

Arden Syntax advanced

Educational material, part 4

Medexter Healthcare
Borschkegasse 7/5
A-1090 Vienna

www.medexter.com

www.meduniwien.ac.at/kpa (academic)

```
logic:
  let hmi be weight / (size ** 2); // BMI
  age := currenttime - birth; // AGE
  if the age is less than 19 years then
    classification := null;
  elseif the hmi is less than 18.5 then
    classification := localized 'under';
  elseif the hmi is less than 25 then
    classification := null; // BMI normal range
  else
    let the classification be localized 'over';
  endif;
```

Better care, patient safety, and quality assurance by Medexter, Vienna, Austria

Arden Syntax objects

- **Purpose**

- Allow for the logical grouping of data elements.
- May contain multiple named attributes, each of which may contain any valid Arden type (including lists or objects).
- Allows for complex data structures to be manipulated by an MLM (e.g., lists within lists) which would otherwise not be possible.

- **Definition**

- **Declaration:** An object is declared by the **OBJECT** keyword.

```
MedicationDose := OBJECT [Medication, Dose, Status];
```

- **Instantiation:** Objects are created and with the **NEW** keyword.

```
medDose1 := NEW MedicationDose; // empty object  
medDose2 := NEW MedicationDose WITH "Ampicillin", "500mg", "Active";  
medDose3 := NEW MedicationDose WITH "Amoxicillin", "500mg", "Active";
```

Arden Syntax objects (continued)

- **Reading values**

- **Access:** By using the dot (.) operator, object fields can be accessed.

```
"Ampicillin" := medDose2.Medication;
```

```
("Ampicillin", "Amoxicillin") := (medDose2, medDose3).Medication;
```

- **Read As:** The **READ AS** statement queries an external data source (e.g., a patient database) and returns a single list of objects. The object type is a compulsory part of the statement, and should have been declared previously.

```
medDoses := READ AS MedicationDose  
{  
  "select med, dosage, status from client"  
};
```

Arden Syntax objects (continued)

- **Modifying values**

- **Attribute assignment:** allows for the assignment to individual attributes of an object.

```
medication := NEW MedicationDose;  
medication.Dose := "500mg";  
medication.Status := "Active";
```

- **Enhanced assignment:** Any expression that ends with a **dot** operation or **element** operation may be placed on the left hand side of an assignment.

```
medList[n].Dose := "300mg";
```

- **Object explication**

- **Extract attribute names:** Returns a list containing the attribute names of the object argument.

```
"Medication", "Dose", "Status" := EXTRACT ATTRIBUTE NAMES medication;
```

Arden Syntax objects – Example

```
22 knowledge:
23     type: data_driven;;
24     data:
25         Patient := object [
26             temperature,
27             heartRate,
28             respRate,
29             PaCO2,
30             WBcellCount,
31             immatureBand
32         ];
33
34     allValues := Argument;
35
36     sirsPat := new Patient with allValues[1], allValues[2], allValues[3],
37         allValues[4], allValues[5], allValues[6];
38     ;;
```

Arden Syntax objects – Example

```
41  logic:
42      //Start - Checking SIRS criteria
43      counter := 0;
44
45      if sirsPat.temperature is greater than 38 or
46         sirsPat.temperature is less than 36 then
47         counter:= counter + 1;
48      endif;
49
50      if sirsPat.heartRate is greater than 90 then
51         counter:= counter + 1;
52      endif;
53
54      if sirsPat.respRate is greater than 20 or
55         sirsPat.PaCO2 is less than 32 then
56         counter:= counter + 1;
57      endif;
58
59      if sirsPat.WBcellCount is greater than 12000 or
60         sirsPat.WBcellCount is less than 4000 or
61         sirsPat.immatureBand is greater than 10 then
62         counter:= counter + 1;
63      endif;
64
65      if counter is greater than or equal 2 then
66         notification:= localized 'SIRS';
67         conclude true;
68      endif;
69      //End - Checking SIRS criteria
70      ;;
71  action:
72      return (sirsPat, notification);
73      ;;
```

Curly braces expressions

- **Purpose**

- Signify institution-specific definitions and mappings
- Allow for data and function access outside the Arden Syntax
- Enable interaction with the host system

- **Syntax**

- **Read statement:** Reads data from the host system. It is used to isolate those parts of a database query that are specific to an institution from those parts that are universal.

```
bodyTemp := READ {select temp from patResults where patID = 'Jeroen'};
```

- **Event statement:** assigns an institution-specific event definition to a variable. This variable is used in the evoke slot, as part of the call statement to call other MLMs.

```
incBodyTempEvent := EVENT {increased body temperature};
```

Curly braces expressions (continued)

- **Syntax (continued)**

- **Message statement:** assigns an institution-specific message (for example, an alert) to a variable. It allows an institution to write coded messages in a patient database.

```
message1 := MESSAGE {increased body temperature};
```

- **Destination statement:** assigns an institution-specific destination to a variable. It allows one to write a message to an institution-specific destination, e.g., an email address.

```
destination1 := DESTINATION {email: support@medexter.com};
```

- **Interface statement:** assigns an institution-specific destination to a variable. It allows one to write a message to an institution-specific destination, e.g., an email address.

```
c_func := INTERFACE {char* c_func(char*, char*)};
```

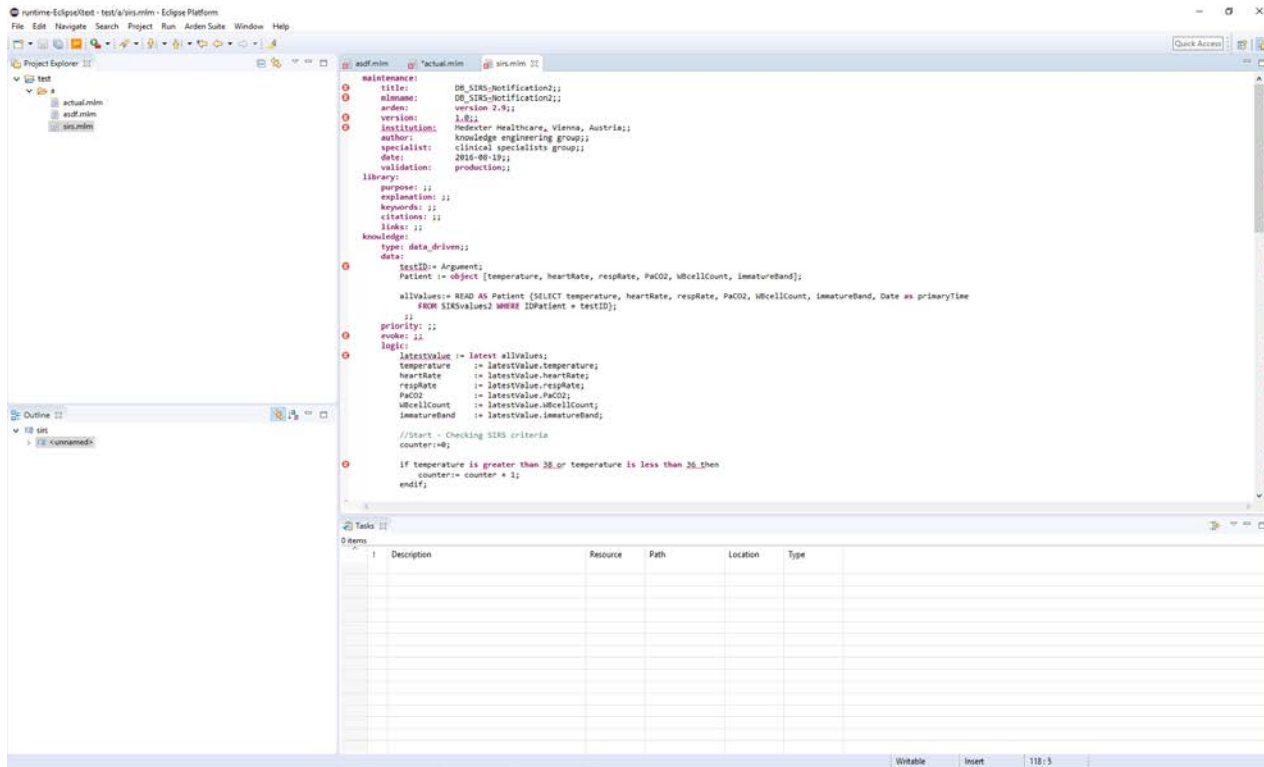
Curly braces expressions – Read Example

```
17 knowledge:
18   type: data_driven;;
19   data:
20     testID := Argument;
21     (temperature,heartRate,respRate,PaCO2,WBcellCount,immatureBand)
22     := READ {SELECT temperature, heartRate, respRate, PaCO2, WBcellCount,
23             immatureBand FROM sirvalues WHERE IDPatient = testID};
24     ;;
```

- This **assignment** statement assigns the result of the read statement (using mapping clause "SELECT ... FROM ... WHERE IDPatient = testID") to a list of variables.
 - IDPatient is a variable that contains the patient ID currently in use and is substituted before execution of the mapping clause
 - The content of the **curly brace expressions** must be evaluated by the host system and its syntax is not part of the Arden Syntax
-

Practical Part II

Eclipse-based ArdenSuite IDE (under development)



The screenshot displays the Eclipse IDE interface for the ArdenSuite project. The main window shows the 'actual.mim' file with the following Arden syntax:

```
maintenance:
  title: DE_SIRS-Notification;;
  sname: DE_SIRS-Notification;;
  arden: version 2.9;;
  version: 1.0;;
  institution: Medexter Healthcare, Vienna, Austria;;
  author: knowledge engineering group;;
  specialist: clinical specialists group;;
  date: 2016-08-19;;
  validation: production;;

library:
  purpose: ;;
  explanation: ;;
  keywords: ;;
  citations: ;;
  links: ;;
  knowledge:
    type: data_driven;;
    data:
      testID= Argument;
      Patient := object [temperature, heartRate, respRate, PaCO2, WbcellCount, ImmatureBand];
      allValues:= READ AS Patient [SELECT temperature, heartRate, respRate, PaCO2, WbcellCount, ImmatureBand, Date as primary/line
      FROM SIRSValues2 WHERE IDPatient = testID];
      ;;
      priority: ;;
      evoke: ;;
      logic:
        latestValue := latest allValues;
        temperature := latestValue.temperature;
        heartRate := latestValue.heartRate;
        respRate := latestValue.respRate;
        PaCO2 := latestValue.PaCO2;
        WbcellCount := latestValue.WbcellCount;
        ImmatureBand := latestValue.ImmatureBand;

        //Start - Checking SIRS criteria
        counter:=0;

        if temperature is greater than 38 or temperature is less than 36 then
          counter:= counter + 1;
        endif;
```

The 'Tasks' table at the bottom is currently empty:

| ID | Description | Resource | Path | Location | Type |
|----|-------------|----------|------|----------|------|
| | | | | | |