Contents lists available at ScienceDirect

Artificial Intelligence in Medicine

journal homepage: www.elsevier.com/locate/aiim

Using Arden Syntax for the creation of a multi-patient surveillance dashboard

Stefan Kraus[a,b,*], Caroline Drescher[b], Martin Sedlmayr[b], Ixchel Castellanos[c], Hans-Ulrich Prokosch[a,b], Dennis Toddenroth[b]

[a] Center for Communication and Information Technology, University Hospital Erlangen, Glückstraße 11, 91054 Erlangen, Germany
[b] Department of Medical Informatics, Biometrics and Epidemiology, Chair of Medical Informatics, Friedrich-Alexander-University Erlangen-Nuremberg, Wetterkreuz 13, 91058 Erlangen-Tennenlohe, Germany
[c] Department of Anaesthesiology, University Hospital Erlangen, Krankenhausstraße 12, 91054 Erlangen, Germany

ARTICLE INFO

Keywords:
Arden Syntax
Medical Logic Modules
Multi-patient clinical decision support
Surveillance dashboard

ABSTRACT

Objective: Most practically deployed Arden-Syntax-based clinical decision support (CDS) modules process data from individual patients. The specification of Arden Syntax, however, would in principle also support multi-patient CDS. The patient data management system (PDMS) at our local intensive care units does not natively support patient overviews from customizable CDS routines, but local physicians indicated a demand for multi-patient tabular overviews of important clinical parameters such as key laboratory measurements. As our PDMS installation provides Arden Syntax support, we set out to explore the capability of Arden Syntax for multi-patient CDS by implementing a prototypical dashboard for visualizing laboratory findings from patient sets.
Methods and material: Our implementation leveraged the object data type, supported by later versions of Arden, which turned out to be serviceable for representing complex input data from several patients. For our prototype, we designed a modularized architecture that separates the definition of technical operations, in particular the control of the patient context, from the actual clinical knowledge. Individual Medical Logic Modules (MLMs) for processing single patient attributes could then be developed according to well-tried Arden Syntax conventions.
Results: We successfully implemented a working dashboard prototype entirely in Arden Syntax. The architecture consists of a controller MLM to handle the patient context, a presenter MLM to generate a dashboard view, and a set of traditional MLMs containing the clinical decision logic. Our prototype could be integrated into the graphical user interface of the local PDMS. We observed that with realistic input data the average execution time of about 200 ms for generating dashboard views attained applicable performance.
Conclusion: Our study demonstrated the general feasibility of creating multi-patient CDS routines in Arden Syntax. We believe that our prototypical dashboard also suggests that such implementations can be relatively easy, and may simultaneously hold promise for sharing dashboards between institutions and reusing elementary components for additional dashboards.

1. Introduction

Clinical decision support (CDS) functions based on Arden Syntax for Medical Logic Systems typically retain a single patient focus. This means that the underlying decision modules are generally limited to access data from a single patient. While many Arden Syntax installations are incapable of multi-patient CDS, the technical specifications of this standard actually leave the definition of input parameters for any interaction with a clinical information system to the particular institution [1,2]. A parameterization of the patient context would thus, in principle, allow for cross-patient database queries. Subsequent extensions of the initial type system of Arden Syntax, in particular the support for nested data structures provided by the object data type, appear specifically advantageous for processing more complex inputs such as for the description of patient sets. We therefore set out to demonstrate the fundamental capability of Arden Syntax for multi-patient CDS by implementing a prototypical graphical dashboard that simultaneously

* Corresponding author at: University Hospital Erlangen, Center for Communication and Information Technology, Glückstr. 11, 91054 Erlangen, Germany. Tel.: +49 9131 85 46474; fax: +49 9131 85 26754.
E-mail address: stefan.kraus@uk-erlangen.de (S. Kraus).

https://doi.org/10.1016/j.artmed.2015.09.009
0933-3657/© 2015 Elsevier B.V. All rights reserved.

visualizes laboratory results from a population of intensive care patients.

Arden Syntax is a standard for encoding and sharing medical knowledge [1]. Its classic field of application is the data-driven monitoring of clinical events [3]. In Arden Syntax, the knowledge base consists of independent modules, termed Medical Logic Modules (MLMs). An MLM typically contains sufficient knowledge to make a single clinical decision. Each MLM has a hierarchical structure, consisting of sections termed *categories*, which are themselves structured into subsections termed *slots*. For an introduction to this basic MLM structure see [4]. The execution of MLMs presupposes their transformation into a runnable format by an Arden compiler. The execution process is controlled by a runtime environment termed Arden engine. Whenever an MLM is evoked by a clinical event, such as the storage of a blood glucose value inside the patient record, several slots are subsequently executed. The DATA slot covers the readout of data from the patient record and maps them to variables. The LOGIC slot, containing the actual institution-independent medical knowledge, decides whether an action is to be performed or not. If so, the ACTION slot is executed. A typical action would be to send an alert notification to the clinical users.

In practice, operating MLMs to support clinical decisions requires access to medical patient records. This may be a problem, as widely accepted standards for the patient records themselves, or for the interfaces to access their contents, have not yet evolved. Therefore, the designers of Arden Syntax decided to leave the access to clinical data to the particular institution, providing a pair of curly braces to enclose the parameters required to specify the database access. The fact that any institution must find its own solution to enable data access is therefore commonly referred to as the “curly braces problem”, which entails that any institution has to implement a way to map data items from patient records to data types of Arden Syntax. The elementary data types of Arden Syntax include those typically found in patient records, such as numbers, strings or timestamps. Initially, flat lists were the only compound data type, which prohibit nested data structures. About a decade ago, however, the type system was complemented with the object data type, which supports nesting to a random depth [5].

While the potential utility of graphical summaries of individual patient records that “can be updated in real time” has been long recognized [6], a number of more recent publications [7–18] underline their increasing use for monitoring more comprehensive inputs. These automatically updated displays of time-varying metrics, also termed *digital dashboards*, have been proposed or applied for continuously summarizing and tracking diverse clinical or administrative parameters, such as department workflows [9,12–15], medication-related alerts [10,16,18], critical event reports [17], influenza surveillance measures [8], or preventative interventions [11,19,20]. A review of graphical methods for CDS purposes identified multi-patient visualizations among the more recent trends [21]. Previously published dashboard implementations had employed heterogeneous technical instruments [22], including Python [16], Ruby [14], PHP [8], and spreadsheets [12]. We are unaware of any previous implementation of a digital dashboard based on Arden Syntax. We assume that the particular density of electronically available intensive care unit (ICU) data, which has also fueled efforts to algorithmically induce new medical knowledge [23–25], specifically justifies investigations of clinical dashboards that process this rich information source.

The purpose of this study was to investigate the opportunities for implementing graphical dashboards by means of Arden Syntax MLMs. We thus studied the technical requirements of an Arden Syntax installation to support the construction of a dashboard. Moreover, we investigated in how far our proposed prototype may be shareable with other institutions, and which parts of its architecture may be reused for the construction of additional dashboards.

2. Methods and materials

University Hospital Erlangen is a tertiary care hospital with 1.400 beds. Starting in 2006, a commercial patient data management system (PDMS, Integrated Care Manager, ICM©, Dräger Medical, Lübeck, Germany) has been installed at nine local ICUs [26]. The PDMS does not natively provide a means to generate patient overviews with customizable CDS functions. This forced local clinicians to subsequently open multiple patient records in order to obtain an overview of clinically relevant data from patient populations, which is time consuming. Thus, clinicians expressed a need for the construction of dashboards that graphically summarize multi-patient overviews of selective clinical parameters processed by CDS functions. During a previously described research cooperation we had technically integrated a commercial Arden engine environment (ArdenSuite©, Medexter Healthcare, Vienna, Austria) with the PDMS [27], providing 30 self-written MLMs of different complexity for daily routine. Another Arden environment, written by one of the authors for research on the language constructs itself, was the technical platform for this study. Our investigation was based on Arden Syntax version 2.8 [2], which supports the object data type.

The first step of our study was to determine the practical requirements of the local clinical users in an interview, and to agree upon a basic graphical layout for the display of a dashboard prototype. They requested a tabular view of rectangular output elements, which we termed “signals”, with one table row for each patient, as shown in Fig. 1A. The first and last cell of each row display the patient name and bed position, while the remaining center cells contain signal outputs that are dynamically computed from patient facts (described below). For our prototype, we agreed on a set of clinical parameters to be processed and displayed, including three important laboratory values (blood glucose, leukocytes, and thrombocytes).

To limit the complexity of the implementation and to ensure an easy extensibility of our prototype, we decided to modularize our approach. In particular, we intended to separate the technical aspects from the representation of actual clinical knowledge as far as possible. We chose to separate the logic controlling the patient context in a “controller MLM”, which iterates over the case numbers of all patients that are currently on the ward. A complementary set of signal-specific MLMs defines the actual medical knowledge according to well-tried conventions in the sense that these signal MLMs can already assume a specific patient context (which is set by the controller MLM).

The next step was to design a data structure for the internal representation of the dashboard. We decided on a list of objects that

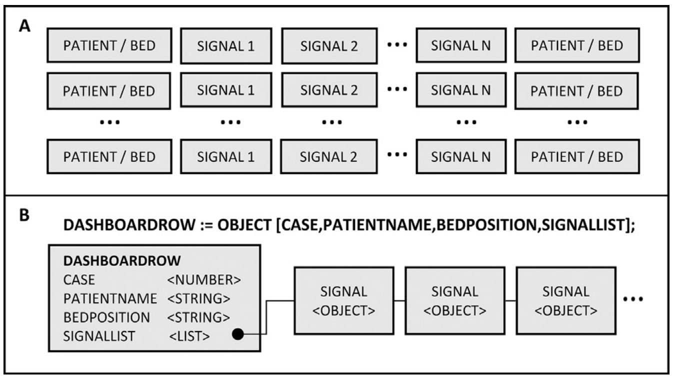
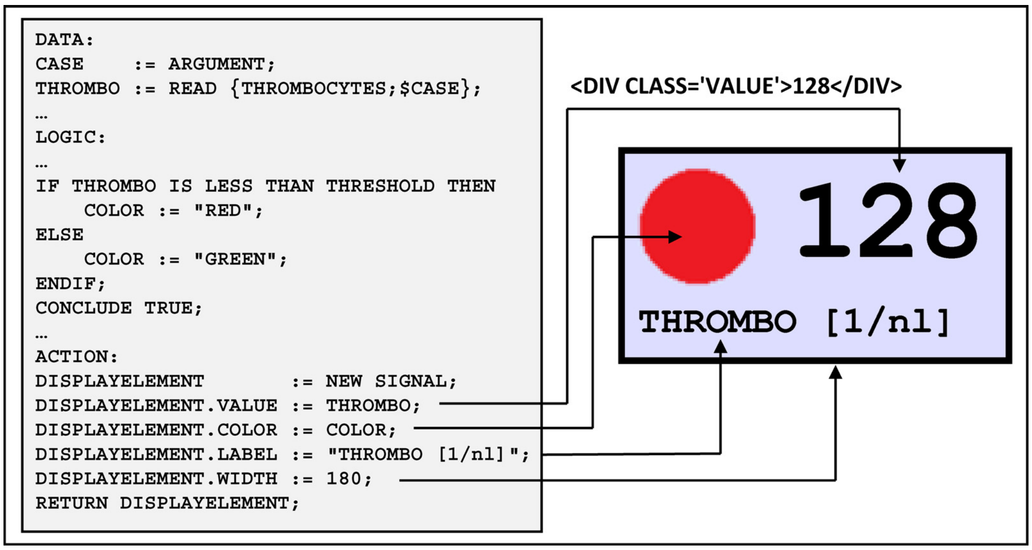


Fig. 1. Basic dashboard layout, with data types for internal representation ((A) Rectangular arrangement of output elements, (B) data structure representing one dashboard row).



```
DATA:
CASE    := ARGUMENT;
THROMBO := READ {THROMBOCYTES;$CASE};
…
LOGIC:
…
IF THROMBO IS LESS THAN THRESHOLD THEN
    COLOR := "RED";
ELSE
    COLOR := "GREEN";
ENDIF;
CONCLUDE TRUE;
…
ACTION:
DISPLAYELEMENT        := NEW SIGNAL;
DISPLAYELEMENT.VALUE  := THROMBO;
DISPLAYELEMENT.COLOR  := COLOR;
DISPLAYELEMENT.LABEL  := "THROMBO [1/nl]";
DISPLAYELEMENT.WIDTH  := 180;
RETURN DISPLAYELEMENT;
```

Fig. 2. Example for the creation of a signal element by the presenter MLM. Excerpt from the implementation of the signal MLM (left), and the corresponding detail of the graphical dashboard output as presented to the clinical user (right).

each represents a table row (data type DASHBOARDROW). Such an object contains four attributes as shown in Fig. 1B. The last attribute SIGNALLIST stores a list of objects of another type, each of them representing one signal (data type SIGNAL). An object of type SIGNAL contains a superset of all attributes collected during the interview, like a label, a color, a numeric value (like a calculated score value), a tooltip, and so on. Each signal MLM has to be called with a case number as the only argument and has to return exactly one signal object. Besides these two requirements, any signal MLM might be programmed according to established Arden Syntax conventions. The controller MLM as the centerpiece of our architecture creates a dashboard representation as a list of DASHBOARDROW objects, complete with case numbers, patient names and bed positions. The list of signals inside each row is initialized to an empty list. Signal objects are appended inside a loop, successively calling signal MLMs for each patient to complete the data structure row by row, signal by signal.

To display the dashboard within the PDMS user interface, we created a presenter MLM that is called by the controller MLM, using the internal data structure that represents the dashboard as an argument. The presenter MLM thereby generates a HyperText Markup Language (HTML) table whose signal-specific cells are individually formatted via Cascading Style Sheets (CSS). Fig. 2 demonstrates our presentation approach. For each attribute of the signal object, the presenter MLM creates an HTML element (DIV), and formats it according to a corresponding style sheet definition.

To gain an idea of the execution speed we created a test script that invokes the dashboard 1000 times with a random delay between 1 and 3 s on a standard server machine. To investigate the potential impact of multithreading on execution speed we implemented an alternative controller component, using a multithreading library to parallelize the execution of the signal MLMs.

3. Results

We successfully implemented a working prototype of an MLM package which creates a dashboard view, meeting the requirements of the clinical users. The interviews showed that the local clinicians had a clear conception of the desired surveillance dashboard. They demanded a rectangular panel with various kinds of output elements. Specifically requested were elements for displaying numeric values, in particular laboratory results and score values, with the use of meaningful colors in case particular values exceed critical limits. Requests with a lower priority included trend arrows, miniature graphs (sparklines), and existence indicators, such as a symbol in case documentation for some critical microbiology result was available. Fig. 3 displays a detail from a generated dashboard view. The leftmost and rightmost columns, which display patient names and bed positions, are not shown.

The dashboard was integrated into the graphical user interface of the PDMS in the form of a tab with an embedded browser element. A dashboard view is created each time this tab is activated by a clinical user. The third row of the dashboard detail in Fig. 3 shows an empty signal (THROMBO). For this patient, no current thrombocyte value was present inside the patient record. The architecture of the resulting prototype is shown in Fig. 4. The controller MLM is automatically called by the activation of the corresponding tab and does not require any arguments. It successively calls the signal MLMs to fill the data structure with signal objects, as displayed in Fig. 5. Each signal MLM in turn receives a case number as its argument, and sets the patient context of any database query accordingly, as illustrated in Fig. 2 within the DATA slot. The parameter "$case" inside the curly braces expression communicates to the Arden engine that the patient context of this database query is to be set according to the content of the variable "case". After accumulating the returned signal objects in the internal data structure of

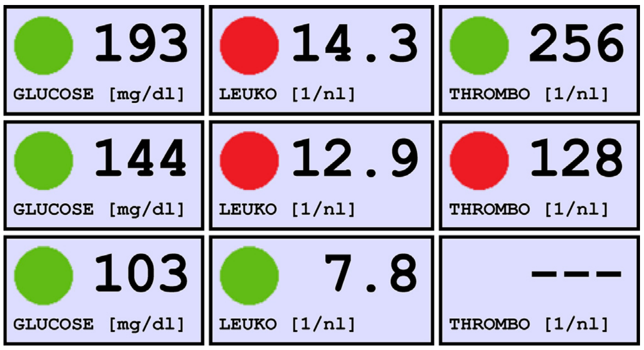


Fig. 3. Detail from a generated dashboard view. Red and green colors indicate whether a critical threshold was violated or not. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

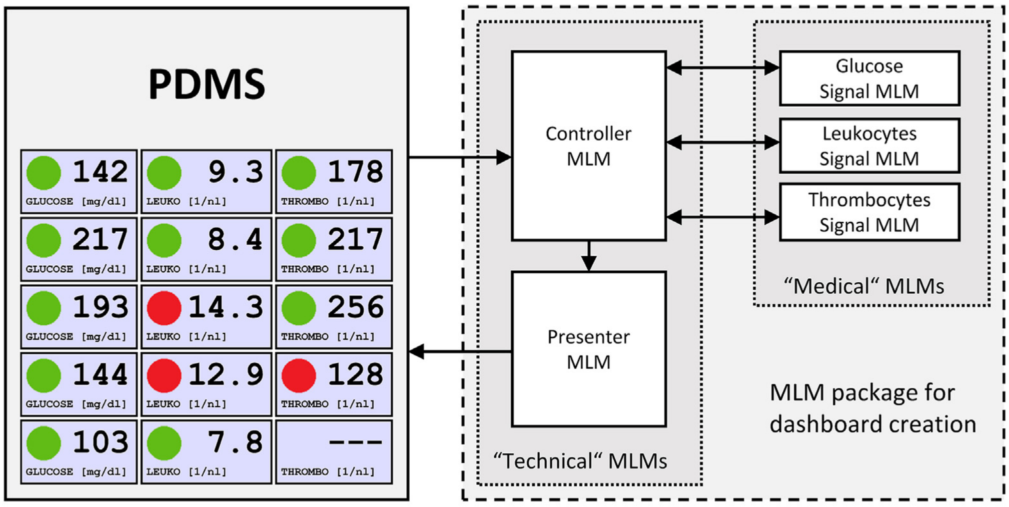


Fig. 4. Architecture of the MLM package for dashboard creation.

the dashboard, the controller MLM passes it to the presenter MLM for display inside the PDMS.

The benchmarks revealed an average execution time for creating a dashboard view of 190 ms, with a minimum of 172 ms and a maximum of 452 ms. Thus, the execution time of a single signal MLM was about 3.2 ms. Preliminary experiments with multithreaded execution of the signal MLMs indicated a slowdown of 35.7% compared to the original consecutive computation, which suggests that parallelization is disadvantageous in case of MLMs with such a low level of complexity.

The technical details of our prototype revealed that an Arden Syntax installation must fulfil three conditions to enable our approach. First, it must support the object data type, provided by Arden Syntax version 2.5 or higher. From a purely technical point of view it would be possible to implement a dashboard based on plain lists, although the code would rapidly become quite intricate and difficult to maintain. Second, it must provide a means to set the patient context in database queries at runtime. Third, it must provide a way to display the HTML and CSS code generated by the presenter MLM to the clinical user.

Extending a given dashboard configuration with additional signals requires adjusting the code inside the controller MLM. More precisely, the set of desired signal MLMs has to be declared inside the DATA slot and called inside the LOGIC slot, just as Fig. 5 demonstrates for the previous MLMs. This implies that in case another dashboard configuration should build on different signal logic, the controller MLM is not entirely reusable, because Arden Syntax requires the names of all medical MLMs to be declared as

```
DATA:
PRESENTER            := MLM 'DASHBOARD_PRESENTER';
SIGNAL_GLUCOSE       := MLM 'DASHBOARD_GLUCOSE';
SIGNAL_LEUKO         := MLM 'DASHBOARD_LEUKO';
SIGNAL_THROMBO       := MLM 'DASHBOARD_THROMBO';
DASHBOARDROW         := OBJECT [CASE,PATIENT,BED,SIGNALLIST];
DASHBOARD            := READ AS DASHBOARDROW {CURRENT_PATIENTS};
DASHBOARD.SIGNALLIST := ();
;;
...
LOGIC:
FOR ROW IN DASHBOARD DO
    DISPLAYELEMENT := CALL SIGNAL_GLUCOSE WITH ROW.CASE;
    ROW.SIGNALLIST := ADD DISPLAYELEMENT TO ROW.SIGNALLIST;

    DISPLAYELEMENT := CALL SIGNAL_LEUKO WITH ROW.CASE;
    ROW.SIGNALLIST := ADD DISPLAYELEMENT TO ROW.SIGNALLIST;

    DISPLAYELEMENT := CALL SIGNAL_THROMBO WITH ROW.CASE;
    ROW.SIGNALLIST := ADD DISPLAYELEMENT TO ROW.SIGNALLIST;
ENDDO;
CONCLUDE TRUE;
;;
ACTION:
CALL PRESENTER WITH DASHBOARD;
```

Fig. 5. DATA, LOGIC and ACTION slots of the controller MLM.

constants. Therefore, any additional dashboard requires its own controller MLM. However, existing signal MLMs may be reused by any other dashboard without adaption. The presenter MLM is also entirely reusable. Sharing an MLM package for dashboard creation with another institution therefore should theoretically only require the usual adjustments of the curly braces expressions, as long as the receiving institution fulfils the general technical requirements described above. However, we have not yet gathered experiences with transferring our prototype to another institution.

4. Discussion

Our study demonstrates the general feasibility of creating dashboards in Arden Syntax, and we believe that our code samples document that this can be surprisingly easy, potentially even simpler than with all-purpose programming languages. In fact, Arden Syntax increasingly appears as a powerful tool to extend clinical information systems with versatile additional functions. Arden Syntax may therefore be seen in a broader sense as a generic plugin language, with capabilities even beyond its originally intended field of application, the representation of medical knowledge. Equipped with access to patient data and a means to display outputs, modern versions of Arden Syntax might even be used as an alternative for programming languages like PHP or Python in the context of clinical information systems. From our experience, notably the well-designed WHERE operator, the temporal, the aggregation and the transformation operators of Arden Syntax significantly facilitate typical steps of patient data processing in a convenient way.

Today, Arden Syntax looks back on a history of a quarter century. During the first 15 years, the typical appearance of Arden Syntax was an alert box generated by a clinical event monitor, besides some attempts to use it for other fields of application “outside the alert box” [28,29]. In Arden Syntax versions before 2.5, multi-patient dashboards would be relatively difficult to implement, because the type system of these versions is literally one-dimensional, lacking appropriate data structures. Recent versions of Arden Syntax that also support objects certainly widen the feasible fields of application, although the literature appears to almost ignore this development. Another important barrier to the construction of dashboards is the fact that most Arden Syntax installations are limited to a single patient focus, such as the model implementation at the Columbia Presbyterian Medical Center [30]. In such installations, whenever an MLM interacts with the clinical information system, the patient context is implicitly set and cannot be altered inside the MLM. For patient-specific alerting functions, this is unproblematic and even simplifies the work of the knowledge engineer insofar as he does not have to care about this aspect. In case of multi-patient CDS, however, an implicitly assumed patient context becomes an obstacle, because iterating over patient sets inside an MLM becomes impossible. Throughout the history of Arden Syntax a single patient focus was so widespread that it has even led to the overall impression that MLMs are basically unsuitable for population-based CDS (e.g. “[. . .] because Arden Syntax is patient-specific, it cannot be used for population-based decision support (such as a quality-of-care dashboard) [. . .]” [31]). Jenders and Shah already discussed this restriction when they described the construction of a multi-patient monitor for nosocomial infections, which oversees the potential spreading of pathogenic organisms [32]. Since their Arden Syntax installation was limited to a single patient focus, it was only possible to preprocess microbiology findings in MLMs separately, while the cross-patient part of the decision logic had to be performed independently by a statistical monitor. Using an Arden Syntax installation that is not bound by the described limitation would have made it possible to implement such a monitor entirely in Arden Syntax. The specification does not prohibit the explicit setting of the patient context inside the curly braces expressions, whose contents are generally left to the particular institution. A comparison of standalone Arden engines, including the commercial system by Medexter [33] and the open source system Arden2Bytecode [34], revealed several related approaches to alter the patient context of database queries at MLM runtime, based on access to the symbol table of the particular MLM. Perhaps it could be reasonable to introduce a special statement to the Arden Syntax standard that sets the patient context, such as

```
SET PATIENT CONTEXT TO <identifier>;
```

We presume that such a statement would be easier to understand than a reference to a declared variable inside a curly braces expression. While we decided to implement the entire dashboard in Arden Syntax, alternatively it might have been feasible to encode the functionality of the controller MLM and the presenter MLM in an all-purpose language such as Java or Python. In this case, knowledge transfer would require the receiving institution to provide technical components for consistently executing the signal MLMs. If the dashboard is implemented entirely in Arden Syntax, as in our example, the standardization of this language should facilitate the transfer as a whole.

Our prototype was based on crisp logic, so the colors inside the signal elements instantly switch from green to red whenever a critical limit is reached. In the context of clinical dashboards it might be interesting to experiment with Fuzzy Arden Syntax [35,36], which was introduced in version 2.9 of the standard. Fuzzy Arden Syntax provides an easy way to implicitly calculate a so-called *degree of applicability* that may, for example, be transformed to a color from a continuous spectrum which indicates the position of a value within a transition area. Moreover, Fuzzy Arden Syntax is capable of applying fuzzy logic to timestamps as well, allowing for an easy way to indicate the timeliness of a value. This will be a field of further investigation.

The construction of dashboards closely relates to presentation mechanism for MLM outputs, an aspect that is usually of low importance in Arden Syntax. MLMs designed for alerting typically create short text messages and send them to communication endpoints such as alert boxes, pagers, or mobile phones. Such messages usually require little formatting. In case of a dashboard, in contrast, formatting becomes a relatively important step to display results in a convenient and ergonomic way. This topic is rarely addressed in the context of Arden Syntax, as an exception see [37], where HTML pages are constructed from MLM outputs. At our institution, MLMs are mostly user-driven, generating output beyond plain text, like tables or line plots. In order to produce such diverse output elements, we have so far used a relatively simple template engine that is entirely written in Arden Syntax, previously described in [27], whose MLMs convert description objects into string representations that can be displayed. The presenter MLM for creating the dashboard view works similarly, being called with the dashboard data structure generated by the controller MLM as an argument. We presume that presentation aspects will become increasingly important in the near future, as evolving communication endpoints will support progressively more versatile outputs, and user-driven MLMs will likely become more widespread. Our PDMS, for example, was recently complemented with full-featured browser elements that may be embedded anywhere inside the user interface for medical and nursing documentation, enabling the use of multiple dashboard instances.

Providing user-driven CDS by generating dashboards may also require consideration of execution time. Traditional data-driven alerting MLMs are executed in the background, invisible to the user. In case of an alert message, a delay of a few seconds might hardly be perceptible for clinicians. However, In case of user-driven creation of a dashboard, where any delay might translate into waiting times for users, speed is a decisive factor to ensure user acceptance [38]. We observed that the signal MLMs processed by the prototype execute within a few milliseconds, because they contain a very simple decision logic. Nevertheless, other types of signals, such as score or trend calculations, may require more complex processing, so that aggregate execution times could plausibly entail noticeable delays when calculated for sets of patients. This aspect must therefore be taken into account, or carefully tested in practice, when future dashboards are assembled from individual CDS modules that are computationally more demanding.

Arden Syntax can be aptly described as a hybrid between production systems and procedural programming [39]. During the creation of the dashboard MLM package, this hybrid character became very obvious. Both the controller and the presenter MLM do not contain any medical knowledge. They are "technical" MLMs insofar as they perform routine tasks that do not depend on specific clinical knowledge. In contrast, the signal MLMs are "medical" MLMs as they comprise clinical decisions whether a laboratory result is deemed critical or not. In case of MLM packages, we recommend such a separation of technical and medical aspects, not only for the sake of simplifying maintenance of the knowledge base, but also to support reuse of Arden Syntax code through modularization. Splitting larger MLMs into packages was already recommended by Adlassnig and Rappelsberger [40]. It would have been theoretically feasible to implement our prototypical dashboard in the form of one single MLM, but that would have complicated a potential reuse of code segments for signal creation or HTML generation. Redeploying parts of the implementation would then involve substantial redundancy, which was an early point of criticism concerning Arden Syntax [41]. At our institution, we make extensive use of modularization to avoid redundancy and to reuse encoded knowledge. The presenter MLM and the individual signal MLMs are entirely reusable for additional dashboards. The controller MLM is not reusable, because the names of the signal MLMs cannot be passed as arguments to the controller, but have to be contained inside the controller MLM as term constants (see specification [2], Section 11.2.4). As a consequence, each dashboard requires its own controller MLM. Since such an MLM is easily composed from a few lines of code, as illustrated in Fig. 5, this is not a matter of concern. Reusing the controller MLM would require a modification of the Arden Syntax implementation. To be specific, the CALL statement would have to be adjusted to call MLMs using MLM names contained in variables. In this case, the names of the signal MLMs may be passed as arguments to the controller MLM, which would then process this input within nested loops.

An alternative implementation of the controller MLM could use an event call in accordance with Section 10.2.5.6 of the specification. Such an event call does not explicitly launch another MLM by its name, but any MLM that subscribes to a specific event as defined in its EVOKE slot; the outputs of all called MLMs are concatenated to a list which in our use case represents one row of the dashboard. This alternative approach has two advantages. First, the controller MLM would be smaller, insofar as multiple call statements inside the loop in Fig. 5 are replaced with one concise event call statement. Second, the controller MLM would be reusable for additional dashboards in Arden Syntax environments that allow for access to variables within curly braces expressions, because this approach does not require the names of the signal MLMs to be encoded inside the controller MLM. However, this alternative method has also some disadvantages on the maintenance level. First, according to the specification, the order of the signal elements depends on the local Arden Syntax environment, although in most implementations this will be determined by the PRIORITY slots of the particular signal MLMs. Second, in case a signal MLM is used by multiple dashboards, adjusting the PRIORITY slot of a signal MLM might unintentionally affect other dashboards because the order of the display elements cannot be configured independently for different dashboards. Third, configuration of a dashboard would be dispersed over multiple MLMs. Although our chosen implementation may require additional code as well as view-specific controller MLMs, we therefore still see it as more explicit and thus maintainable.

In case of sharing dashboards implemented in Arden Syntax between institutions, we expect no additional obstacles beyond the adjustment of the curly braces expressions, as long as the receiving institution fulfils the basic requirements for multi-patient MLMs and provides an appropriate Arden Syntax version supporting the object data type. Our dashboard prototype primarily serves as a proof of concept. However, its basic architecture appears suitable for the integration of dashboards into our PDMS in order to use them in clinical routine and to evaluate their impact on clinical performance. We are currently in the process of implementing a dashboard which displays the most important clinical score values in a surgical ICU. Simultaneously, we are working on a variety of different signal elements and plan to evaluate the usability of different dashboard designs. Moreover, we intend to evaluate the impact of a nutrition dashboard, based on the described approach, on the quality of care at our largest ICU.

References

[1] Hripcsak G, Ludemann P, Pryor TA, Wigertz OB, Clayton PD. Rationale for the Arden Syntax. Comput Biomed Res 1994;27(4):291–324.
[2] Arden Syntax for Medical Logic Systems, Version 2.8, Health Level Seven, 2012.
[3] Hripcsak G, Clayton PD, Jenders RA, Cimino JJ, Johnson SB. Design of a clinical event monitor. Comput. Biomed Res 1996;29(3):194–221.
[4] Hripcsak G. Writing Arden Syntax Medical Logic Modules. Comput Biol Med 1994;24(5):331–63.
[5] Jenders RA, Corman R, Dasgupta B. Making the standard more standard: a data and query model for knowledge representation in the Arden syntax. AMIA Annu Symp Proc 2003:323–30.
[6] Powsner SM, Tufte ER. Graphical summary of patient status. Lancet 1994;344(8919):386–9.
[7] Bakos KK, Zimmermann D, Moriconi D. Implementing the clinical dashboard at VCUHS. Proc Int Congr Nurs Inf 2012;2012:11.
[8] Cheng Calvin KY, Ip Dennis KM, Cowling BJ, Ho LM, Leung GM, Lau Eric HY. Digital dashboard design using multiple data streams for disease surveillance with influenza surveillance as an example. J Med Internet Res 2011;13(4):e85.
[9] Krupinski EA. Optimizing the pathology workstation cockpit: challenges and solutions. J Pathol Inf 2010;1:19.
[10] McCoy AB, Thomas EJ, Krousel-Wood M, Sittig DF. Clinical decision support alert appropriateness: a review and proposal for improvement. Ochsner J 2014;14(2):195–202.
[11] McMenamin J, Nicholson R, Leech K. Patient dashboard: the use of a colour-coded computerised clinical reminder in Whanganui regional general practices. J Prim Health Care 2011;3(4):307–10.
[12] Minnigh TR, Gallet J. Maintaining quality control using a radiological digital X-ray dashboard. J Digit Imaging 2009;22(1):84–8.
[13] Morgan MB, Branstetter BF, Lionetti DM, Richardson JS, Chang PJ. The radiology digital dashboard: effects on report turnaround time. J Digit Imaging 2008;21(1):50–8.
[14] Nagy PG, Warnock MJ, Daly M, Toland C, Meenan CD, Mezrich RS. Informatics in radiology: automated web-based graphical dashboard for radiology operational business intelligence. Radiographics 2009;29(7):1897–906.
[15] Park KW, Smaltz D, McFadden D, Souba W. The operating room dashboard. J Surg Res 2010;164(2):294–300.
[16] Waitman LR, Phillips IE, McCoy AB, Danciu I, Halpenny RM, Nelsen CL, et al. Adopting real-time surveillance dashboards as a component of an enterprisewide medication safety strategy. Jt Comm J Qual Patient Saf 2011;37(7):326–32.
[17] Ratwani RM, Fong A. 'Connecting the dots': leveraging visual analytics to make sense of patient safety event reports. J Am Med Inf Assoc 2015;22(2):312–7.
[18] Simpao AF, Ahumada LM, Desai BR, Bonafide CP, Gálvez JA, Rehman MA, et al. Optimization of drug-drug interaction alert rules in a pediatric hospital's electronic health record system using a visual analytics dashboard. J Am Med Inf Assoc 2015;22(2):361–9.
[19] Starmer J, Giuse D. A real-time ventilator management dashboard: toward hardwiring compliance with evidence-based guidelines. AMIA Annu Symp Proc 2008:702–6.
[20] Zaydfudim V, Dossett LA, Starmer JM, Arbogast PG, Feurer ID, Ray WA, et al. Implementation of a real-time compliance dashboard to help reduce SICU ventilator-associated pneumonia with the ventilator bundle. Arch Surg 2009;144(7):656–62.

[21] West VL, Borland D, Hammond WE. Innovative information visualization of electronic health record data: a systematic review. J Am Med Inf Assoc 2015;22(2):330–9.
[22] Dowding D, Randell R, Gardner P, Fitzpatrick G, Dykes P, Favela J, et al. Dashboards for improving patient care: review of the literature. Int J Med Inf 2015;84(2):87–100.
[23] van Loon K, Guiza F, Meyfroidt G, Aerts J, Ramon J, Blockeel H, et al. Dynamic data analysis and data mining for prediction of clinical stability. Stud Health Technol Inf 2009;150:590–4.
[24] Morik K, Imhoff M, Brockhausen P, Joachims T, Gather U, Imboff M. Knowledge discovery and knowledge validation in intensive care. Artif Intell Med 2000;19(3):225–49.
[25] Meyfroidt G, Güiza F, Ramon J, Bruynooghe M. Machine learning techniques to examine large patient databases. Best Pract Res Clin Anaesthesiol 2009;23(1):127–43.
[26] Bürkle T, Castellanos I, Tech H, Prokosch H. Implementation of a patient data management system—an evaluation study of workflow alterations. Stud Health Technol Inf 2010;160(Pt 2):1256–60.
[27] Kraus S, Castellanos I, Toddenroth D, Prokosch H, Bürkle 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(5):465–73.
[28] Sailors RM, Bradshaw RL, East TD. Moving Arden Syntax outside of the (Alert) box: a paradigm for supporting multi-step clinical protocols. Proc AMIA Symp 1998:1071.
[29] Sherman EH, Hripcsak G, Starren J, Jenders RA, Clayton P. Using intermediate states to improve the ability of the Arden Syntax to implement care plans and reuse knowledge. Proc Annu Symp Comput Appl Med Care 1995:238–42.
[30] Hripcsak G, Cimino JJ, Johnson SB, Clayton PD. The Columbia-Presbyterian Medical Center decision-support system as a model for implementing the Arden Syntax. Proc Annu Symp Comput Appl Med Care 1991:248–52.
[31] Wright A, Sittig DF. A four-phase model of the evolution of clinical decision support architectures. Int J Med Inf 2008;77(10):641–9.
[32] Jenders RA, Shah A. Challenges in using the Arden Syntax for computer-based nosocomial infection surveillance. Proc AMIA Symp 2001: 289–93.
[33] Fehre K, Adlassnig KP. Service-oriented Arden-syntax-based clinical decision support. In: Schreier G, Hayn D, Ammenwerth E, editors. Proceedings of the eHealth 2011. Vienna, Austria: OCG; 2011. p. 123–8.
[34] Gietzelt M, Goltz U, Grunwald D, Lochau M, Marschollek M, Song B, et al. ARDEN2BYTECODE: a one-pass Arden Syntax compiler for service-oriented decision support systems based on the OSGi platform. Comput Methods Programs Biomed 2012;106(2):114–25.
[35] Vetterlein T, Mandl H, Adlassnig KP. Fuzzy Arden Syntax: a fuzzy programming language for medicine. Artif Intell Med 2010;49(1):1–10.
[36] Vetterlein T, Mandl H, Adlassnig KP. Processing gradual information with Fuzzy Arden syntax. Stud Health Technol Inf 2010;160(Pt 2):831–5.
[37] Karlsson D, Ekdahl C, Wigertz O, Shahsavar N, Gill H, Forsum U. Extended telemedical consultation using Arden Syntax based decision support, hypertext and WWW technique. Methods Inf Med 1997;36(2):108–14.
[38] Bates DW, Kuperman GJ, Wang S, Gandhi T, Kittler A, Volk L, et al. Ten commandments for effective clinical decision support: making the practice of evidence-based medicine a reality. J Am Med Inf Assoc 2003;10(6): 523–30.
[39] Samwald M, Fehre K, de Bruin J, Adlassnig KP. The Arden Syntax standard for clinical decision support: experiences and directions. J Biomed Inf 2012;45(4):711–8.
[40] Adlassnig KP, Rappelsberger A. Medical knowledge packages and their integration into health-care information systems and the World Wide Web. Stud Health Technol Inf 2008;136:121–6.
[41] Shwe M, Sujansky W, Middleton B. Reuse of knowledge represented in the Arden syntax. Proc Annu Symp Comput Appl Med Care 1992:47–51.