9.10.8, 9.10.10, 9.10.12, 9.10.14, 9.10.16, 9.10.18 Adding "Year [Of]", "Month [Of]", "Day [Of]", "Hour [Of]", "Minute [Of]", "Second [Of]".

It was very helpful to see these new operators separated into their own section numbers. It was also helpful to see these operators next to the "Extract" operators. Consequently, I am wondering if it would be better to put the new operators on the right-hand-side of the assignment. It is little confusing when the operator is on the left-hand-side of the assignment. Would the following "Replace" syntax be possible to use instead? For example,

2011-02-28 11:15:07 := Replace Year of 1990-02-28 11:15:07 with 2011;
1990-06-28 11:15:07 := Replace Month of 1990-02-28 11:15:07 with 6;
1990-02-08 11:15:07 := Replace Day of 1990-02-28 11:15:07 with 8;
1990-02-28 12:15:07 := Replace Hour of 1990-02-28 11:15:07 with 12;
1990-02-28 11:17:07 := Replace Minute of 1990-02-28 11:15:07 with 17;
1990-02-28 11:15:19 := Replace Second of 1990-02-28 11:15:07 with 19;

9.12.5

If we're including the statement below, we'll need some examples. Does it mean the median operator can use the new SORT . . .USING, or just the USING modifier? I don't see how this would change the value returned.

The median operator can also be extended by the using modifier as defined for the sort operator (see 9.2.4) to allow more complex calculations of the median.

This same question applies to sections 9.12.9, 9.12.10, 9.12.17, 9.14.2, 9.14.3, 9.14.11, 9.14.12

Our intention was to make handling of lists of objects easier. Each of the above mentioned operators may use the word "using" to modify its behaviour.

E.g., we can get the oldest person from a list of persons (represented by objects) by the following expression:

<n:any-type> := max <n:any-type> using it.age;

Note that in this case the person (the object) is returned, not only the maximum age.

For the operators "earliest" and "latest" the "using" modifier may be used to refer to a time other than the primary time. E.g., to get the earliest born person we may write:

<n:any-type> := earliest <n:any-type> using it.birthday;

Other example:

```
90 := max (0, 30, 90, 120, 200) using sinus of it; // will return the value of the
                                                   // list with the maximum sinus value
```

9.13.4 Index Of . . . From

Since this operator is returning one or more indexes, not a subset of data, preserving primary times seem unnecessary, and is inconsistent with section 9.13.3. Similarly,

existing section 9.13.3 seems to be returning null, not an empty string, when there is no match. We need to be consistent.

Can this sentence be clarified “It returns the index respectively a list of the indices of the occurrences of the given data value within the provided list.” ?

Does the sentence means this “It returns the respective index of the list that matches the given data value within the provided list.”?

The intention of this operator was to easily search a list for data-elements. E.g., if we want to get all occurrences of the value 2 in a list, we may write

(1,3,5) := Index Of 2 From (2, 9, 2, 7, 2);

9.13.5 and **9.13.6** Adding "At Least... From" and "At Most...From".

I think these new operators are going to be very useful. However, they can easily be confused with the operators that are for indexes. Would it be possible to introduce a new word like "AreTrue" or "IsTrue" to help the user understand that a list is being searched for Boolean values? For example:

TRUE := AT LEAST 2 AreTrue FROM (TRUE, TRUE, FALSE, FALSE)
TRUE := AT LEAST 1 IsTrue FROM (TRUE, TRUE, FALSE, FALSE)
TRUE:= AT MOST 2 AreTrue FROM (TRUE, TRUE, FALSE, FALSE)
FALSE:= AT MOST 1 IsTrue FROM (TRUE, TRUE, FALSE, FALSE)

The operators “ANY”, “ALL”, and “No” (9.12.13-15) also don’t need the words “AreTrue” or “IsTrue”. If we decide to add these words here, we have to add them to those operators as well.

9.16.18 As Time

Is it accurate to rephrase this statement

The common use for this will be to convert a string which contains a valid time representation e.g., "1999-12-12" into the represented time with respect to the ISO 8601:1988(E).

to this?

The common use for this is to convert a string containing a valid date/time format as described in ISO 8601:1988(E), .e.g., "1999-12-12" or "1999-12-12T13:41", into a time.

10.2.3 and **11.2.12** and **12.2.4** Adding a "Switch-Case"

Are "Switch-Case" and "Switch-Case-Default" another way to handle If-Then-Elseif and If-Then-Elseif-Else?

Can a simple example be provided for each, so that the reader can understand their use?

A simple example is:

```
switch inVal
    case 1
        returnVal := 7;
    case 2
        returnVal := 9;
    default
        returnVal := 0; //error State
endswitch;
```

This example will set the variable “returnVal” to 7 if the value of the incoming variable “inVal” is equal to 1, to 9 if the value of the incoming variable “inVal” is equal to 2 and to 0 otherwise.

10.2.6 adding "Break" to the While Loop

Since break is suggested new functionality, it needs to be added to Reserved Words. It also needs its' own short descriptive section in the standard, with an example of how it is used in a loop..

It would be useful to allow "Break" in other parts of the MLM's where code is being executed.

Allowing break points in the Data, Logic, and Action slots would help the person debugging the MLM to see what the variable values are at specific places during the MLM's execution.

“Break“ is suggested to be used to stop the execution of a loop statement. E.g.,

```
num:= 1;
/* Checks each allergen in the medications and stops if patient is allergic to it
*/
while num <= (count med_allergen) do
        allergen:= last(first num from med_allergens);
        allergy_found:= (patient_allergies = allergen);
        /* be returned to the calling MLM */
        If any allergy_found then
                Break;  // execution of the while-loop will stop immediately
        endif;
        /* Increments the counter that is used to stop the while-loop */
        num:= num + 1 ;
enddo;
```

10.2.9 WRITE

Permitting the WRITE statement almost anywhere in an MLM is good for debugging, but is a major change – a rule could generate a message without concluding true. This warrants some lively discussion.

I am wondering why a Write statement is needed in the Logic and Data slots. Also, would the Write statement be executed immediately in the Logic and Data slots? Or

would it only be executed when the MLM concludes TRUE? It makes a difference because an immediate execution could be a problem when a user viewing the alert message decides to cancel the action (such as adding a new medication order) that caused the alert to pop up. For example, the Write could store to the database, but the alert is not stored to the database when the user cancels.