benutzen
  - Hände abtrocknen
- Beispielsystem COACH:
  - Video Kamera beobachtet Patienten beim Händewaschen
  - Sensor-Tracking und Planerkennungsalgorithmus schließen auf den in der Aktivitätsfolge aktuell ausgeführten Schritt
  - bei Feststellung eines Problems, Ausgabe eines Signals

Beurteilungssysteme

- zur kontinuierlichen lebensnahen Beurteilung des kognitiven Status eines Patienten
- kognitive Beeinträchtigungen einer Person bleiben häufig über einen langen Zeitraum undiagnostiziert
- Kombination von sensor-basiertem Monitoring mit ausgeklügelten Analysealgorithmen um auf den kognitiven Status einer Person in deren gewohntem Umfeld zu schließen
- Beispiele:
  - Wired Independence Square; Sensoren in der Küche des Patienten
  - lernende Version des Solitaire Spieles FreeCell

Autominder: Einführung

- Software-System für ältere Menschen mit leichter und mittelschwerer Gedächtnisbeeinträchtigung
- entwickelt im Rahmen der Initiative für Personal Robotic Assistance (Nursebot Project)
- vollständig implementiert in Java und Lisp
- zusätzlich Web-basierties Interface für die Plan-Initialisierung und für Plan-Updates

Autominder: Plattform

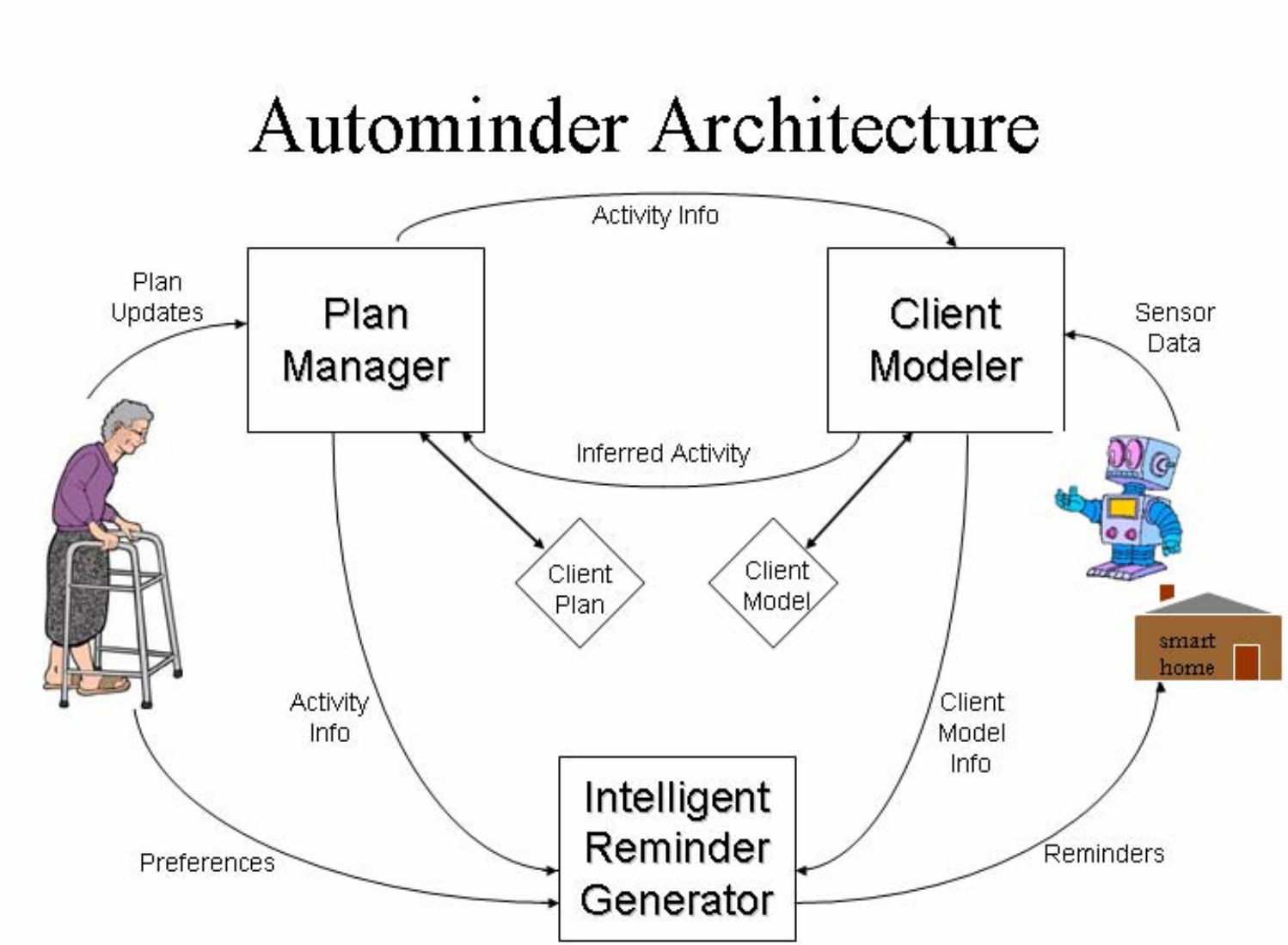
Autominder: Flexibilität

Interaktion Autominder <-> Patient

Beispiel:

- 80-jährige vergessliche Diabetes Patienten
- Tagesplan:
  - 07.00 Uhr; 11.00 Uhr; 15.00 Uhr; 19.00 Uhr etwas Essen
  - innerhalb höchstens einer Stunde nach dem Frühstuck und nach dem Abendessen Einnahme einer Tablette aufgrund einer Infektion
  - Lieblingsfernsehserie läuft zwischen 15.00 Uhr und 15.30 Uhr
- Autominder erkennt, dass um 11.10 Uhr zu Mittag gegessen wurde
- =>Erinnerung um 15.10 Uhr bzw. etwas früher aufgrund der Lieblingsfernsehserie

Autominder: Architektur



Der Plan Manager

- Technologie wurde übernommen und weiterentwickelt aus einem früheren Projekt (intelligentes Kalender Tool)
- Modellierung des sogenannten Client Plans
- Innerhalb des Client Plans wird die Sprache der disjunktiven temporalen Probleme (DTP) genutzt, um auf der Basis von Plänen Schlussfolgerungen ziehen zu können
- Ein Plan wird modelliert als 4-Tupel
  - <S,O,L,B>
  - S sind die einzelnen Schritte im Plan
  - O sind temporale Ordnungsbedingungen
  - L sind kausale Verknüpfungen
  - B sind Verknüpfungsbedingungen zwischen den einzelnen Schritten des jeweiligen Plans

Disjunctive Temporal Problem

- Temporale Bedingungen sind besonders wichtig in der Anwendungsdomäne der Termin Management Systeme
- formal definiert als konjunktive Menge von disjunktiven Ungleichungen
  - lb1 <= X1 – Y1 <= ub1 v … v lbn <= Xn – Yn <= ubn
  - Xi und Yi sind Start und Endzeitpunkte einzelner Schritte im Plan
  - lbn und ubn sind untere und obere Grenzen (reelle Zahlen)
  - zusätzlich existiert ein Time Reference Point (TRP), z.B. 0.00 Uhr
- durch DTP können nun eine Vielzahl von Constraints die
  - absolute Ereigniszeiten
  - relative Ereigniszeiten
  - und die Dauer von Ereignissen enthalten

  kodiert werden

DTP Beispiel (1)

- Planfragmente:
  - „Mit dem Frühstück soll zwischen 7.00 Uhr und 08.00 Uhr begonnen werden“

    => *420 <= Frühstück(S) – TRP <= 480*
  - „Frühstück dauert zwischen 20 und 30 Minuten“

    => *20 <= Frühstück(E) – Frühstück(S) <= 30*
  - „Innerhalb einer Stunde nach dem Frühstück soll eine Medizin eingenommen werden“

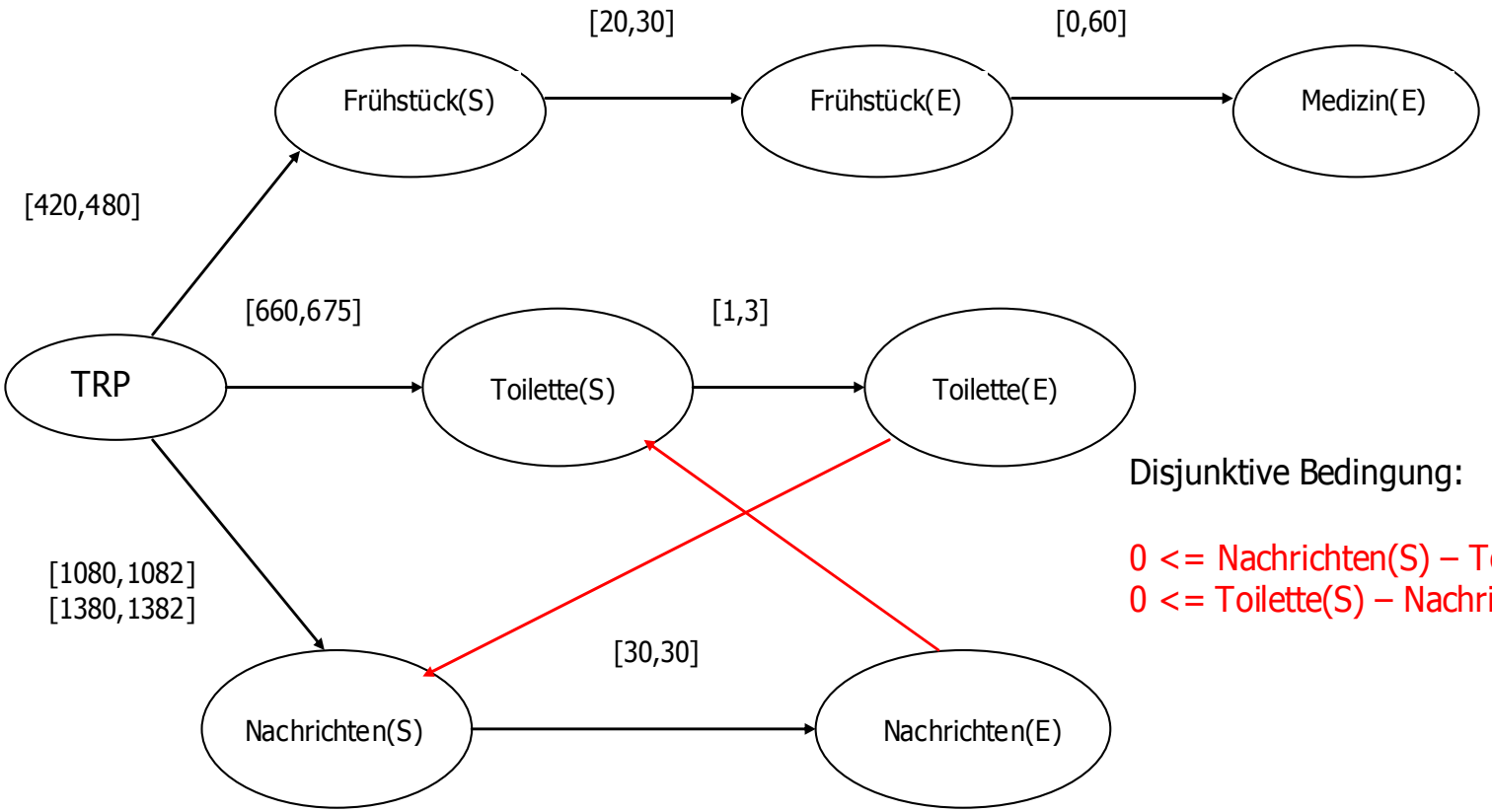
    => *0 <= MedizinNehmen(S) – Frühstück(E) <= 60*
  - „Toilettengang zwischen 11.00 Uhr und 11.15 Uhr“

    => *660 <= Toilette(S) – TRP <= 675*

DTP Beispiel (2)

- Planfragmente:
  - „Toilettengang dauert zwischen einer und drei Minuten“
    => 1 <= Toilette(E) – Toilette(S) <= 3
  - „Nachrichtensendung entweder um 18.00 Uhr oder um 23.00 Uhr“
    => 1080 <= Nachrichten(S) – TRP <= 1082 v
    1380 <= Nachrichten(S) – TRP <= 1382
  - „Die Nachrichtensendung dauert genau 30 Minuten“
    => 30 <= Nachrichten(E) – Nachrichten(S) <= 30
  - „Auf Toilette gehen und Nachrichten ansehen nicht gleichzeitig“
    => 0 <= Nachrichten(S) – Toilette(E) <= ∞ v
    0 <= Toilette(S) – Nachrichten(E) <= ∞

Temporal Network



Disjunktive Bedingung:

0 <= Nachrichten(S) – Toilette(E) <= ∞ v
0 <= Toilette(S) – Nachrichten(E) <= ∞

Plan Initialisierung

- PM wird mit einem Tagesplan, der vom Pfleger in Zusammenarbeit mit dem Patienten spezifiziert wird, initialisiert
- Spezifikation unterschiedlicher Tagespläne möglich

  (z.B. Wochentag, Wochenende)
- Über eine im Moment noch sehr rudimentäre GUI wählt der Benutzer unter bereits vorkonstruierten Routineaktivitäten aus und gibt zusätzlich zeitliche Bedingungen an
- Nachdem alle Planfragmente und zeitliche Restriktionen hinzugefügt wurden, prüft der PM die Konsistenz des Tagesplans und löst eventuell entstandene Konflikte

Plan Updates (1)

- Ein Update eines Tagesplans kann vier unterschiedliche Ereignisse als Grund haben:
  1. Das Hinzufügen einer neuen Aktivität zum Tagesplan
     (-> zusätzlicher Arzttermin)
  2. Die Modifikation bzw. das Löschen einer Aktivität
     (-> andere Zeitrestriktionen)
  3. Die Ausführung einer Aktivität (-> Medizin wurde eingenommen)
  4. Das überschreiten einer Zeitgrenze im Tagesplan
     (-> Patient sieht die Nachrichten um 23.00 Uhr)

Plan Updates (2)

- Algorithmus für Updates nachdem neue Planfragmente in den Tagesplan eingefügt wurden

Update-Plan-for-Addition(existing,new frag)

E = Convert-to-DTP(existing)
N = Convert-to-DTP(new frag)
C = Identify-conflicts(existing ∪ new frag)
R = {}
For each member c of C
    R = R ∪ a DTP-constraint representing the alternative temporal resolution of c
P = E ∪ N ∪ R
P` = Solve-DTP(P)
Return(Convert-to-Plan-Representation(P`))

Plan Updates (3)

- Algorithmus für Updates nachdem Modifikationen an den bereits existierenden Planfragmenten vorgenommen wurden

Update-Plan-for-Modification(existing,mods)

plan = Make the modifications in mods to existing
(i.e. remove and/or replace constraints)
M = Convert-to-DTP(plan)
C = Identify-conflicts(plan)
R = {}
For each member c of C
R = R ∪ a DTP-constraint representing the alternative
temporal resolution of c
P = M ∪ R
P` = Solve-DTP(P)
Return(Convert-to-Plan-Representation(P`))

Client Modeler (1)

- Aufgabe: Überwachung der Ausführung des Client Plans
  (An welcher Stelle das Client Plans befindet man sich??)
- Hierzu werden Informationen über beobachtbare Aktivitäten des Patienten, sowie die unterschiedlichen Zeitpunkte der Beobachtungen genutzt
- Momentan ist die Beobachter-Komponente des Client Modeler noch recht rudimentär in Form einer Kamera am Roboter umgesetzt
- Folge: Es kann nur auf den Ort (Raum), an dem sich der Patient im Moment befindet, geschlossen werden

Client Modeler (2)

- Um den aktuellen Zustand des Client Plans zu bestimmen, muss der CM mit temporalem Schließen unter Unsicherheit umgehen können
- Bisher gab es dafür zwei Ansätze:
  1. Zeitnetze (Time Nets)
  2. Dynamische Bayes`sche Netze (Dynamic Bayes Nets)
- Die Entwickler von Autominder kombinieren die Vorteile beider Ansätze in sogenannten Quantitative Temporal Bayesian Networks
- Dabei werden Daten über eine Interface Funktion zwischen Zeitnetzen und Dynamischen Bayes`schen Netzen transferiert

Intelligent Reminder Generator (1)

- Entscheidet darüber, welche und wann Erinnerungen generiert werden
- Zielsetzung -> Folgende vier Kriterien beachten:
  1. Darauf achten, dass der Patient die geplanten Aktivitäten ausführt
  2. Ein hohes Maß an Zufriedenheit, für den Patienten schaffen
  3. Vermeidung von Ineffizienz innerhalb der Patientenaktivitäten
  4. Verhindern, dass der Patient vom Erinnerungssystem abhängig wird (Aufrechterhaltung der Unabhängigkeit)
- Balance zwischen den vier Kriterien zu halten ist die schwierige herausfordernde Aufgabe

Intelligent Reminder Generator (2)

- Was zeichnet einen guten Plan für Erinnerungen aus?
- Ein qualitativ hochwertiger Erinnerungsplan sollte folgende Eigenschaften besitzen:
  1. Längere Zeiträume zwischen den einzelnen Erinnerungen
  2. Zeitnahe Erinnerungen sollten zu einer einzelnen Erinnerung aggregiert werden
  3. Vermeidung von überlappenden Erinnerungen
- Hierfür benutzt Autominder das Planning by Rewriting Paradigma

Planning by Rewriting (1)

- PbR ist ein Planungssystem, das lokale Suche und verschiedene Regeln benutzt, um suboptimale Pläne zu qualitativ hochwertigen Plänen zu transformieren
- PbR besteht aus vier Phasen:
  1. Die Generierung einer initialen Lösung
  2. Die Ausführung von verschiedenen Regeln um neue Lösungskandidaten zu produzieren
  3. Die Evaluierung der unterschiedlichen Kandidaten
  4. Die Selektion eines Plans für die weitere Entwicklung
- PbR wird als iterativer Prozess betrieben, der zu jeder Zeit während der Ausführung von Autominder passiert

Planning by Rewriting (2)

Rewrite Rules:

- Präferierte Zeit des Pflegers
- Früheste mögliche Startzeit einer Aktivität
- Späteste mögliche Startzeit einer Aktivität
- Lösche Erinnerungen die selten vom Patienten vergessen werden
- Kombiniere Erinnerungen die zeitlich eng aufeinander folgen

Planning by Rewriting (3)

PCO

1. Generate initial reminder plan P

Loop:

While no interrupt:

2. a. Use all applicable rewrite rules to create a neighborhood of reminder plans $P_1' \ldots P_n'$

b. For each plan P_i

i. Repair reminder plan as needed

ii. Check for consistency

c. Evaluate reminder plan quality

d. Choose P') with max value

3. If reminder interrupt:

a. Issue reminder;

b. Update reminder plan

4. If other interrupt:

a. Update reminder plan

End Loop

Zukunftsarbeit

- Benutzerfreundliche GUI zur Erstellung von Tagesplänen
- Erweiterung des Client Modelers:
  - Implementierung von zusätzlichen Sensoren, um das beobachtbare Patientenverhalten besser interpretieren zu können
  - Erweiterung zum Lernenden System, um Autominder individueller auf das Verhalten der einzelnen Patienten anzupassen
  - Evaluierung von QTBN
- Vom System generierte Erinnerungen sollten durch gewisse Rechtfertigungen unterstützt werden
- Migration des gesamten Systems vom Roboter auf kleinere, mobilere Systeme (Handhelds)