ICT for Health Science Research
A. Shabo (Shvo) et al. (Eds.)

doi:10.3233/978-1-61499-959-1-36

Complementing Medical Records with Precalculated Data Items to Facilitate Decision Support and Phenotyping

Stefan KRAUS[a,1] and Hans-Ulrich PROKOSCH [a]
[a] *Medical Informatics, University Erlangen-Nürnberg, Erlangen, Germany.*

Abstract. The Arden Syntax is a standard for clinical decision support functions in the form of Medical Logic Modules (MLMs). While the data type system of the early versions was limited to flat lists, later versions introduced an object type, supporting complex data structures, even up to entire electronic medical records (EMRs). Such objects are static insofar as their structure cannot be modified at MLM runtime. University Hospital Erlangen uses an experimental Arden Syntax version termed PLAIN, which provides an integrated mapper for arbitrary data structures, including entire EMRs. To facilitate knowledge encoding and reduce MLM complexity, we searched for a way to complement patient records with precalculated data items. We modified the object data type in two ways. The first was to include a statement for the explicit creation of new attributes; the second was to implicitly create an attribute whenever a value is assigned to a previously non-existing attribute. As a proof of concept, we complemented the ventilation section of every accessed EMR with a patient-individual recommendation for the expiratory tidal volume. A means to extend the structure of an object at runtime provides several advantages. The precalculated data items need no longer be calculated by the MLMs themselves, which reduces complexity and facilitates code maintenance. This might be beneficial not only for clinical decision support, but also with respect to the use of Arden Syntax language constructs for phenotyping queries, as well as with respect to the frequently required preprocessing of EMR data.

Keywords. Electronic medical records, Arden Syntax, clinical decision support.

1. Introduction

University Hospital Erlangen (UHER) uses multiple self-written clinical decision support (CDS) functions in the intensive care unit (ICU) setting [1]. These CDS functions are based on the Arden Syntax for Medical Logic Systems, a Health Level 7 standard for encoding and sharing knowledge in the form of Medical Logic Modules (MLMs) [2]. An MLM corresponds to a condition-action rule, whereas the actual decision logic of the rule is encoded by means of procedural programming language constructs. In version 2.5, the Arden Syntax introduced an object data type that enables MLMs to map complex data structures, in principle even up to entire electronic medical records (EMRs). The resulting objects, however, are static insofar as their structure cannot be modified at MLM runtime. Consequently, it is not possible to extend the structure of mapped EMRs

[1] Corresponding Author: stefan.d.kraus@fau.de

in order to complement them with precalculated data prior to the actual decision-making, although such a strategy might have advantages in some specific use cases.

The local experiences with many characteristics of the Arden Syntax standard were so promising that it appeared as a suitable starting point for a common clinical application language, intended to cover multiple medical sub-domains. The prototype of the resulting language, which is currently evaluated in clinical routine at UHER, has been termed **P**rogramming **L**anguage, **A**rden-**IN**spired (PLAIN) in order to emphasize its origin [3]. The already existing MLMs have been ported to the new platform, without changing the actual decision logic.

In contrast to the Arden Syntax, PLAIN provides an integrated data mapper and a straightforward XML format termed PLAIN Data Markup Language (PDML) to facilitate patient data access. At the largest ICU of UHER, we use this feature to map entire EMRs onto congruent objects in a single step. We adjusted multiple language features to enable the complementation of EMRs with precalculated medically relevant parameters, such as the patient-individual recommendation of the tidal volume for ventilated patients, or a time series of clinical score values calculated by specific MLMs. The purpose of this paper is to describe these modifications, and to discuss the benefits of our approach.

2. Methods

In the Arden Syntax standard, the creation of an object at runtime within an MLM presupposes that its structure is defined in the source code. For this purpose, it provides an OBJECT statement ([4], 11.2.17), which lists all the attributes of a specific object type. Instances of such a type are created at runtime using the NEW statement ([4], 11.2.18), as shown in example (A). PLAIN uses an alternative approach, creating objects without previous declaration by means of the JavaScript Object Notation, as shown in (B). The objects created by both examples are exactly the same.

```
(A) patient     := OBJECT [name, case] ;
    testpatient := NEW patient WITH "John Doe", 12345 ;

(B) testpatient := { name: "John Doe",
                     case: 12345 } ;
```

The Arden Syntax standard does not provide language constructs to extend an object at runtime with additional attributes; assigning a value to a non-existing attribute has no effect ([4], 10.2.1.1), and PLAIN initially adopted this behavior. To provide a means to adjust the structure of objects at runtime, we implemented two different approaches. The first one was to integrate an EXTEND statement that extends an existing object with a specific attribute name. The second one was to create an attribute automatically whenever a value is assigned to a previously non-existent attribute, like for example in the JavaScript programming language. As an additional feature for the frequently required preprocessing of clinical time series data, we implemented a way to create attributes of which the name is not available until runtime, by enabling the use of the ATTRIBUTE FROM operator ([4], 9.18.4) on the left side of an assign statement.

PLAIN provides a means to use MLMs as user-defined functions (UDFs) and call them in arbitrary expressions by prefixing their name with "@". We created an UDF

"@complement", which itself calls arbitrary other UDFs to complement an EMR with precalculated data items. To test our approach in clinical routine, we used the UDF "@ardsnet_recom" that calculates the patient-individual tidal volume recommendation as 6 ml per kg predicted body weight, requiring the patient's height and gender as input [5].

3. Results

The PLAIN processor has successfully been modified with respect to the object datatype to change its behavior from static to dynamic, resulting in a means to extend objects at runtime with additional attributes. The first approach, creating new attributes using the EXTEND statement, is outlined in example (C), which creates a new attribute "bedposition". The assignment to a new attribute can be done with the DOT operator ([4], 9.18.1), as shown in (D), or with the ATTRIBUTE FROM operator. The second approach does not require a previous EXTEND statement for the creation of an attribute. Instead, an assignment such as in (D) automatically creates an attribute if it does not already exist. Currently, both approaches are supported by PLAIN.

```
(C) EXTEND testpatient WITH "bedposition" ;

(D) testpatient.bedposition := "Bed 12" ;

(E) record := @patientrecord casenumber ;
    WRITE @table record.lab.creatinine ;

(F) record := @complement @patientrecord casenumber ;

(G) record.ventilation.ardsnet_recom := @ardsnet_recom {
        height: record.adt.height,
        gender: record.adt.gender
    } ;
```

Example (E) shows the most common mapping method of EMRs at UHER. The UDF "@patientrecord" returns a single EMR, retrieved from a network resource using the REST GET statement provided by PLAIN. This UDF takes a case number as the argument, which is automatically transmitted to any MLM by the commercial patient data management system used in our local ICU setting. The EMR is mapped onto a single object, which, in this example, is assigned to the variable name "record". Specific EMR items can be addressed with DOT operator expressions, such as in the WRITE statement in example (E), which uses the UDF "@table" to display the patient's creatinine values in the form of a HTML table. Example (F) is similar to example (E), but additionally makes use of the UDF "@complement". This UDF, in turn, contains the code shown in example (G), which extends the ventilation section of the EMR with the patient-individual recommendation for the expiratory tidal volume, calculated by the UDF "@ardsnet_recom". This example is based on the second approach, where non-existing attributes are automatically created on assignment.

4. Discussion

The Arden Syntax has undergone a long evolution [6]. The early versions were intentionally kept simple for the sake of understandability [7]. Since then, multiple additional features have been introduced, and some of them have considerably extended the capabilities of the standard. However, the approach to interoperability with the clinical information system has remained unchanged so far, namely that each institution must develop its own approach to data mapping and interoperability in general. After years of experience with the use of MLMs at UHER, we concluded that, in our local ICU setting, the best approach to access patient data is to map entire EMRs. In the early versions of the Arden Syntax, such an approach was not yet possible, since the data type system was limited to flat lists. In version 2.5, the object data type was introduced. The Arden Syntax specification calls this introduction an evolutionary step, considerably increasing the capabilities of the standard ([4], A6.1), and our own experiences corroborate this point of view. Our first approach to map complex data, i.e. data with a structure beyond list or tables, in clinical routine, referred to tree-structured microbiology results [8].

When UHER migrated from the Arden Syntax to PLAIN in 2017, we started to map EMRs onto congruent objects using its integrated PDML mapper. Such an EMR does not necessarily contain all data items documented for a specific patient. Currently, PDML-encoded EMRs at UHER contain the superset of all data elements required by the full set of MLMs, with a tendency towards significant expansion. This approach considerably simplified interoperability at UHER, but the fact that objects were static turned out to be a limitation in some use cases. The demand for flexible objects relates to the fact that MLMs at UHER are rather untypical. The traditional use case for MLMs is the data-driven monitoring of clinical events, whereas the decision logic of such MLMs is usually straightforward. At UHER, in contrast, we use MLMs mostly for user-driven information retrieval at the point of care [1]. These MLMs are often far more complex than typical MLMs, with a considerably larger code size. If frequently required data elements are already provided in the EMR, the MLMs do not have to calculate them on their own, which reduces complexity and thus simplifies maintenance of the code.

Another benefit of flexible objects relates to preprocessing of patient data, like in the context of clinical score calculations in the form of a time series. The input values of such calculations often do not match with respect to their quantity and timestamps, requiring both the aggregation of values and the harmonization of timestamps. In various use cases, flexible objects can considerably facilitate aggregation and harmonization of raw clinical data, since they provide a generic means to transform the input to customized data structures at MLM runtime, a task whose practical importance is often underestimated. Whether the EXTEND statement should be continued is currently the subject of discussion. From a technical point of view, it is not necessary, but it might make the code more understandable, since it accentuates the creation of additional attributes.

The potential benefits of complementing patient records with precalculated data items are also under research in the context of the first use case of the Medical Informatics in Research and Care in University Medicine (MIRACUM) project [9]. Currently, there is an investigation at our local chair of medical informatics to transform patient data from the Observational Medical Outcomes Partnership (OMOP) common data model to PDML-encoded EMRs, in order to apply language constructs of the Arden Syntax for patient phenotyping, as outlined in an earlier investigation [10]. For this

purpose, we investigate the suitability of a customized PLAIN version as a query language for MIRACUM that processes sequences of EMRs. A phenotyping query may consider data items that are not provided by an EMR and therefore must be calculated at query runtime. Complementing EMRs with precalculated data items may thus be advantageous with respect to query complexity, comprehensibility, and execution time. To give an example, a CDS function or a phenotyping query may consider a patients' liver condition. The progression of the liver condition can be assessed using a time series of model for end-stage liver disease (MELD) values [11]. The MELD, however, is currently not included in the routine documentation at UHER, but calculated during query execution. This, in turn, requires the relatively complex preprocessing of the time series data of multiple laboratory results within the MLM. Complementing the EMR in advance with the MELD time series relieves the MLM author of the need to calculate it at runtime. The same applies to many other clinical scores and medical formulae, the precalculation of which could significantly simplify and accelerate calculation-intensive phenotyping queries.

References

[1] Kraus S, Castellanos I, Toddenroth D, Prokosch H-U, Burkle T. Integrating Arden-Syntax-based clinical decision support with extended presentation formats into a commercial patient data management system. J Clin Monit Comput. 2014;28:465–73. doi:10.1007/s10877-013-9430-0.

[2] Hripcsak G, Ludemann P, Pryor TA, Wigertz OB, Clayton PD. Rationale for the Arden Syntax. Comput Biomed Res. 1994;27:291–324.

[3] Kraus S. Generalizing the Arden Syntax to a Common Clinical Application Language. Stud Health Technol Inform. 2018;247:675–9.

[4] Health Level Seven International: The Arden Syntax for Medical Logic systems, Version 2.8, 2012.

[5] Brower RG, Matthay MA, Morris A, Schoenfeld D, Thompson BT, Wheeler A. Ventilation with lower tidal volumes as compared with traditional tidal volumes for acute lung injury and the acute respiratory distress syndrome. N Engl J Med. 2000;342:1301–8. doi:10.1056/NEJM200005043421801.

[6] Jenders RA, Adlassnig K-P, Fehre K, Haug P. Evolution of the Arden Syntax: Key Technical Issues from the Standards Development Organization Perspective. Artif Intell Med 2016. doi:10.1016/j.artmed.2016.08.001.

[7] Hripcsak G, Wigertz OB, Clayton PD. Origins of the Arden Syntax. Artif Intell Med 2015. doi:10.1016/j.artmed.2015.05.006.

[8] Kraus S, Enders M, Prokosch H-U, Castellanos I, Lenz R, Sedlmayr M. Accessing complex patient data from Arden Syntax Medical Logic Modules. Artif Intell Med 2015. doi:10.1016/j.artmed.2015.09.003.

[9] Prokosch H-U, Acker T, Bernarding J, Binder H, Boeker M, Boerries M, et al. MIRACUM: Medical Informatics in Research and Care in University Medicine. Methods Inf Med. 2018;57:e82-e91. doi:10.3414/ME17-02-0025.

[10] Mate S, Castellanos I, Ganslandt T, Prokosch H-U, Kraus S. Standards-Based Procedural Phenotyping: The Arden Syntax on i2b2. Stud Health Technol Inform. 2017;243:37–41.

[11] Singal AK, Kamath PS. Model for End-stage Liver Disease. J Clin Exp Hepatol. 2013;3:50–60. doi:10.1016/j.jceh.2012.11.002.