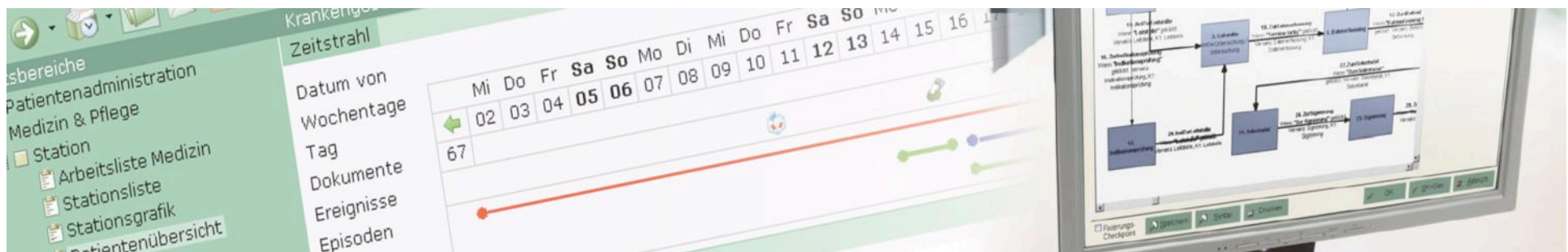


# Arden Syntax for Medical Logic Systems



HL7 Benutzergruppe Deutschland

Tutorial zur Jahrestagung 2005  
Freizeit-Inn Göttingen



Dr. Sven Tiffe

[sven.tiffe@gwi-ag.com](mailto:sven.tiffe@gwi-ag.com)

GW Research GmbH, Wien



# Überblick

- Teil 1: Hintergründe der Arden Syntax
- Teil 2: Einführung in die Arden Syntax
  - Aufbau der Wissensbasis
  - Anwendungsbeispiele
- Teil 3: Auswahl weiterer Tätigkeiten im TC



# Organisation des Technischen Komitees

- Clinical Decision Support TC
  - Co-Chairs: Ian Purves, Robert Jenders, R. Matthew Sailors, Robert Greenes
  - Arden Syntax SIG
    - This group supports the HL7 mission [...] by maintenance and further development of the **Arden Syntax**, a standard for representing and sharing clinical knowledge in Medical Logic Modules.
  - Clinical Guidelines SIG
    - This group supports the HL7 mission [...] by creating the standard for the communication of **clinical practice guidelines** to facilitate their integration into health care systems, electronic medical records, and a variety of applications.



# Teil 1: Einführung in klinische Entscheidungsunterstützung (CDS)

Nutzen und Beweggründe für  
entscheidungsunterstützende Systeme in der  
Medizin  
(CDSS: Clinical Decision Support Systems)



## Ein Unfall

Beide Beteiligten sind leicht verletzt.

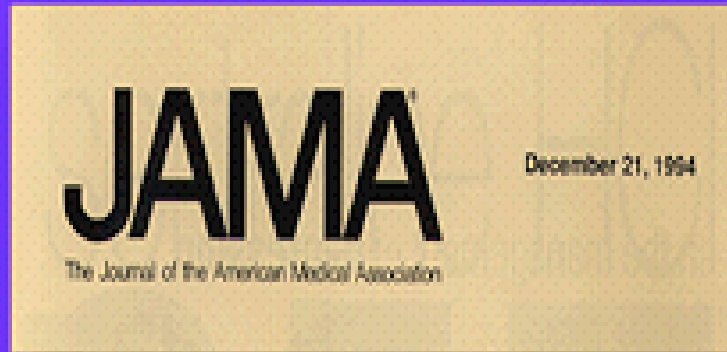


Sagt der eine:

„Glück gehabt, ich bin Arzt!“

Entgegnet der andere:

„Pech gehabt, ich bin Anwalt!“



**...180,000 people die each  
year partly as a result of  
iatrogenic injury...**

**Harvard Medical Practice Study (1991)**



Quelle: HL7 tutorial „Standards in Clinical Decision Support: Using Arden Syntax“



**The Inquirer**

**Medical Mistakes**

*Philadelphia Inquirer*  
September 12, 1996

**React Online**

[Discussion forum](#)

[Ask the experts](#)

**The Series**

[Part 1: Health care's deadly secret](#)

**Case Studies**

[A patient's story](#)

[Drug was 11 times the dose](#)

[Surgical misadventure](#)

[Seven hours waiting](#)

[Blood vessels pierced during surgeries](#)

[Blood thinner](#)

philly.com

## Health care's deadly secret: Accidents routinely happen

**Accidental Deaths in the U.S.**

An estimated one million people are injured by errors during hospital treatment each year and 120,000 people die as a result of those injuries, according to a study led by Lucian Leape of the Harvard School of Public Health. Here's how that number compares with other causes of accidental death in the United States\*.

\*SOURCE (for accidental deaths shown in blue): National Safety Council. Data are for 1996.  
KEVIN BURKETT / Inquirer Staff Artist

Cause of Death	Number of Deaths
Medical Error	120,000
Motor Vehicle Deaths	43,649
Deaths from Falls	14,986
Drowning Deaths	3,959
Commercial Aviation Deaths	329

The Medical College of Pennsylvania is a typical teaching hospital. It is known for cutting-edge research programs, for training medical students and newly graduated doctors, and for providing advanced medical care.

It is also representative of modern American hospitals in another respect: In the last decade alone, records show that hundreds of MCP Hospital patients have been seriously injured, and at least 100 have died after medical mistakes.

Helping AVOID costly clinical errors

DaSilva / For the Inquirer)

administration in the last decade. In the last decade, 100 survivors were never told that the injuries were caused by medical error. One of the doctors involved in the incidents was subjected to disciplinary



## Fehlervermeidung: „Adverse Events“

- Laut einer Analyse wichtiger Studien zu **unerwünschten medizinischen Ereignissen (Adverse Events) sterben** in amerikanischen Krankenhäusern **jährlich zwischen 44.000 und 98.000 Menschen** in Folge von **vermeidbaren Medikationsfehlern**. (Kohn LT, et al. *To Err Is Human: Building a Safer Health System*. Institute of Medicine, Washington, DC: National Academy Press, 2000.)
- Weitere amerikanische Studien kommen zu dem Ergebnis, dass in cirka **3 bis 4 Prozent der Krankenhausaufenthalte Adverse Events** eintreten, von denen circa 7 bis 14 Prozent zum Tode führen und die zum großen Teil vermeidbar gewesen wären. (Brennan TA *et al.* Results of the Harvard Medical Practice Study. *New England Journal of Medicine* 324(6):370-376, and 377-384 respectively, 1991, Thomas EJ *et al.* Incidence and Types of Adverse Events and Negligent Care in Utah and Colorado. *Medical Care*, Spring 2000)
- Ausgehend von der unteren Grenze, liegt die **Todesrate** aufgrund von Adverse Events an achter Stelle, **noch vor Verkehrsunfällen (~44k), Brustkrebs (~43k) oder AIDS (~16,5k)** (Death: Final data for 1997. CDC-National Vital Statistics Reports. 47(19):27, 1999. Births and Deaths: Preliminary data for 1998. CDC, National Vital Statistics Reports. 47(25):6, 1999.)
- Eine Studie der WHO für **europäische Länder** ermittelte einen Anteil von **Arzneimittelnebenwirkungen von 11,5% bis 16%** (World Health Organization. *Safety of Medicines: a guide to detecting and reporting ad-verse drug reactions*, Geneva, 2002.)



## Beispiel: Medikationsfehler

- Quellen von Medikationsfehlern

- Ärztliche Verschreibung
- Falsch interpretierte Handschrift
- Ausgabe der Medikamente
- Verabreichung

„CPOE“  
(Computerized Physician Order Entry)

- Arten von Medikationsfehlern

- Falsches Arzneimittel
- Falsche Dosis
- Falscher Patient
- Falscher Verabreichungsweg

Validierung und Benachrichtigung beispielsweise durch eingebettete Prüfreden



## Fakten Deutschland

⌘ 2003: 837 Mio. Packungseinheiten / Jahr (~1100 Pillen/Kapseln pro Einwohner) <sup>1</sup>

⌘ Arzneimittelverursachte Krankenhauseinweisungen <sup>2</sup>

⌘ Aufnahmen: ~1 pro 1000

⌘ Direkte Kosten: 400 Mio/€ pro Jahr

⌘ Rheumatherapie: <sup>3</sup>

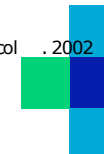
⌘ Verordnungen NSAR: 85 Mio./Jahr (0.5 €/Tag)

⌘ Therapie Nebenwirkungen(GKV): 50,5 Mio. €/Jahr

<sup>1</sup> Bund der pharmazeutischen Industrie 2002

<sup>2</sup> Schneeweis et al: Admissions caused by adverse drug events to internal medicine and emergency departments in hospitals; Eur.J.Clin.Pharmacol. 2002

<sup>3</sup> Deutsche Rheumaliga 2001





## Unerwünschte Arzneimittelwirkungen

Prospektive Kohortenstudien:

Station	EII/99	EAIII	LI/98	LII/98	GI/02
Bettenzahl	29	38	34	34	25
Dauer / Monate	7	7	2	2	4
Behandlungstage	6213		1300	1393	3586
Anzahl Patienten*	532	915	153	199	163
Anzahl der UAW	251	102	40	71	153
Pat. mit UAW	131	78	38	64	99
<b>UAW Inzidenz</b>	<b>25</b>	<b>9*</b>	<b>25</b>	<b>32</b>	<b>61</b>
UAW KHA in %	4	6	<b>9</b>		

\*entspricht Krankenhausaufnahmen

Dormann Drug Safety 2003; 26:353 -62  
 Egger Drugs Aging 2003; 20:769 -76  
 Dormann APT 2001; 15:171 -80  
 Levy Eur J Clin Pharmacol 1999; 54:887 -92

- **mehr als jeder 4. station äre Patient erleidet UAW**
- **Jede 10. station äre Aufnahme ist eine Nebenwirkung**





## Unerwünschte Arzneimittelwirkungen

<u>Studie:</u>	<u>EI/98</u>	<u>EII/99</u>	<u>EAI/01</u>	<u>LI/98</u>	<u>PII/02</u>	<u>GI/02</u>
<b>UAW total:</b>	<b>46</b>	<b>251</b>	<b>102</b>	<b>40</b>	<b>73</b>	<b>153</b>
<b>nicht erkannt :</b>	<b>63</b>	<b>64</b>	<b>57</b>	<b>60</b>	<b>45</b>	<b>45</b>
<b>Vorhersagbar*:</b>	<b>74</b>	<b>75</b>			<b>61</b>	
<b>Vermeidbar** :</b>		<b>48</b>	<b>41</b>		<b>34</b>	<b>20</b>

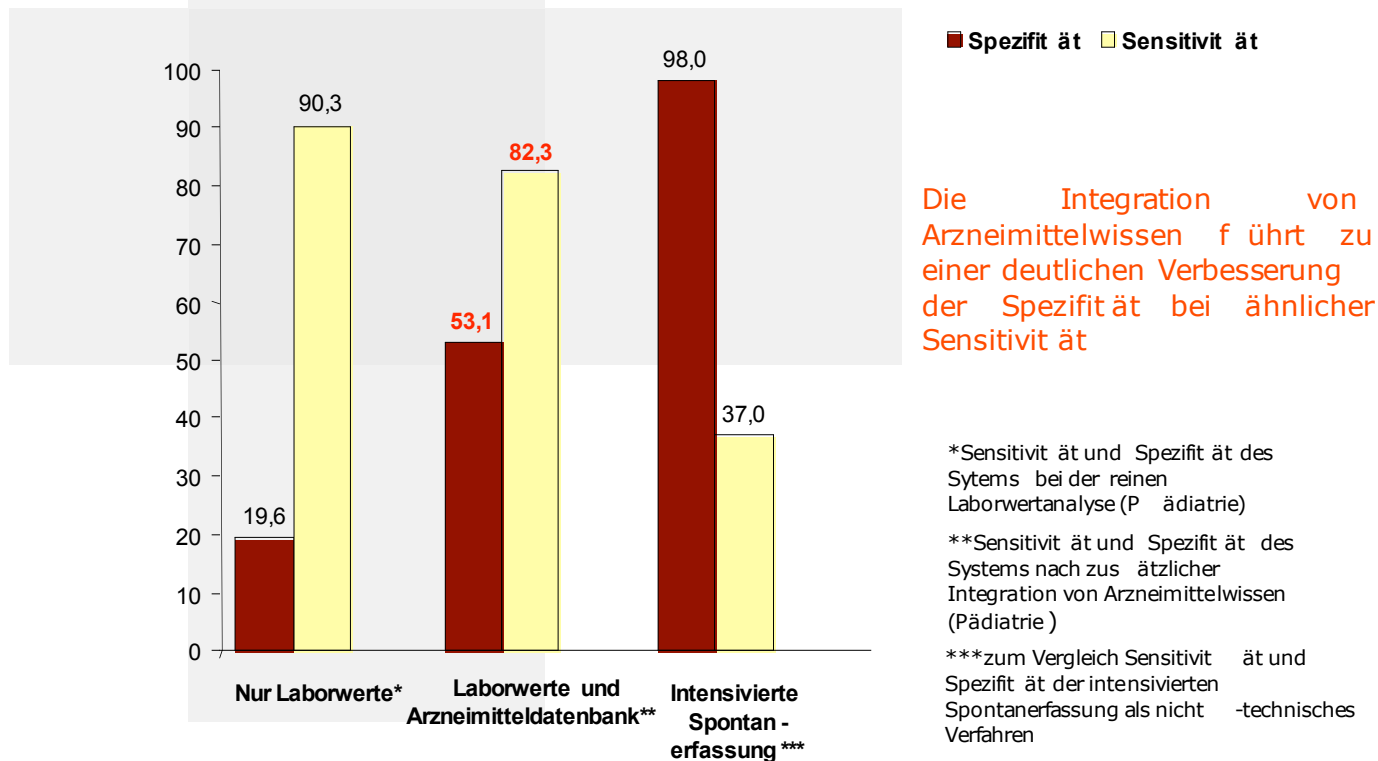
\* nach Edwards; \*\* nach Schumock

- ca. 2/3 der UAW vorhersagbar
- >50% der UAW nicht erkannt
- ca. 1/3 der UAW vermeidbar





## Vergleich verschiedener Erkennungssysteme

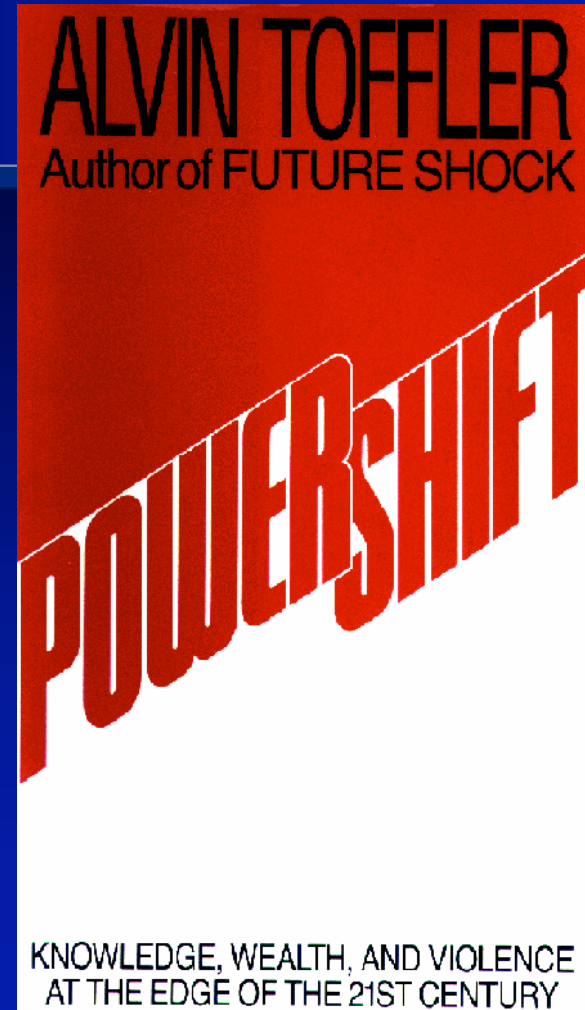




“

The problem facing decision-makers: from information underload to information overload.”

Data [...] are plentiful. Understanding is rare.





## Entscheidungsunterstützende Systeme

Dem klinischen Personal kann durch in die Arbeitsumgebung integrierte CDS-Systeme geholfen werden, indem große Datenmengen automatisiert gefiltert und geprüft werden.

- Verbesserung der Behandlungsqualität
- Reduktion der Behandlungskosten
- Unterstützung der Prozesse und Arbeitsabläufe



## Ansatzpunkte zur Nutzung von CDS-Systeme

- Finanzieller Aspekt von Behandlungsfehlern
- CDS-Systeme können helfen
  - Fehler zu vermeiden
  - Kosten zu senken
- Wachsender Bedarf, CDSS zu implementieren und zu nutzen
  - Patientensicherheit zentrales Thema der nächsten Jahre

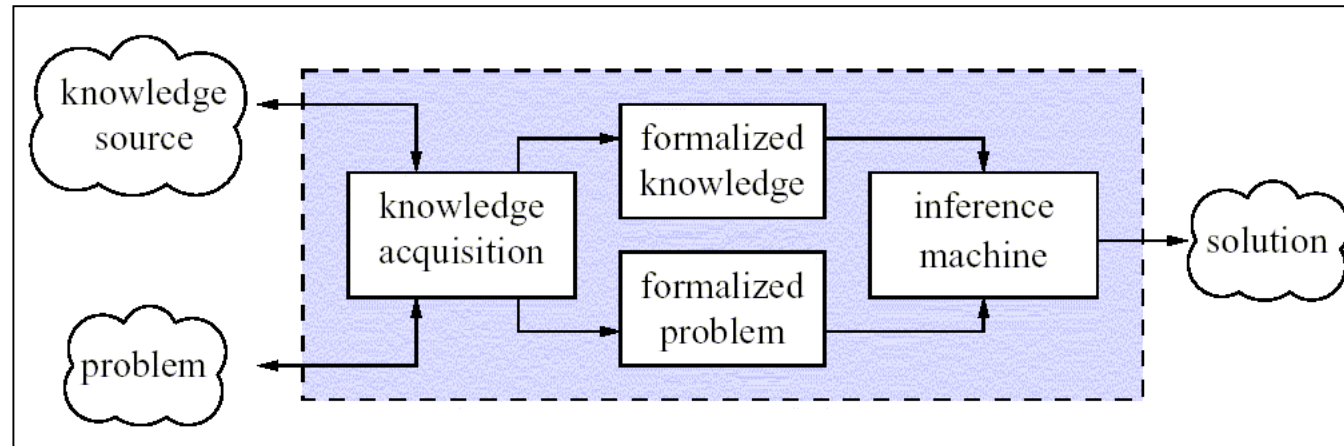


# Teil 1: Hintergründe der Arden Syntax

Geschichte der Arden Syntax,  
Vorteile einer standardisierten Wissens-Formats



## Wissensbasierte Systeme



- Trennung von Programm und „Wissen“
  - Explizite statt implizite Repräsentation von Wissen (Beispiel: Wenn-dann-Regeln, Leitlinien)
- Wissensrepräsentationsformat
  - Vom System ausführbar (Inferenzmaschine)
  - Vom Benutzer lesbar und verstehbar (und nicht nur vom Programmierer/Experten/Wissensingenieur)



## Arden Syntax - History

**HELP**

**LDS Hospital  
Salt Lake City, UT**

**CARE**

**Regenstrief Institute  
Indianapolis, IN**



**Arden Syntax  
1989**



## Beweggründe für die Arden Syntax

- „Warum nicht Visual Basic?“
- Definition von medizinischen Regeln durch Algorithmen
  - Situation um '89: vereinzelt Wissensbasen
  - Annahme, dass keine Organisation alles Wissen abbilden wird und dass Wissen daher austauschbar sein muss
  - Das Hintergrundwissen verändert sich mit der Zeit, Wissensbasen müssen gepflegt werden
- Ziel
  - *Wiederbenutzbarkeit* durch Standardisierung
  - *Übertragbarkeit* durch Trennung von Daten und Logik
  - *Verständlichkeit* durch leicht lesbare Syntax



## Arden Syntax: „sense of ownership“

- Wissensbasen werden üblicherweise durch Experten und Wissensingenieur erstellt
- Wichtiger sind die *Benutzer* des Systems
  - Akzeptanz von wissensbasierten Systemen kann erhöht werden, wenn Benutzer
    - nachvollziehen können, was passiert und warum es passiert
    - die Wissensbasis erweitern und modifizieren können
  - Benutzer sind selten erfahrene Programmierer
    - einfache, intuitiv verständliche Syntax für Algorithmen
    - Optimal: verschiedene Abstraktionsniveaus der Repräsentation (Text-basiert für Profis, graphisch-abstrakt für Laien)



## Geschichte der Arden Syntax SIG im CDS TC

- Anfänge der Arden Syntax in den frühen 90ern, erstmalig standardisiert als ASTM Standard 1992
- Eingliederung der Arden Syntax in die HL7
  - Aus der CDS SIG ging das Arden Syntax TC hervor
  - Umbenennung in CDS & Arden Syntax TC
- Anfang 1999 veröffentlichte die HL7 Arden Syntax 2.0
- Anfang 2000: Annäherungen zwischen dem TC und dem Intermed-Projekt (GLIF)
- Danach Umbenennung in CDS TC mit einer Arden Syntax SIG und einer Guidelines SIG (zwischenzeitlich GLIF SIG)



## Geschichte der Arden Syntax SIG im CDS TC

- 2002: Veröffentlichung der Arden Syntax 2.1
  - Kleinere Erweiterungen an der Sprachbasis zum String-Handling
  - „structured write“ Initiative von Motorola
- 2003: Erweiterungen um Fuzzy Logic angeregt
- 2005: Veröffentlichung der Arden Syntax 2.5
  - Objekte als Datentypen
- Arden Syntax homepage  
<http://cslxinfmtcs.csmc.edu/hl7/arden/>



## Jüngste Aktivitäten

- Aktuelle Arbeiten für 2.6
  - „time of day“ Datentyp
  - Internationalisierung der Arden Syntax
    - Generierung von mehrsprachigen MLMs
    - Betrifft im ersten Schritt nur Strings in der Logik
- Ausblick 3.0
  - Harmonisierung mit RIM
  - Erweitertes XML-Format
  - Weitere Arbeiten an Fuzzy Arden



# Arden Syntax in der Anwendung

- Universitäre Einrichtungen
  - Cedars-Sinai Medical Center, CA
  - CHU Henri Mordor, Paris
  - Linköping Universität
  - Universität Giessen
- Softwareanbieter
  - GWI AG
  - Siemens AG
  - Eclipsys
  - McKesson
- Wissensanbieter
  - Micromedex, USA
  - Medexter, Österreich



## Fazit

- Geringer Verbreitungsgrad und Nutzungsgrad von Expertensystemen in der Vergangenheit allgemein
  - Speziell zugeschnittene Expertensysteme sind häufig eng an die Infrastruktur gebunden
  - Keine breitflächige Nutzung von IT in Krankenhäusern
- Standardisierte Formate können den Austausch und die kommerzielle Wiederverwertung von Systemen fördern
- Der technische Fortschritt im KIS-Feld erleichtert die Einsetzbarkeit von CDS-Systemen



## Teil 2: Einführung in die Arden Syntax

Aufbau der Wissensbasis



## Medical Logic Modules

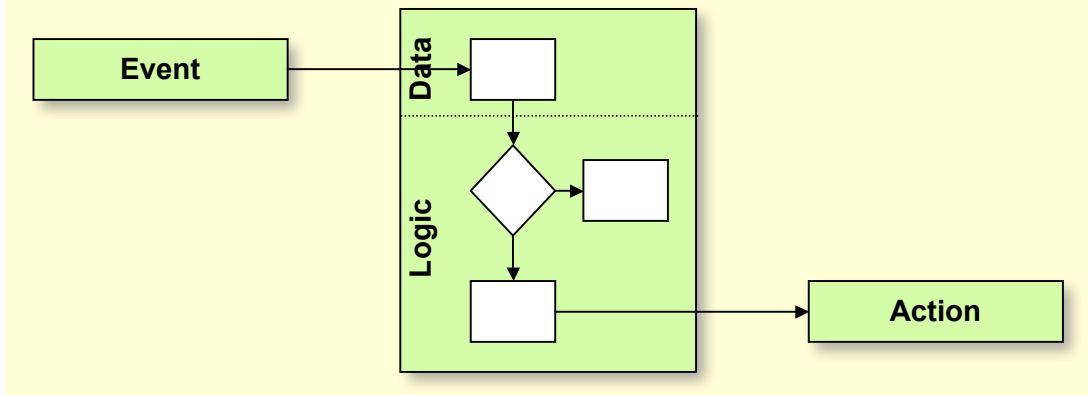
- Wissensbasis besteht aus einzelnen „Medical Logic Modules“ (MLMs)
  - Wiederverwendbarkeit
  - Literaturquellen, nähere Erklärungen zu Zweck und Funktion und Verweise auf zusätzliche Informationsquellen
    - Transparenz fördert die Akzeptanz
    - Begründungen können gelesen und nachvollzogen werden



## Arden Syntax: Struktur

- Aufbau der Wissensbasis durch einzelne Module, sog. „Medical Logic Modules“ (MLMs, üblicherw. als ASCII-Datei)
- Jedes MLM enthält ausreichend Wissen, um unabhängig von anderen MLMs Entscheidungen treffen zu können

„Wenn etwas passiert und eine Bedingung erfüllt ist, dann führe etwas aus“

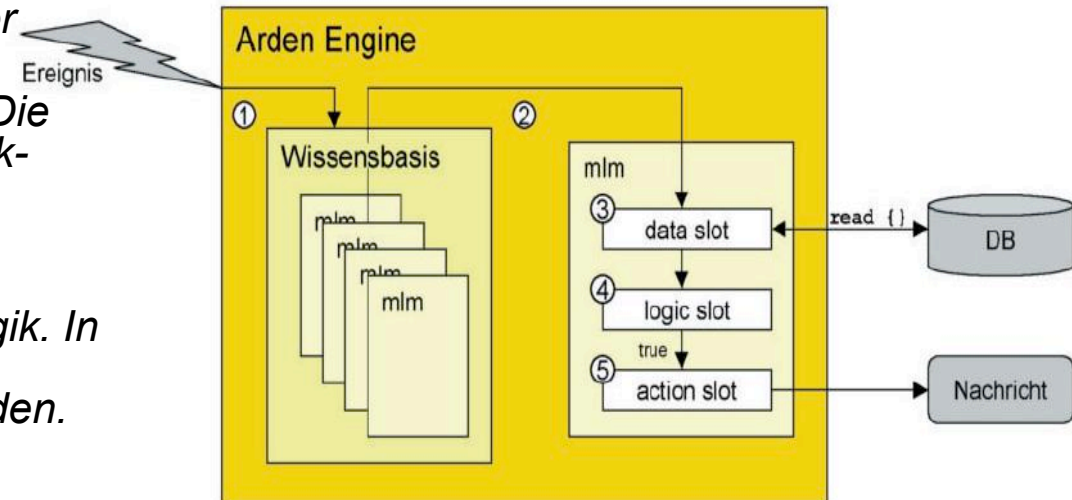


- Trennung von Datenbankabfrage und Logik
  - „curly braces problem“



## Schematischer Ablauf

1. Auswahl der auszuführenden MLMs aus der Wissensbasis. Die Bedingung für die Ausführung eines MLMs ist im evoke slot eines Modules definiert.
2. Aufruf jedes ausgewählten MLMs. Der Aufruf kann zeitlich verzögert oder periodisch wiederholt geschehen.
3. Einlesen der Daten im data slot. Die eigentliche Syntax des Datenbankzugriffs steht innerhalb der geschweiften Klammern und ist systemabhängig.
4. Ausführung der Entscheidungslogik. In diesem Schritt können die zuvor gelesenen Daten verarbeitet werden.
5. Schließt die Logik mit positivem Ergebnis ab, wird der action slot ausgeführt.
6. Damit ist die Ausführung des MLMs abgeschlossen.





# Aufbau eines MLMs

Inhaltlich in drei Kategorien geteilt:

- Maintenance:  
Organisatorische Informationen zu dem MLM, z.B. der Autor und der Verantwortliche
- Library:  
Inhaltliche Hintergrundinformationen, wie eine Erklärung oder Literaturquellen
- Knowledge:  
Die eigentliche Regel als Algorithmus, Datenbankzugriffe und Eventdefinitionen

```
maintenance:  
slotname: slot-body;;  
slotname: slot-body;;  
...  
library:  
slotname: slot-body;;  
...  
knowledge:  
slotname: slot-body;;  
...  
end:
```



## Maintenance slot: Beispiel

**title:** Check for penicillin allergy;;  
**mimname:** pen\_allergy;;  
**arden:** version 2.1;;  
**version:** 1.00;;  
**institution:** Columbia University;;  
**author:** George Hripcsak, M.D.;;  
**specialist:** Jane Doe, Ph.D.;;  
**date:** 1989-01-02;;  
**validation:** testing;;



## Library-Slot: Beispiel

**purpose:** When a penicillin is prescribed, check for an allergy. (This MLM demonstrates checking for contraindications.);;

**explanation:** This MLM is evoked when a penicillin medication is ordered. An alert is generated because the patient has an allergy to penicillin recorded.;;

**keywords:** penicillin; allergy;;

**citations:**

1. SUPPORT Steiner RW. Interpreting the fractional excretion of sodium. Am J Med 1984;77:699-702.
2. Goldman L, Cook EF, Brand DA, Lee TH, Rouan GW, Weisberg MC, et al. A computer protocol to predict myocardial infarction in emergency department patients with chest pain. N Engl J Med 1988;318(13):797-803.;;

**links:**

```
OTHER_LINK 'CTIM .34.56.78';  
MESH 'agranulocytosis/ci and sulfamethoxazole/ae';  
URL "NLM Web Page", 'http://www.nlm.nih.gov/';  
URL "Visible Human Project",  
'http://www.nlm.nih.gov/research/visible/visible_human.html';  
URL "DOS HTML File", 'file://doslinx.htm';  
URL "UNIX HTML File", 'file://UnixLinx.html/';;
```



## Knowledge-Slot: Beispiel

**type:** data-driven;;

**data:**

/\* an order for a penicillin evokes this MLM \*/

penicillin\_order := event {medication\_order where class = penicillin};

/\* find allergies \*/

penicillin\_allergy := read last {allergy where agent\_class = penicillin};

::

**evoke:**

penicillin\_order;;

**logic:**

if exist (penicillin\_allergy) then

conclude true;

endif;

::

**action:**

write "Caution, the patient has the following allergy to penicillin documented: " ||

penicillin\_allergy;;

**urgency:** 50;;



## Variablen und Datentypen

- Variablen: alphanumerische Platzhalter für Daten
- Nicht deklarativ
  - Variablen können an jeder Stelle des MLMs verwendet werden, ohne vorher deklariert werden zu müssen
  - Variablen sind nicht Typ-gebunden und können ihren Typ während der Ausführung wechseln
- Seit Arden Syntax 2.5: Datenstrukturen als „Objekte“
  - Spezifikation sieht die Deklaration von Objekttypen vor
  - Reine Datensammlung, keine Methoden
  - Zugriff über „dot-notation“



# Datentypen

- **Null:**  
Spezieller Datentyp für undefinierte Ergebnisse
- **Boolean:**  
Wahrheitswerte für eine dreiwertige Logik
- **Number:**  
Zahlen (Fließkomma- und Ganzzahlen)
- **Time:**  
Zeitstempel (Datum und Zeit)
- **Duration:**  
Zeitdauer (in Jahren, Monaten, Tagen, Stunden, etc)
- **String:**  
Zeichenketten mit Zeilenumbruch
- **Term:**  
feste Zeichenkette für spezielle Funktionalitäten
- **List:**  
Eindimensionale Listen
- **Objekte:**  
Datenstrukturen beliebiger Typen



# Null

- Spezieller Datentyp zur Repräsentation von unbestimmten Daten (“uncertainty” laut Spezifikation)
- Üblicherweise das Ergebnis von Datenbankabfragen ohne Ergebnis oder fehlerhafter Verwendung von Operatoren
  - Z.B. Division durch 0
  - Z.B. direkte Zuweisung von *null*



# Boolean

- Zwei Werte vom Typ Boolean *true* und *false*.
  - Der Wert *true* entspricht dem Wahrheitswert „wahr“
  - Der Wert *false* entspricht dem Wahrheitswert „falsch“
- Da Variablen nicht typgebunden sind und bei der Verwendung für Wahrheitswerte neben *true* und *false* auch *null* sein können, sind alle logischen Operatoren auf eine dreiwertige Logik ausgelegt, zum Beispiel
  - `null := true and null;`
  - `true := true or null;`



# Number

- Es gibt genau einen Datentyp für Zahlen
  - Es wird nicht zwischen Ganzzahlen (integer) und Fließkommazahlen (float) gemacht
  - Intern basiert jede Berechnung von Zahlen laut Spezifikation auf Fließkommazahlen, ohne Angabe der Genauigkeit



# Time

- Werte vom Typ time referenzieren absolute Zeitpunkte, z.B. 2005-10-25T13:00:00
- Bei Konstanten werden Datum oder Datum und Zeit spezifiziert
  - Ein Datentyp, der nur eine Zeit repräsentiert, wird aktuell für Arden Syntax 2.6 erarbeitet
- Zeiten vor dem 1.1.1800 sind ungültig
- Die Granularität von Zeiten ist unendlich (d.h. nicht beschränkt auf diskrete Sekunden)
  - Zeiten aus einer Datenbank haben üblicherweise abweichende Granularitäten



## Spezielle Zeitkonstanten

- Now                      Startzeitpunkt des MLMs
- Eventtime              Zeit des auslösenden Ereignisses
- Triggertime            Technischer Zeitpunkt des Events
- Curretime              Aktuelle Zeit ("right now")

**eventtime <= triggertime <= now <= curretime**



## Duration

- Durations repräsentieren ein Zeitintervall ohne direkten Bezug zu einem Zeitpunkt
- Es gibt keine Konstanten, Zeiten werden mittels der duration Operatoren erstellt.



# String

- Zeichenketten variabler Länge
- Durch Anführungszeichen (“”) definiert, z.B  
“HL7 Deutschland”  
oder  
“HL7 Deutschland Jahrestagung  
Göttingen 2005”



## Term

- Einzeilige Zeichenketten variabler Länge, durch einfache Anführungszeichen definiert (‘)
- Derzeitig nur zur Verwendung als Konstanten genutzt, z.B.
  - Referenzierung von MLMs
  - Links im link-slot



## List

- Geordnete Mengen von Elementen
- Listen können heterogen sein und Elemente unterschiedlicher Typen enthalten
- Listen können nicht verschachtelt werden
- Leere Liste als Konstante definiert
  - Als Paar leerer Klammern: ()
- Listen sind das Ergebnis von
  - Datenbankabfragen
  - Aneinanderreihung von einzelnen Elementen durch Listenoperatoren, z.B. Komma (,)



## Objekte

- Objekte sind lokal definierte Datenstrukturen
  - Elemente: Menge von benannten Variablen
  - Elemente sind nicht Typgebunden
- Objekte haben keine Methoden
- Vereinfachung bei der Handhabung großer Datenmengen oder Abfrageergebnisse
- Beispiel:
  - `let Patient be object [ name, vorname, geburtsdatum ];`



## Abfrageergebnisse

- Die Ergebnisse einer Datenbankabfrage werden im data slot Variablen zugewiesen, um sie im logic slot auswerten zu können
- Ergebnisse sollten zusätzlich zu jedem eigentlichen Ergebniswert einen dazugehörigen Timestamp assoziieren
- **„Primary Time“**
  - Medizinisch relevanter Zeitstempel
  - Für jedem Zugriff auf die Datenbank wird implizit ein Zugriff zur Ermittlung der primary time vorausgesetzt
  - Herstellerabhängig
- Die primary time ist zentral notwendig, um die Zeit-bezogenen Funktionen der Arden Syntax verwenden zu können



## Exkurs: Grenzen der Standardisierung

- Curly braces Problem
  - Schnittstellen zum implementierenden System nicht spezifiziert
    - Datenzugriffe
    - Event-Definitionen
    - Nachrichten
  - Große Unterschiede zwischen Datenmodellen erfordern ggf. große Änderungen an der Logik eines MLMs



## Exkurs: Mit den curly braces umgehen lernen

- Alternative 1: Umschreiben der MLMs
- Alternative 2: Verzicht auf Nutzung von Datenbank-Zugriffen, Definition einer Schnittstelle
  - MLMs können sich gegenseitig referenzieren
  - MLMs können mit Parametern aufgerufen werden und Ergebnisse an das aufrufende MLM zurückliefern
- Alternative 3: Lösung des curly braces problems
  - Im CDS TC wird an Lösungen zu den bislang nicht spezifizierten Schnittstellen gearbeitet
  - Das gleiche Problem betrifft auch die einheitliche Repräsentation von klinischen Leitlinien



## Exkurs: vMR

### *Virtual Medical Record*

- Vereinfachte Darstellung einer Patientenakte
- Ausschließlich CDS-relevante Aspekte
- Ziel des Projektes ist es, die minimale Auswahl an Attributen und Datenstrukturen zu finden, die eine semantische Interoperabilität von CDS-Systemen ermöglichen



# Operatoren und Statements

- Operatoren verarbeiten Daten zu neuen Daten (expressions)
- Statements steuern den „Programmablauf“ in den data, logic und action slots



# Operatoren-Gruppen

- Die Arden Syntax legt einen Schwerpunkt auf die Handhabung von Listen, Vergleichen und zeitlichen Bedingungen
- Übliche Operatoren zur
  - Handhabung von Listen (list operators)
  - Verknüpfung von Wahrheitswerten (logical operators)
  - Einfachen und komplexen Vergleichen (comparison operators)
  - Verarbeiten und Vergleichen von Strings (string operators)
  - Rechenoperationen (arithmetic & numeric operators)
  - Verarbeiten von Listen (aggregation & transformation operators)
  - Arbeiten auf Objekten (dot und clone operator)



Operators



## Spezielle Operatoren

- Fokus auf zeitliche Vergleiche
  - `measurements occurred within past 3 hours`  
wendet das Kriterium „geschah in den letzten drei Stunden“ auf jedes Element an
- Where-Operator
  - Wählt aus einer Liste die Elemente aus, deren korrespondierende Elemente aus der zweiten Liste `true` ergeben
  - `measurements where they occurred within past 3 hours`



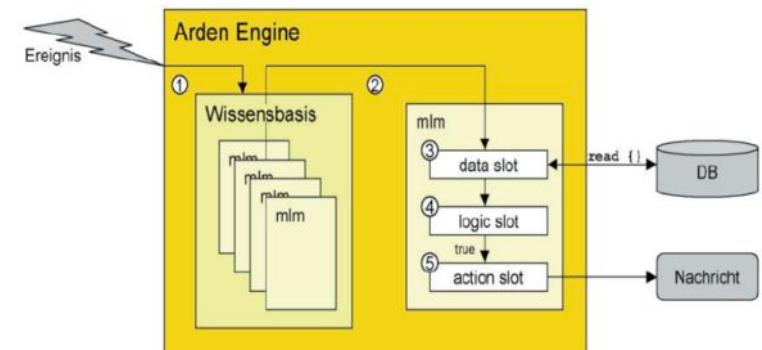
## Elemente zur Ablaufsteuerung (statements)

- Data, Logic und Action slot erlauben unterschiedliche Elemente zur Ablaufsteuerung
  - Zuweisungen
  - Verzweigungen (if-then-else)
  - Beendung (conclude)
  - Aufrufe (call)
  - Schleifen (for, while)
  - Objekte (new)
- Beispiel: Zuweisungen zu Variablen sind im data und logic slot erlaubt, nicht aber im action slot



## Statements im Data Slot

- Variablen („Bezeichner“) im MLM werden auf Datenbankelemente gemappt
  - laborwerte := read { laborwerte des letzten auftrags };
- Die curly braces { } erlauben ein flexibles Mapping auf die lokalen Datenbankstrukturen
- Diese Art des Mappings sollte die Logik der Regel von den lokalen, Anbieter-abhängigen Datenstrukturen trennen





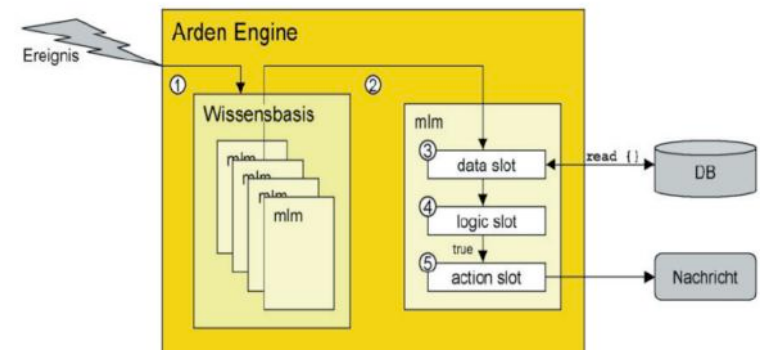
## Data Slot – Statements

- Ein einfaches read statement *ohne* einen optionalen Operator ergibt eine Liste von Werten
  - `creatinine := read {'dam'="PDQRES2"}`
- Ein read statement *mit* einem Operator ergibt ein einzelnes Ergebnis:
  - `last_creat := read last {select "OBSRV_VALUE" from "LCR" where qualifier in ("CREATININE","QUERY_OBSRV_ALL")};`
- Beispiele für Operatoren:
  - first, last,*
  - earliest, latest,*
  - min, max,*
  - count, average, sum*



## Evoked Slot

- evoked slot definiert, wie ein MLM aufgerufen werden kann
- Beispiele:
  - Das Geschehen eines Ereignis
  - Verzögerte Ausführung nach einem Ereignis
  - Zyklische Ausführung nach einem Ereignis
  - Direkter Aufruf aus einem anderen MLM (immer möglich)





## Evoke Slot - Beispiele

```
data: creatinine_storage := event {'32506', '32752'};
```

```
evoke: creatinine_storage;;
```

```
evoke: 3 days after time of creatinine_storage;
```

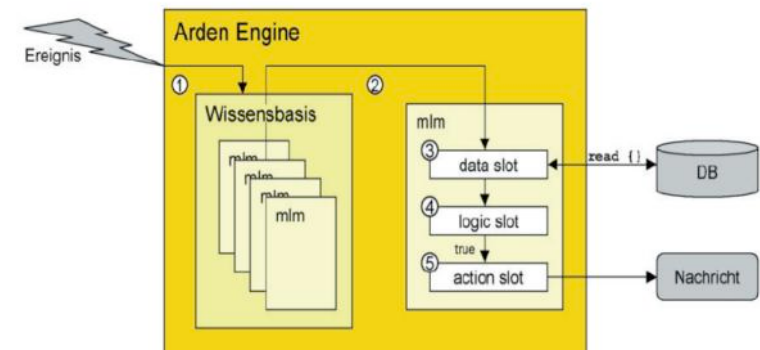
```
evoke: every 1 day for 7 days starting at time of  
creatinine_storage;
```

```
evoke: every 1 day starting at  
time of K_storage until K>=3;
```



## Logic Slot

- Die medizinische Logik wird als Algorithmus beschrieben
- Wird durch das “conclude statement” beendet
  - conclude true;
  - conclude false;
  - statt Konstanten auch Ausdrücke erlaubt, z.B.
    - conclude (value is greater than 5);
- Beendet die Ausführung des logic slots sofort
- Bei conclude true wird der Action Slot angesprungen





## Logic/Action/Data Slot - if ... then ...

```
if <expr1> then  
    <block1>  
endif;
```

muss true ergeben

```
if <expr1> then  
    <block1>  
else  
    <block2>  
endif;
```

```
if <expr1> then  
    <block1>  
elseif <expr2> then  
    <block2>  
...  
elseif <exprN> then  
    <blockN>  
else  
    <blockE>  
endif;
```



## Logic Slot - Schleifen

```
while <expr> do  
    <block>  
enddo;
```

Führt den Block solange aus, wie der Ausdruck true ergibt.

```
for <expr> do  
    <block>  
enddo;
```

Führt für jedes Element einer Liste den Block einmal aus.



## Logic Slot - Aufrufe

`var1 := call my_mlm with param1, param2;`

`var1 := call my_event with param1, param2;`

`var1 := call my_interface_function with param1, param2;`



## Logic Slot - Beispiel

logic:

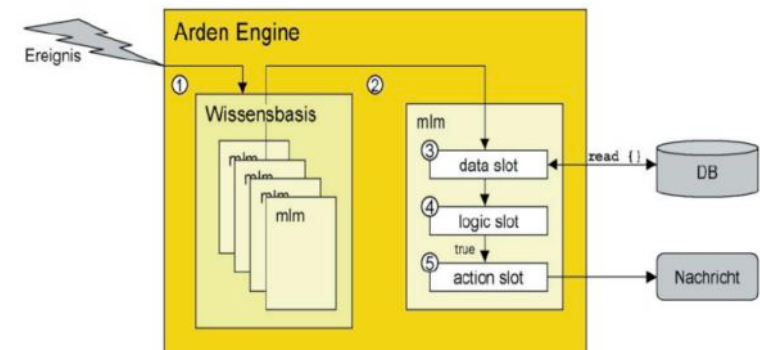
```
if last_creat is not present then
    alert_text := "No recent creatinine available. Consider
ordering creatinine before giving IV contrast.";
    conclude true;
elseif last_creat > 1.5 then
    alert_text := "This patient has an elevated creatinine.
Giving IV contrast may worsen renal function." ;
    conclude true;
else conclude false;
endif;
```



# Action Slot

Übliche Aktionen im action slot:

- Erzeugen einer Nachricht
  - mit Anzeige auf dem Bildschirm
  - als Mail
  - in die Datenbank
- Aufruf weiterer MLMs
- Schreiben in Datenbank





## Action Slot - Beispiel

```
action: write "Last creatinine: " || last_creatin || " on: "  
        || time of last_creatin;  
ergibt: Last creatinine: 2.36 on: 1997-02-16T06:30:00
```

data:

```
john_email := destination {'john@arden.edu'};
```

← Abhängig vom Anbieter

...

action:

```
write "Patient who may qualify for study registered  
today."  
Pt #: "|| patient_no at john_email ;
```



## Teil 2: Einführung in die Arden Syntax

Änderungen in Arden Syntax 2.5  
und Arbeiten an Arden Syntax 2.6



## Arden Syntax 2.5

- Objekte
  - Schrittweise Erweiterung um Objekt-Strukturen in der Arden Syntax
    - “dot notation”
    - In erster Linie Vereinfachung der Listen-Handhabung
  - Anbieter bestehender Lösungen möchten in Arden Syntax 2.x aus Kompatibilitätsgründen nur behutsame Änderungen
- XML



## Punkt-Notation

- Ein behutsamer, schrittweiser Ansatz zu Objekten in der Arden Syntax
- Abwärtskompatibel, betrifft nicht existierende MLMs
- Annahme, dass die Änderung von der Benutzer-Zielgruppe leicht angenommen wird



## Punkt-Notation: Phasen

- Phase 1 – Kosmetische Änderungen
- Phase 2 – Einfache Objekte und Attribute
- Phase 3 – Listen mit Objekten



## Phase 1 – Kosmetische Änderungen

- Punkte als Bestandteil von Variablen-Bezeichnern erlauben (verwendbar ähnlich wie Unterstriche)
- Trotzdem: der Variablenname muß komplett angegeben werden – noch keine „echten“ Objekte
- Patient\_FirstName := "BILLY";
- Patient.FirstName := "BILLY";

```
(Patient.FirstName, Patient.LastName) :=  
  READ last {PatientName};
```



## Phase 2 – Einfache Objekte und Attribute

- Variablen werden als ‚Objekt.Attribut‘ verwendet
  - Objekte sind reine Container
  - Attribute sind Elemente des Objekts
- Patient.FirstName := "BILLY";
- Patient.LastName := "SMITH";
- Patient.DOB := 1960-10-01T00:00;



## Einfache Arden Objekte

- Einfache Objekte entsprechen einer heterogenen Arden Syntax Liste
- Attribute eines Objektes entsprechen „benannten Elementen“,
  - vgl. Maps/HashMaps in Java
- Aber: Zuweisung von Elementen und Änderungen von Elementen einer einzelnen Liste in Arden nicht möglich
  - Bistlang: Neuaufbau der Liste



## Einfache Arden Objekte

- Arden Syntax Liste

```
Pt_info := "SMITH", "BILLY", 1960-03-07T00:00:00;  
New_pt := pt_info;  
new_pt[1] is "SMITH",  
new_pt[2] is "BILLY",  
new_pt[3] is 1960-03-07T00:00:00
```

- Einfaches Arden Objekt

```
pt_info := READ last {Patient_Info};  
          // this READ mapping returns 3 elements  
new_pt := pt_info;  
new_pt.LastName is "SMITH",  
new_pt.FirstName is "BILLY",  
new_pt.DOB is 1960-03-07T00:00:00
```



## Default-Elemente eines Objekts

- Arden Objekte haben zwei Standard-Elemente
  - `.value`
  - `.primary_time`
- Wird ein Objekt mit einem herkömmlichen Operator verwendet, ohne ein Element anzugeben, wird `.value` verwendet:
  - `glucose := 98`                      gleich
  - `glucose.value := 98`



## Default-Elemente eines Objekts

- Analog wird durch den Operator 'time of' auf das *.primary\_time* Element zugegriffen
  - `time of glucose`      `gleich`
  - `glucose.primary_time`
- Ein Objekt hat immer genau dann eine primary time, wenn alle Elemente des Objekts die gleiche primary time haben



## XML und Arden 2.5

- XML-kodierte MLMs
- Was wurde strukturiert
  - Maintenance category slots und deren Inhalte
  - Library category slots und deren Inhalte
  - Separate knowledge category slots, aber nicht die Inhalte (strukturierte Slots werden als Plain Text eingebunden)



## XML-kodierte MLMs – Warum?

- Verwendung von verbreiteten Standard-Werkzeugen möglich
  - Authoring tools
  - Entwicklung von Interpretern
  - Leichtere Indizierung, Verwaltung und Suche von MLMs
- Übermittlung in XML-basierten Nachrichtensystemen
- Übersetzung in andere Repräsentationen
  - Programmier- und Skriptsprachen
  - Anzeige am Bildschirm



## Arden Syntax 2.6

- Kleine Erweiterungen mit kurzfristigen Auswirkungen und Implementierbarkeit für Anbieter
- Voraussichtlich letzte Version vor Arden Syntax 3.0
- Time-of-day Operator
- Internationalisierung



## Aktueller “time” Datentyp

- Zwei Informationseinheiten: ein Datum und eine Zeit

2004-09-20T12:00:00

oder

2005-01-25

- Time Werte können für zeitliche Kriterien verwendet werden, zum Beispiel

lab\_values where they occurred  
within now - 6 days to now - 3 days

- Time Werte beziehen sich aber **immer** auf ein konkretes Datum



## Datum-unabhängige Zeitkriterien

- “Anforderung xy nur zwischen 6.30 Uhr und 16.15 Uhr erlaubt”
  - Unabhängig gültig von einem bestimmten Datum
  - Nicht ohne weiteres in Arden Syntax (komfortabel) formulierbar
- Workaround: extract hour/minute Operatoren

```
hhmm = hour of now * 100 + minute of now;  
if hhmm >= 630 and hhmm <= 1615 then ...
```
- Noch lesbar, aber mühsam
- “nicht zwischen 22:00 Uhr und 04:00 Uhr”

```
if hhmm >= 2200 or hhmm <= 400 then ...  
or  
if hhmm is not within 400 to 2200 then ...
```



## Lösungsansatz: neuer Datentyp für Zeit

- Vorschlag: Definition eines zusätzlichen Daten-Subtype
  - Kann nicht ‘time’ genannt werden, da diese Bezeichnung schon für den “timestamp” verwendet wird
  - Konsens: **‘time-of-day’**



## Time-of-day

- Der time-of-day Datentyp bezieht sich auf Zeiten ohne Bezug zu einem Datum
- time-of-day Konstanten werden wie normale times definiert (nur ohne Datum)
- Zum Beispiel

06:00

or

00:00:00

or

23:20:00.122



## Komplexes Beispiel

```
usual_time_of_order := read {...};
time_of_order_weekend := usual_time_of_order + 1 hour;
...
if now is not Sunday then
    if pat_order occurred within 30 minutes surrounding
        usual_time_of_order then
        ...
    endif;
else
    if timeofday pat_order is within 45 minutes surrounding
        time_of_order_weekend then
        ...
    endif;
endif;
```



# Time-of-day status

- In Diskussion



## Internationalisierung: Hintergrund

- Üblicherweise sind textbasierte Nachrichten das Ergebnis eines MLMs
- Internationale Verwendung von MLMs
  - Unterschiedliche Länder erfordern Nachrichten in der jeweiligen Landessprache
  - Einige Länder erfordern unter Umständen sogar mehr als eine Sprache (US, Belgien,...)



## Mehrsprachige Nachrichten mit Version 2.x

- Workaround: für jede Sprache ein MLM mit den Texten
  - Keine Logik, nur Text
  - Durch Logik-MLMs aufgerufen
  - Parameter/Result Schnittstelle
- Schwierig zu warten
  - Große Wissensbasen werden aufgebläht, wenn zusätzliche Sprach-MLMs verwendet werden
- Uneinheitlich zu nutzen
  - Jeder Autor/Anbieter würde eigene Schnittstellen verwenden



## Mehrsprachige MLMs mit 2.6

- Zusätzliche Kategorie  
“resources”

```
maintenanance:  
  [...]  
;;  
library:  
  [...]  
;;  
knowledge:  
  [...]  
;;  
resources:  
  [...]  
;;
```



## Resources category

- Zwei neue slots
- Language slots
  - Mindestens einer erforderlich
  - 2-Zeichen Sprachcode (ISO 639-1)
  - Liste von Bezeichner/Wert Paaren
    - Bezeichner: Arden Syntax Term
    - Wert: Arden Syntax String
    - Könnte **Unicode** erfordern

```
resources:  
  default: en ;;  
  language: en  
    'hello': "Hello World"  
    'language': "English"  
  ;;  
  language: es  
    'hello': "Hola mundo"  
    'language': "Españiol"  
  ;;  
  language: de  
    'hello': "Hallo Welt"  
    'language': "Deutsch"  
  ;;
```



## ‘Localized’ operator

- Operator zur Verwendung im data slot, logic slot und action slot
  - `<n:String> := localized <n:Term>`
  - Liefert den Wert des gültigen language-slots zurück, dessen Bezeichner dem Term gleicht
- Beispiele:
  - Deutsche Umgebung: `“Hallo Welt” := localized ‘hello’;`
  - Englische Umgebung: `“Hello World” := localized ‘hello’;`
  - Unbekannt: `“Hello World” := localized ‘hello’;`



## Beispiel

```
knowledge: [...]  
  logic: msg := localized 'penicillin' || penicillin_allergy;  
        msg2 := format creat_cl with localized 'creat' ;  
  [...]  
resources:  
  default: en;  
  language: en  
    'penicillin': "Caution, the patient has the following allergy  
to penicillin documented: "  
    'creat': "The patient's calculated creatinine clearance is  
%f ml/min."  
  ;;  
  language: de  
    'penicillin': "Vorsicht, zu dem Patienten wurde die folgende  
Penicillin-Allergie dokumentiert: "  
    'creat': "Die berechnete Kreatinin-Clearance des Patienten  
beträgt %f ml/min."  
  ;;
```



## Beispiel

- Ergebnis in einer deutschen Sprachumgebung:
  - Msg := „Vorsicht, zu dem Patienten wurde die folgende Penicillin-Allergie dokumentiert: ...“
  - Msg2 := „Die berechnete Kreatinin-Clearance des Patienten beträgt 12,5 ml/min.“
- Ergebnis in einer englischen Sprachumgebung :
  - Msg := „Caution, the patient has the following allergy to penicillin documented: ...“
  - Msg2 := „The patient's calculated creatinine clearance is 12.5 ml/min.“



## Teil 2: Einführung in die Arden Syntax

Anwendungsbeispiele



## Entwicklungszyklus

- Identifikation des zu lösenden Problems
- Befragen eines Experten
- Beschreibung des Entscheidungsprozesses
- MLM schreiben
- MLM testen und verbessern bis
  - sowohl Autor als auch Experte zufrieden sind



## Beispiel: Prüfen von Dokumentation

- Tagsüber soll 2 Stunden nach stationärer Aufnahme eines Patienten auf der Station der Inneren Medizin die Anamnese erhoben sein
  - event := Fallstatuswechsel, Aufnahme, etc.
  - read := station des Patienten (aus Event)
  - read := Dokumente des Patienten



# Beispiel: Prüfen von Dokumentation

**data:**

```
evt1 := event { /* Aufnahme, Verlegung, Fallstatuswechsel */ };
```

**evoke:**

```
2 hours after time of evt1;;
```

**logic:**



## Beispiel: Prüfen von Dokumentation

data:

```
evt1 := event { /* Aufnahme, Verlegung, Fallstatuswechsel */ };  
(pid, station) := read {  
  /* patient und station, beispielsweise aus Event-Context */  
};  
anamnese_id := read latest { /* Anamnese-Daten des Patienten */ };
```

evoke:

```
2 hours after time of evt1;;
```

logic:



## Beispiel: Prüfen von Dokumentation

data:

```
    evt1 := event { /* Aufnahme, Verlegung, Fallstatuswechsel */ };  
    (pid, station) := read {  
        /* patient und station, beispielsweise aus Event-Context */  
    };  
    anamnese_id := read latest { /* Anamnese-Daten des Patienten */ };
```

evoke:

```
    2 hours after time of evt1;;
```

logic:

```
    if station is equal "SINT1" then
```

```
    endif;;
```



## Beispiel: Prüfen von Dokumentation

data:

```
evt1 := event { /* Aufnahme, Verlegung, Fallstatuswechsel */ };  
(pid, station) := read {  
    /* patient und station, beispielsweise aus Event-Context */  
};  
anamnese_id := read latest { /* Anamnese-Daten des Patienten */ };
```

evoke:

```
2 hours after time of evt1;;
```

logic:

```
if station is equal "SINT1" then  
    if no anamnese_id then  
        msg := "Für den Patienten sollte spätestens zwei Stunden nach der Aufnahme die  
        Anamnese erhoben werden.";  
        conclude true;  
    endif;  
endif;;
```

action:

```
write msg;;
```



## Beispiel: Einfaches Rechenbeispiel

- Der BMI berechnet sich aus dem Körpergewicht [kg] dividiert durch das Quadrat der Körpergröße [m<sup>2</sup>]. Die Formel lautet:
  - $BMI = \text{Körpergewicht} : (\text{Körpergröße in m})^2$ . Die Einheit des BMI ist demnach kg/m<sup>2</sup>.
  - Dies bedeutet, eine Person mit einer Körpergröße von 160 cm und einem Körpergewicht von 60 kg hat einen BMI von 23,4
- Der "wünschenswerte" BMI hängt vom Alter ab. Folgende Tabelle zeigt BMI-Werte für verschiedene Altersgruppen:

Alter	BMI
19-24 Jahre	19-24
25-34 Jahre	20-25
35-44 Jahre	21-26
45-54 Jahre	22-27
55-64 Jahre	23-28
>64 Jahre	24-29



## Beispiel: Entwicklung eines BMI MLMs

- BMI-Klassifikation (nach DGE, Ernährungsbericht 1992):

Klassifikation	m	f
1: Untergewicht	<20	<19
2: Normalgewicht	20-25	19-24
3: Übergewicht	25-30	24-30
4: Adipositas	30-40	30-40
5: massive Adipositas	>40	>40

- Die Klassifikationen Adipositas und schwere Adipositas müssen auf jeden Fall behandelt werden, da ein erhöhtes Risiko für Diabetes und Herzerkrankungen besteht!
- Zu beachten ist, daß sportliche Personen häufig durch den Muskelaufbau ebenfalls ihren BMI um 25 haben. In diesem Fall liegt natürlich kein Übergewicht vor.



## Schritt 1: Schnittstelle

- Aus Datenbank lesen oder als Parameter  
weight := argument; /\* kg \*/  
height := argument; /\* cm \*/  
age := argument; /\* years \*/
- Prüfen, ob Werte vorhanden und Zahlen  
if all ((weight, height, age) are number) then...



## Schritt 2: der BMI und Klassifikation

- $\text{bmi} := \text{weight} / (\text{height}/100)**2;$

Klassifikation	m	f
1: Untergewicht	<20	<19
2: Normalgewicht	20-25	19-24
3: Übergewicht	25-30	24-30
4: Adipositas	30-40	30-40
5: massive Adipositas	>40	>40

- if  $\text{bmi} > 40$  then  
      $\text{classification\_cd} := 5;$   
      $\text{classification} := \text{"massive Adipositas"};$   
   elseif  $\text{bmi}$  is within 30 to 40 then...



## Schritt 3: Zusatztexte

- Die Klassifikationen Adipositas und schwere Adipositas müssen auf jeden Fall behandelt werden, da ein erhöhtes Risiko für Diabetes und Herzerkrankungen besteht!
- Zu beachten ist, daß sportliche Personen häufig durch den Muskelaufbau ebenfalls ihren BMI um 25 haben. In diesem Fall liegt natürlich kein Übergewicht vor.
- `interpretation := (bmi, classification) formatted with "Basierend auf den eingegebenen Informationen wurde ein BMI von %2.2f ermittelt. Dieser Wert entspricht einer Klassifikation von '%s' . ";`

```

if classification cd is in (4,5) then
    interpretation := interpretation || "
Die Klassifikationen Adipositas und schwere Adipositas ...";
elseif bmi is within 24 to 26 then
    interpretation := interpretation || "
Bitte beachten Sie, daß sportliche Personen ...";
endif;

```



## Schritt 4: Feinheiten

- Alter und Geschlecht berücksichtigen



## Beispiel: Kriterien in Leitlinien prüfen

- Wurde über die letzten drei Wochen regelmässig, mindestens einmal die Woche Blutzucker gemessen?
- `BG := read {...}`

```
BG_w1 := BG where they occurred within last 1 week;
```

```
BG_w2 := BG where they occurred within now - 2 weeks to now - 1 week;
```

```
BG_w3 := BG where they occurred within now - 3 weeks to now - 2 weeks;
```

```
if no BG_w1 or no BG_w2 or no BG_w3 then...
```

```
...
```

```
endif
```



# Anwendungen

- Klassische Aufgaben sind
  - Warnings: beispielsweise bei Aufträgen
  - Remainder: zeitgetriggerte Prüfungen
- Realisierbar sind darüber hinaus
  - Rechenregeln
  - Plausibilitätsprüfungen
  - Bedingungen und andere Elemente von Leitlinien
  - Expertensysteme zur Diagnose-Unterstützung
- Für medizinische und organisatorische Problemstellungen



Vielen Dank

[sven.tiffe@gwi-ag.com](mailto:sven.tiffe@gwi-ag.com)

**Credits:** M.Criegee-Rieck, A. Neubert, Universität Erlangen:  
Informationen zu UAW

R. Jenders, M. Sailors, Arden Syntax SIG:  
Internationales Tutorial zur Arden Syntax

