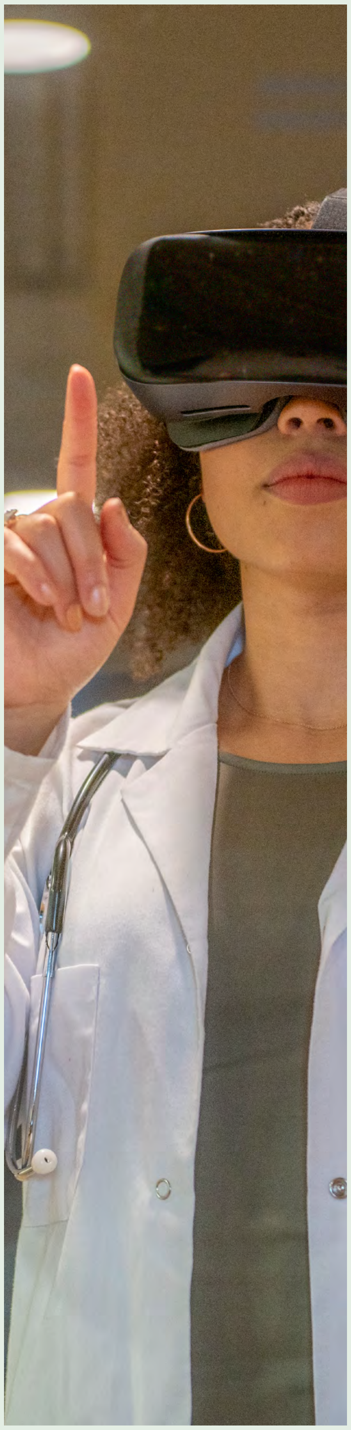
WHITE PAPER



AN INSIGHT INTO CREATING DOMAIN SPECIFIC LANGUAGES (DSLS) FOR MEDICAL DEVICES

Infosys®
Navigate your next

1. Introduction

Domain Specific Languages (DSL), existent since the beginning of computing, are languages designed to address a specific problem. Some well-known examples include SQL, HTML, make, regular expressions, UML and VHDL. An expert who knows the semantics of a particular domain can express its requirements using a DSL. Thus, DSLs help reduce the communication gap between the domain expert and the software developer.

DSLs are designed to factor in application stability and performance, essential considerations when building software that meets regulatory requirements. For example, the financial, medical, and telecommunication domains are some regulatory domains where DSLs are used.

Medical device manufacturers rely on DSLs to derive productivity, lower cost of development and decreased time to market as they create derivative products. This document presents the considerations for creating DSLs for medical device software development.

The document provides an overview of DSLs and enumerates some best practices in DSL design for medical device software. These views are derived from the author's experience developing Class 3 Medical device software using DSL.

In addition, this document does not cover the following -

- DSL implementation details such as DSL patterns. More detailed sources are available, including one in the Reference section[8].
- Details of building a parser as creating a new language will need parsing of the syntax. Again, the reader can refer to document [9] in the Reference section.

2. Target Audience

The document targets medical product designers and medical application developers who intend to use DSLs. It explores the benefits of using DSL for developing software for medical devices. It also presents potential design solutions derived from the author's experience, which can be utilized.

3. Usage of DSL – A landscape view

- DSL has been used in multiple domains. Some examples from the financial domain are:[1]
- Actulus Modeling Language (AML) is a platform for defining advanced life insurance and pension products and for efficient computations.
- Financial Products Markup Language (FpML) - an XML-based standard used for electronic dealing and processing over the counter (OTC) derivatives.
- Money - a Scala DSL for money-related operations, the language automatically performs conversions between currencies as per the exchange rates specified.
- Some of the popular DSLs that are widely used today are - [2]
- SQL – A standard language for storing, manipulating and retrieving database data.
- HTML - The standard markup language for defining the meaning and structure of web content.
- Interface Definition Language (IDL) - is a generic term for a language that lets a program or object written in one language communicate with another program written in an unknown language. For example, an Object Request Broker (ORB) program would use an interface definition language to "broker" communication between one object program and another.

4. Why are DSLs created?

- There are several reasons why DSLs are created, including -
- To express a solution to a specific problem more intuitively in a particular domain.
- Have powerful abstractions that solve complex domain-specific problems in a simple manner.
- Document complex business requirements and or application behavior of one domain that is machine readable.
- Domain experts can quickly review functional requirements written in DSL by software developers.

5. What to expect from a DSL

5.1 A view of a use case realized with a DSL and a general-purpose programming language

USE CASE: Let us try to get the hottest temperature in Atlanta for August.

You can see how eloquently this use case is achieved using a DSL such as SQL.

SELECT MAX(value) FROM TEMPERATURES WHERE city="Atlanta" AND month="August";

See the difference in the same use case achieved through a general-purpose programming language.

City myCity("Atlanta");

MyList Temperatures;

Temperatures.setList(myCity.Month("August").getTemperatures());

int result = Temperatures.getMax();

cout<<"The Max temperature in Atlanta in the month of August is "<< result << endl;

5.2 Advantages

There are several ways in which DSL is the better option as they:

- Are more concise
- Can provide powerful abstractions that help address complex domain-specific problems in a simple manner
- Can be written more quickly and are intuitive in nature
- Can generate code and bring significant productivity improvements
- Allow domain experts to write application requirements
- While the initial cost of DSL development is high in comparison to a General Purpose Language (GPL), it can be seen that the cost of development is lower when a DSL is used over a period.[2]

5.3 Disadvantages

- The higher initial cost of development is mainly due to the creation of the one-time tool used by the DSL.
- A DSL has a learning curve, and creators must create specific training material.
- Debugging paradigms can be specific to the DSL used.
- DSL creators need domain knowledge and good language-development knowledge; this is difficult to find.

6. DSL Implementation Approach.

A language in which the DSL is processed is termed the Host Language. A DSL can be created using the following two approaches. [3]

- Internal DSL uses a host language in a particular way to give the host language the feel of a DSL. They are also referred to as embedded DSLs or fluent interfaces. A fluent interface is easily readable, like an English language sentence.
- External DSL has its own syntax; another variation is to encode the DSL syntax in XML and uses regular XML parsers. The syntax can be very powerful and processed using a YACC or ANTLR parser.
- DSLs can be implemented using interpretation or Code Generation, although it is easier to interpret the DSL.

7. Design considerations when creating DSLs for medical device software

Medical device software can significantly benefit from the usage of medical DSLs. It is beneficial in promoting close collaboration between medical domain experts and application programmers. Consider creating a DSL for a medical device application with the following key functionality.

- User interface
- Network based device communication to read or change the state of the medical device
- Storage of application session context
- This feature set will influence the design choices.

7.1 Language to base the DSL on

The following language choices would be appropriate.

- XML is suitable since transforming medical device data will be essential to present to the end user.
- HTML will facilitate the display of charts or other standard user interface components like buttons, checkboxes and lists.

7.2 Define the medical device requirement using DSL.

Medical device features and requirements are specific to the domain they address, e.g., insulin pumps, cardiac implants and neural implants.

Most device requirements involve algorithms and other computations that are core parts of the device manufacturer's IP. Therefore, the DSL should represent these core device requirements. When this is done, medical domain experts can easily verify the requirements.

The other significant advantage of representing the domain requirements using a DSL is that the software architecture is partitioned into the domain and other generic application requirements.

7.3 Data transformation

Medical devices typically use embedded platforms where memory is often a constrained resource. Hence data generated by these devices would be stored compactly and needs to be transformed into a human readable form. The DSL can represent the data stored in the device using a markup language such as XML. A layered architecture can be designed to transform data as required by the services that consume it.

7.4 Network communication

Medical devices use proprietary protocols to communicate with applications that manage them. DSL is well suited to describe this proprietary communication. This facilitates a closer collaboration and a more effective review between the firmware engineers who define the device communication and the software engineers who work on applications that manage the device.

7.5 User experience design

Medical device applications ensure that the end user can input only valid data. A few user experience design considerations include the following -

- Restricting user inputs to acceptable values - DSL effectively defines rules limiting the input to acceptable values and providing a simple abstraction that application developers can use.
- Display of UI screens with their transitions- A DSL can represent the screens and their layout along with the widgets used in the screens. It can incorporate data connections that connect the input data sources to the widgets that display this data. Action to be taken on click of the widgets along with screen transitions can be programmed using the DSL.

7.6 Application Events

Medical device applications typically log and display application events that have an impact on patient safety. These application events may be triggered based on specific rules and interdependencies between states of different system parameters. A DSL will be able to represent these conditions and provide an abstraction that is easy for the application developer to use.

8. DSL tools

A set of tools are needed to build DSLs, such as: [4]

- Textual DSL - If the DSL is based on XML, then off-the-shelf XML parsers can be used. The XML elements could contain complex expressions, which could be parsed by a parser such as YACC or ANTLR.
- Graphical DSL - Graphical languages are intuitive, and domain experts feel more at ease than textual languages. However, graphical languages require building specific editors to be used, such as Eclipse Sirius.

8.1 Some of the tools for developing DSLs

A few of the tools used to develop DSLs are -

Xtext – an open-source tool used to build textual languages, it is integrated with Eclipse. It includes a parser, linker, type checker and compiler. [5]

textX – a tool to build textual languages, textX is a Python framework inspired by Xtext. The language's grammar can be defined with a syntax like the one used by Xtext. [6]

Sirius - an Eclipse project that helps create a graphical modeling workbench using Eclipse Modeling technologies, including Eclipse Modelling Framework (EMF) and Graphical Modelling Framework (GMF). This can be used for building Graphical Languages. [7]

8.2 Debugging the DSL

If the DSL is external, a mechanism for debugging the DSL must be created. Since parsers are used as part of the execution of the DSL, it is easy to create a user readable log of all the events the application handles and states that the application transitions into. This log would be the primary source of debugging the application. In addition, DSL users would need training on how to use these logs.

8.3 Unit Test and Test Automation

Test framework can be designed usin g DSL to load test inputs, perform user desired operations and compare the test output against expected values. Furthermore, a test harness can be created to automate the tests written using the DSL.

9. Summary

Here is a recap of the best practices discussed in the document -

- DSLs can provide powerful abstractions that help address complex domain specific problems in a simple manner.
- Medical device software development can significantly benefit from the usage of DSLs. In addition, this will promote close collaboration between medical domain experts and application programmers.
- Medical device requirements that are part of the device manufacturer's IP can be well represented using DSL. In addition, medical domain experts can now easily verify these requirements resulting in higher software quality.
- Medical device applications try to enforce data integrity at various levels starting from user input. A DSL effectively defines rules that can enforce these input constraints at different software layers and provide simple abstractions that application developers can use.
- The initial cost of DSL development and tooling will be higher than traditional software development. However, the software development costs will decrease over a period due to increased developer productivity. In addition, the costs of releasing different medical device product families will be lower as it will be easier to make an incremental change in the DSL based requirement specifications.

Using DSL to create medical devices can transform medical product software development. It helps bridge the gap between domain experts and software developers, making domain experts central to medical device software development. As a result, it can drive innovation and cost optimization in the way medical devices are created. Successful creation of medical devices using DSLs can be achieved through careful language design and by using the best practices followed by industry leaders, as discussed in the paper.

References:

[1] http://dslfin.org/resources.html

[2] http://cs448h.stanford.edu/DSEL-Little.pdf

[3] https://martinfowler.com/dsl.html

[4] https://tomassetti.me/domain-specific-languages/

[5] https://www.eclipse.org/Xtext/

[6] http://textx.github.io/textX/3.1/

[7] https://www.eclipse.org/sirius/overview.html

[8] https://martinfowler.com/dslCatalog/

[9] https://www.antlr.org/

About the author

Balaji Lakshmi Narasimhan

Senior Principal Technology Architect, Infosys Engineering Services