Artificial Intelligence in Medicine 92 (2018) 10–14

Contents lists available at ScienceDirect

Artificial Intelligence in Medicine

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

Evolution of the Arden Syntax: Key Technical Issues from the Standards Development Organization Perspective

Robert A. Jenders[a,b,*], Klaus-Peter Adlassnig[c,d], Karsten Fehre[d], Peter Haug[e,f]

[a] Center for Biomedical Informatics, Charles Drew University, Los Angeles, CA 90059, USA
[b] Department of Medicine, Clinical and Translational Science Institute, University of California, Los Angeles, CA 90095, USA
[c] Section for Artificial Intelligence and Decision Support, Center for Medical Statistics, Informatics, and Intelligent Systems, Medical University of Vienna, Spitalgasse 23, 1090 Vienna, Austria
[d] Medexter Healthcare GmbH, Borschkegasse 7/5, 1090 Vienna, Austria
[e] Homer Warner Research Center, Intermountain Healthcare, 5121 South Cottonwood Street, Murray, UT 84107, USA
[f] Department of Biomedical Informatics, University of Utah, 421 Wakara Way, Salt Lake City, UT 84108, USA

ARTICLE INFO

Article history:
Received 17 June 2016
Accepted 9 August 2016

Keywords:
HL7 International
Arden Syntax
Clinical decision support

ABSTRACT

Background: The initial version of the Arden Syntax for Medical Logic Systems was created to facilitate explicit representation of medical logic in a form that could be easily composed and interpreted by clinical experts in order to facilitate clinical decision support (CDS). Because of demand from knowledge engineers and programmers to improve functionality related to complex use cases, the Arden Syntax evolved to include features typical of general programming languages but that were specialized to meet the needs of the clinical decision support environment, including integration into a clinical information system architecture.
Method: Review of the design history and evolution of the Arden Syntax by workers who participated in this evolution from the perspective of the standards development organization (SDO).
Results: In order to meet user needs, a variety of features were successively incorporated in Arden Syntax. These can be grouped in several classes of change, including control flow, data structures, operators and external links. These changes included expansion of operators to manipulate lists and strings; a formalism for structured output; iteration constructs; user-defined objects and operators to manipulate them; features to support international use and output in different natural languages; additional control features; fuzzy logic formalisms; and mapping of the entire syntax to XML. The history and rationale of this evolution are summarized.
Conclusion: In response to user demand and to reflect its growing role in clinical decision support, the Arden Syntax has evolved to include a number of powerful features. These depart somewhat from the original vision of the syntax as simple and easily understandable but from the SDO perspective increase the utility of this standard for implementation of CDS. Backwards compatibility has been maintained, allowing continued support of the earlier, simpler decision support models.

1. Introduction

1.1. Background

Key design goals of the originators of The Arden Syntax for Medical Logic Systems included the explicit representation of the data and logic used in clinical reasoning in a standard executable format so that it could be used to implement a knowledge base for a computer-based clinical decision support system (CDSS) and so that such knowledge bases could be shared with minimal changes [1]. The explicit representation of knowledge in a relatively simple, English-like syntax was intended to facilitate composition and validation by clinical domain experts without significant expertise in computer science. The standard representation of knowledge was intended to facilitate knowledge sharing: Ideally few changes would be required to the logic implemented as Medical Logic Modules (MLMs) in CDSSs that were compliant with the Arden Syntax standard. These design goals undergirded the original version of the Arden Syntax that was created by an ad hoc assembly of experts

* Corresponding author at: Center for Biomedical Informatics, Charles Drew University, Los Angeles, CA, 90059, USA.
E-mail address: jenders@ucla.edu (R.A. Jenders).

https://doi.org/10.1016/j.artmed.2016.08.001
0933-3657/

from academia, government and industry in 1989, drawing on constructs included in two pioneering CDSSs of the time—the Health Evaluation through Logical Processing (HELP) system at the University of Utah and the CARE language at the Regenstrief Institute [1]. This initial version was later formalized as the official version 1 of the Arden Syntax as by the standards development organization (SDO) ASTM as standard E1460-92 in 1992 [2]. The name of the standard was drawn from the name of the Columbia University retreat center where the initial assembly to discuss such a knowledge representation formalism was held in New York. Over time, a number of commercial vendors of health information system software incorporated the Arden Syntax as part of the clinical decision support (CDS) functionality in their products or offer it as vendor-independent solution, including Agfa, Allscripts, McKesson, Medexter and Siemens. With the widespread implementation of these systems, the Arden Syntax ended up being used in the operations of numerous health care organizations.

While this initial version, in its straightforward, easy-to-read syntax and standard format supported composition, validation and sharing by clinical domain experts, over time it became clear that such individuals were not the principal authors of MLMs. Instead, health care organizations using CDSSs compliant with the Arden Syntax employed knowledge engineers and programmers to reduce the expertise and logic supplied by clinical domain experts to Arden Syntax MLMs. As these workers in turn were accustomed to the powerful operators and flow of control of traditional programming languages, they began to petition the SDOs (ASTM and later Health Level Seven International) to incorporate such features in the Arden Syntax. This in turn led to the ongoing tension between readability and composability on the one hand and programming power and efficiency on the other.

The result of this tension has been an evolution of the Arden Syntax from a relatively simple programming language offering only linear flow of instructions, basic subroutine calls and simple data structures into a complex but also powerful executable knowledge representation formalism that offers intricate flow of control, complex data structures and highly useful operators that, while they better allow direct representation of complex clinical reasoning and data, also potentially conflict with composition and validation by clinical domain experts [3]. As workers who have participated in the development of the Arden Syntax over a significant part of its quarter-century history, who have worked with others in industry and government who have made use of it and who have actively implemented it, we describe here the evolution of the Arden Syntax from the SDO perspective to meet the varied needs of knowledge engineers, programmers, health informaticians and health care organizations to implement and share clinical knowledge in CDSSs.

1.2. The original Arden Syntax

In its original formulation, ultimately standardized by ASTM in 1992, the Arden Syntax was, compared to most computer programming languages of the time, relatively simple. Computable knowledge was represented in MLMs designed as structured, hierarchical frames, each having three subparts known as categories, each of which in turn consisted of several slots, each of which contained documentary information about the MLM and its subject or executable statements. The three categories organized the knowledge necessary to document and execute a CDS intervention [1]. The Maintenance category consists of slots that address the software engineering aspects of the MLM, including author, health care origin, date, version and the like. The Library category includes slots for a narrative description of the logic for and the rationale for the MLM as well as free-text references to the scientific literature and other supporting material. The Knowledge category contains the executable statements needed to provide decision support. These include a data slot, which can contain database query statements and other operations that involve the retrieval of clinical data; a logic slot, which contains the production rules that encode logic that, in turn, processes the data to reach a conclusion; and an action slot that specifies the decision support system output, typically a message to a human decision-maker, in the event that the logic concludes true. An urgency slot allowed an MLM writer to indicate the clinical importance of the knowledge or intervention, while a priority slot could be encoded to allow for conflict resolutions in the event that multiple MLMs were triggered to execute at the same time. Finally, an evoke slot allowed definition of conditions that would trigger the execution of the MLM by the CDSS.

The heart of the Arden Syntax was the shareable, executable statements that comprised the data and logic slots. Variables were loosely-typed, which provided flexibility for developers, although this reduced the capacity of compilers or interpreters to detect programming errors. Data types were the conventional, basic ones found in programming languages of the time: number (including integer and real), string and Boolean. The only complex or structured data type was the list, which consisted of zero or more elements of a basic data type. Operators that could accept these data as parameters and process them included the conventional arithmetic and geometric ones, such as addition, subtraction, comparisons (equality, less than, greater than, etc.), aggregation (AVERAGE, MEDIAN, MINIMUM, MAXIMUM, etc.) and assignment. However, because of the importance of temporal reasoning and trends over time in the health science domain, operators that supported such reasoning were included, including comparison operators such as BEFORE, AFTER and AGO as well as duration operators. The Syntax also provided the means of creating subroutines, as MLMs could invoke other MLMs.

Manifesting the simplicity and consequent understandability of the Syntax, the flow of control in an MLM was linear. The CDSS processed the statements in the order in which they were written, with none of the syntactic features, such as WHILE or FOR loops, that supported iteration in conventional programming languages.

Finally, given the importance of clinical data in executing CDS, mapping statements were provided in order to retrieve data from external stores into variables in the MLM. However, because there was no standard data model widely prevalent at the time, the syntax for such mappings, particularly those for data queries, was not specified. Instead, site-specific mappings could be encoded inside curly braces ({}) for easy identification by the compiler or interpreter. While this flexibility eased the process of MLM implementation, it also represented a challenge for knowledge sharing, as these mappings had to be updated when an MLM was implemented at an organization with a different information system architecture. This challenge, which is not unique to the Arden Syntax but pertains to any formalism meant to be used to share knowledge and includes lack of a standard data model, data query syntax and uniformity in use of standard terminologies, consequently came to be known as the “curly braces problem” [4].

Despite offering only a relatively simple and straightforward structure and with only a basic set of operators and data types, albeit augmented with powerful operators to support temporal reasoning, this initial version of the Arden Syntax nonetheless could be used to implement significant CDS interventions, including complex clinical practice guidelines [5,6]. In addition, the simplicity of the Syntax facilitated the creation and validation of MLMs by domain experts without deep knowledge of information systems or computer programming.

Nevertheless, as a result of the aforementioned demand by programmers and knowledge engineers for increasingly powerful computing constructs, the Arden Syntax, beginning mainly with its change in standards development organizations from ASTM to HL7

in 1998, underwent a steady evolution to incorporate additional and more powerful features that could aid its utility in facilitating clinical decision support.

2. Methods

The authors, who collectively trace their history of working with and supervising the Arden Syntax standard back to its original ASTM version, reviewed the evolution of the Arden Syntax from its initial version (v1) sponsored by ASTM in 1992 and subsequent versions (v2.0, v2.1, v2.5, v2.6, v2.7, v2.8, v2.9 and v2.10) sponsored by HL7 starting in 1998 and concluding with the latest version published to date in 2014 [7]. The authors identified major themes in the ongoing evolution of the Syntax and documented the rationale for the changes that were undertaken.

3. Results

The Arden Syntax evolved over a period of twenty-four years and continues to evolve in response to requirements brought to standards development organizations by workers interested in improving its utility for customers, clinicians and health care organizations as well as facilitating its shareability. These developments can be categorized in a number of themes: Control flow, operators, data structures, external links and overall structure. Instead of a strictly linear, temporal progression of enhancements to the standard, we present these changes organized by themes and their rationales. Because backward compatibility has been preserved during this evolution, the latest version of the standard contains all the elements of the preceding versions. Examples of how these changes may be used in composing MLMs for knowledge-based CDS interventions are included as an appendix in the standard itself [7] as well as in a separate implementation guide [8].

3.1. Control flow

Flow of control refers to the ordering of executable statements by the processor or CDSS. While in its simplicity the original Arden Syntax did not provide iteration statements that could allow for multiple execution of one or more statements depending on conditions, developers recognized that constructs to support iteration would improve the implementation of certain kinds of clinical reasoning, e.g., evaluating a series of test results, even as it increased the risk of incorrect programming such as the creation of infinite loops. Balancing these risk, then, in the transformation of the Arden Syntax from its original version to v2.0, FOR and WHILE loops were added to support iteration. While iteration of a kind could be implemented previously using the WHERE clause applied to lists, the implementation of explicit iteration clarified this reasoning. Further authority by the knowledge engineer to manipulate flow control was added in the transformation from v2.6 to v2.7 of the standard with the inclusion of a SWITCH-CASE statement.

3.2. Operators

The original version of the Arden Syntax already possessed a rich set of operators for manipulating and evaluating data. These included operators for processing lists (the most complex data structure in the original Syntax), comparison operators, string operators, temporal operators, duration operators, aggregation operators and arithmetic operators. Nevertheless, representation of nuanced clinical reasoning demanded increasingly powerful operators. Additional list operators were added in order to manipulate data in lists, including SORT, REVERSE, EARLIEST, LATEST and operators to identify the index of a list element as well as to insert or remove elements from a list arbitrarily. Considering the importance of text data in clinical reasoning, developers added additional operators for manipulation of strings, including truncation, substring searches, determination of the length of a string, extraction of characters from a string and string formatting and conversion of numeric strings to numbers.

While the original version of the Syntax already offered powerful operators to support temporal reasoning, increasing requirements to be able to implement still more nuanced temporal reasoning led to the creation of additional related operators. These included the ability to manipulate time values by substituting parts of a time stamp with different values; extraction of different parts of a complete time value, such as time of day and day of week; and conversion of a string to a time value.

An additional significant advance in the standard with regard to inclusion of powerful operators occurred in the transformation of v2.8 to v2.9 with the addition of operators to implement fuzzy sets and fuzzy logic directly [9]. As contrasted with crisp sets, in which set membership could be described by simple “yes” or “no” (a particular data element either is or is not a member of a set possessing certain identifying characteristics) fuzzy sets manifest degrees of membership. These degrees of membership are elements of the closed interval [0,1] expressing not only “no = 0” and “yes = 1” membership of a particular data element in a set, but also intermediate membership degrees, identified in the Syntax as the “truth value.” Consequently, in addition to a value and a primary time associated with a data element in the original version of the Syntax, a data element now could have a truth value. Fuzzy logic propagates these degrees of membership through the inference network. Fuzzy logic also may involve the manipulation of linguistic variable such as “some” or “few.” As it happens, clinical reasoning is replete with fuzzy sets and fuzzy logic, such as in quality measures and clinical practice guidelines [10]. Consequently, the inclusion of fuzzy logic operators represented another advance in facilitating the direct representation of clinical reasoning in this knowledge representation standard.

3.3. Data structures and models

Reflecting their aim of understandability, the developers of the initial version of the Arden Syntax included only basic data types, such as number, time, duration, string and Boolean. The one complex data type was the list, which was comprised of zero (an empty list) or more elements of another data type. A data element of one of these types could be stored in a variable that in turn was loosely-typed. The only other structure beyond the list was the implicit primary time that was associated with any data element retrieved using a READ statement. This was defined as the time most clinically relevant to a particular value, such as the venipuncture time for obtaining a blood sample on which a clinical laboratory test was later performed.

Over time, however, knowledge engineers became increasingly facile with object-oriented programming and database paradigms and with recognition that clinical data naturally can be organized into structured primitives, e.g., blood pressure (with the dimensions of location of measurement, method of measurement, body position at the time of measurement, systolic pressure, diastolic pressure and the like). Such primitives group related attributes into a single structure. This led to the inclusion of structured objects in the Arden Syntax along with operators to manipulate objects and their parameters in the transformation of v2.1 of the standard to its immediate successor v2.5. (This jump in the numbering of versions of Arden Syntax was deliberately chosen to mark the significance of the changes made therein, particularly the introduction of objects as a data type and operators to manipulate them.)

The Arden object model allowed structured data to be manipulated as complex objects instead of a collection of disconnected variables. These operators included formalisms for defining and cloning an object as well as assigning and retrieving the value of its attributes using a dot notation. This functionality was bolstered further in the transformation from v2.6 to v2.7 of the standard by providing an enhanced assignment statement that accommodated objects and nesting. This view of data as a collection of structured primitives has been realized more generally in HL7 as a whole with the adoption of the Fast Healthcare Interoperability Resources (FHIR), in which structured primitives known as resources bring together related data elements in a coherent way [11].

3.4. External linkages and internationalization

Further improvements came about in the Arden Syntax in response to requirements presented to HL7 to enhance linkages between MLMs in a CDSS and other information systems, such as computer-based provider order entry (CPOE) systems and bibliographic knowledge bases. One such advance, introduced in the transformation of v2 to v2.1, was the structured WRITE statement. The Action slot provides the mechanism by which the CDSS performs some task if the logic of the MLM evaluates to true. The CDSS can call another MLM, return a value to a calling MLM or execute a WRITE statement.

The WRITE statement constitutes the primary mechanism by which an MLM communicates its conclusion and any knowledge-based intervention to human decision-makers. While this may entail writing an intermediate value back to the clinical database, using it as a blackboard to communicate among MLMs or systems, typically this involves communicating a text message to a human decision-maker. In the original version of the Syntax, this was an unstructured text string. However, with the rising demand to integrate CDSS with other systems such as CPOE systems and communication devices such as pagers, requirements were presented to HL7 to increase the utility of the WRITE statement by providing actionable parameters that could be used by these other devices and systems, such as communication routing parameters or actionable clinical orders. This led to the inclusion of an optional structured WRITE statement in the Arden Syntax expressing these parameters in XML.

Additional formalisms to further support for linkages to systems external to the CDSS inference engine came in the transition from v2.1 to v2.5 of the standard. These focused on the structuring of the Citations and Links slots in the Knowledge category. The purpose of the Citations slot was to allow knowledge engineers to list bibliographic references to the scientific literature that supported the knowledge contained in an MLM. The purpose of the Links slot was to provide a digital pathway to other resources such as electronic textbooks and educational modules. HL7 provided an optional structured form for the Citations slot that recommended use of either the Vancouver style or the OpenURL, optionally denoted as SUPPORT or REFUTE for evidence that affirms or questions, respectively, the knowledge encoded in the MLM. Providing a structured format facilitates direct linkage to bibliographic knowledge bases from the CDSS that, by providing explanatory material, augments the decision support being provided. HL7 provided an optional structure for the Links slot by pointing to the OpenURL standard as well. This allows knowledge engineers to encode links to specific Web pages or to pass parameters such as Medical Subject Headings (MeSH) that in turn can be used to display additional knowledge to the human decision-maker, further augmenting the power of CDS.

Additional enhancement of linkages to external systems that bolster one of the original design goals of the creators of the Arden Syntax are found in the Resources category that extended the standard in the transition from v2.5 to v2.6. This added a fourth major section to the MLM to complement the original Maintenance, Library and Knowledge categories. The new Resources category allows specification of the content of equivalent messages in different natural languages so that, upon specification of a default language for anticipated recipients, the CDSS can suggest knowledge-based interventions in this language. The parallel messages are structured into sections denoted by a mandatory language code optionally coupled with a regional or country specification, drawing on ISO 639-1 and ISO 3166-1 standards, respectively. This facilitates knowledge sharing by reducing the need to recode an MLM when it is shared across national borders and better integrates the CDSS with other systems in a particular country by aligning the language of communication.

3.5. Miscellaneous

In addition to these advances in the Arden Syntax, there are a few other changes that have been made in the standard that fall outside these organizing themes. One such change that directly supports the evolution of the Arden Syntax through multiple stages, introduced in the transformation of v2 to v2.1, is the Version slot in the Library category. This allows the knowledge engineer to declare the specific version of the Arden Syntax that was used to encode the executable clinical knowledge so that the appropriate compiler or interpreter can be used in the execution process. While the Arden Syntax is backward compatible so that a compiler or interpreter written for a particular version can accept as input an MLM encoded in that version or any prior one, this feature aids in error-checking and efficiency by ensuring that an interpreter or compiler that aligns with the same Arden Syntax version as the MLM is used.

Another advance that was the substance of the most recent version of the Arden Syntax—that is, the transformation of v2.9 to v2.10—is inclusion of an optional XML version in the form of an XML schema of Arden Syntax in the normative part of the standard. An Arden Syntax MLM conventionally is encoded as a frame with English-like statements organized into the four categories and using keywords to express concepts and executable instructions. By providing a declaration of the Syntax in XML; including XML tags to structure the MLM at different levels of detail; HL7 has aligned the Arden Syntax with the contemporary HL7 messaging standard and important innovations such as FHIR; which also is rendered in XML. In addition; fine-grained structuring of the Arden Syntax in XML even to the level of operators and operands allows users to take advantage of tools aligned with this standard to do things such as error-checking; cross-compilation into other executable formalisms [12] and even transformation of an MLM written in XML back to the conventional format of Arden.

4. Discussion

The progressive evolution of the Arden Syntax over a number of years has introduced features, including powerful operators, data structures and methods of reasoning, that better align the Syntax with the nuances of clinical reasoning and the way that clinical data are structured and stored and with the demands of its user base for such features. Nevertheless, reflecting the ongoing tension between meeting user requirements for a powerful programming language on the one hand while preserving a formalism that can be understood and validated by clinical domain experts, such evolution may have taken the Arden Syntax past the point where MLMs can be composed and inspected directly by clinicians.

Addressing this potential challenge has been the development of tools that allow composition of MLMs directly by clinicians without having to know the details of the underlying Syntax and health information systems [13,14]. Vendors that produce CDSSs that use

the Arden Syntax typically include knowledge editors and testing tools that simulate MLM execution that allow clinicians to create, modify and validate MLMs in ways that do not require deep knowledge of the latest features of the Syntax. In this way, the best of both worlds is achieved: A rich, powerful formalism for representation of executable clinical knowledge that serves the needs of knowledge engineers and computer programmers is being combined with the availability of mechanisms that allow use of the formalism without requiring deep knowledge of it.

Despite this extensive evolution in the Arden Syntax and tools to support it, including expansion of its data structures and linkages to external systems, a key challenge that affects any knowledge formalism in health care that will be used in knowledge transfer remains: The "curly braces problem" as it is known in the Arden Syntax context. This includes lack of a standard data model, variations in data query syntax and use of different clinical terminologies from one site to another to code data. Work is underway to address this directly in Arden Syntax v3. Despite improvements that permit better alignment between Arden Syntax constructs and structured clinical data such as the introduction of object dot notation, no overall standard data model has been identified that would allow queries and other mappings to be expressed in a completely shareable way and thus help solve the "curly braces problem." The Virtual Medical Record (vMR) was developed by HL7 precisely as a standard data model specifically for use in providing CDS. Moreover, this standard has been implemented in a few production systems, and there is evidence that the vMR would suffice in allowing the representation of most query data elements in typical MLMs [15]. Finalization of the effort to replace the curly braces is planned for the next instantiation of the Arden Syntax, v3.0, but it may use FHIR instead of the vMR given the popularity of this next-generation standard among system implementers.

In addition, the curly braces challenge may be sidestepped by sharing knowledge through service-oriented access instead of knowledge transfer. In this way, health information systems can invoke executable expert knowledge at a central point instead of having to execute it on a variety of different vendor systems, thus avoiding the need for a standard syntax. HL7 standards such as the Decision Support Services (DSS) standard provide a standard syntax for accessing knowledge servers or a remote CDSS and contribute to ongoing uptake of CDS to improve the quality of care, thus complementing the role of the Arden Syntax in this regard [16].

5. Conclusions

The Arden Syntax is a robust standard for representing executable clinical knowledge that facilitates knowledge sharing and thus the provision of CDS. It has evolved from the relatively simple yet powerful formalism documented in its first standard version in 1992 to include advances in control flow, operators, data structures and models, external linkages and other aspects. The result has been an increasingly powerful formalism for expressing complex and nuanced clinical reasoning and clinical data that, when combined with tools for editing and testing knowledge, facilitates the implementation and uptake of CDS.

Acknowledgements

Dr. Jenders is supported in part by grants U54MD007598 and 5S21MD000103 from the National Center for Minority Health and Health Disparities as well as UL1TR001881 from the National Center for Advancing Translational Sciences, US National Institutes of Health.

References

[1] Hripcsak G, Ludemann P, Pryor TA, Wigertz OB, Clayton PD. Rationale for the Arden Syntax. Comput Biomed Res 1994;27(August (4)):291–324.
[2] American Society for Testing and Materials. Standard specification for defining and sharing modular health knowledge bases (Arden Syntax for Medical Logic Systems). In: Designation E1460-92. West Conshohocken, Pennsylvania: ASTM; 1992.
[3] Jenders RA, Huang H, Hripcsak G, Clayton PD. Evolution of a knowledge base for a clinical decision support system encoded in the Arden Syntax. Proc AMIA symp 1998:558–62.
[4] 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.
[5] 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.
[6] Jenders RA, Hripcsak G, Sideli RV, DuMouchel W, Zhang H, Cimino JJ, et al. Medical decision support: experience with implementing the Arden Syntax at the Columbia-Presbyterian Medical Center. Proc Annu Symp Comput Appl Med Care 1995:169–73.
[7] Health Level Seven International. The Arden Syntax for Medical Logic Systems Version 2.10. Ann Arbor, Michigan: HL7; 2014.
[8] Health Level Seven International. HL7 Arden Syntax Implementation Guide Release 1. Ann Arbor, Michigan: HL7; 2014.
[9] Vetterlein T, Mandl H, Adlassnig K-P. Fuzzy Arden Syntax: a fuzzy programming language for medicine. Artif Intell Med 2010;49(1):1–10.
[10] Jenders RA. Utility of Arden Syntax for representation of fuzzy logic in clinical quality measures. Stud Health Technol Inform 2015;216:1096.
[11] Jenders RA. Evaluation of the Health Level Seven Fast Health Interoperable Resources (FHIR) standard as a query data model for the Arden Syntax. AMIA Annu Symp Proc 2014:1438.
[12] Jung CY, Sward KA, Haug PJ. Executing medical logic modules expressed in ArdenML using drools. J Am Med Inform Assoc 2012;19(July (4)):533–6.
[13] Jenders RA, Dasgupta B. Assessment of a knowledge-acquisition tool for writing medical logic modules in the Arden Syntax. Proc AMIA annu fall symp 1996:567–71.
[14] Jenders RA, Dasgupta B. Challenges in implementing a knowledge editor for the Arden Syntax: knowledge base maintenance and standardization of database linkages. Proc AMIA Symp 2002:355–9.
[15] Jenders RA. Evaluation of the Health Level Seven Virtual Medical Record standard as a query data model for the Arden Syntax. AMIA Annu Symp Proc 2013:701.
[16] Kawamoto K, Honey A, Rubin K. The HL7-OMG Healthcare Services Specification Project: motivation, methodology, and deliverables for enabling a semantically interoperable service-oriented architecture for healthcare. J Am Med Inform Assoc 2009;16(November–December (6)):874–81.