VizDSL: A Visual DSL for Interactive Information Visualization

Rebecca Morgan[1] *, Georg Grossmann[1], Michael Schrefl[1], Markus Stumptner[1], and Timothy Payne[2]

[1] University of South Australia, Adelaide, SA, Australia, 5000
[2] Lockheed Martin STELaRLab, Edinburgh, SA, Australia, 5111

Abstract. The development of systems of systems or the replacement of processes or systems can create unknowns, risks, delays and costs which are difficult to understand and characterise, and which frequently result in unforeseen issues resulting in overspend or avoidance. Yet maintaining state of the art processes and systems and utilising best of breed component systems is essential. Visualization of disparate data, systems, processes and standards can help end users to understand relationships such as class hierarchy or communication across system components better. There are many visualization tools and libraries available but they are either a black box when it comes to specifying possible interactions between end users and the visualization or require significant programming skills and manual effort to implement. In this paper we propose a visual language called VizDSL that is based on the Interaction Flow Modeling Language (IFML) for creating highly interactive visualizations. VizDSL can be used to model, share and implement interactive visualization based on model-driven engineering principles. The language has been evaluated based on interaction patterns for visualizations.

Keywords: Model-Driven Visualization, Domain-Specific Modelling, Interactive Information Visualization, IFML

1 Introduction

Modern operational systems are often large, complex and composed of distributed systems of heterogeneous components. Lack of integration and interoperability is costly and often not discovered until late in the development process. Identification of issues, such as non-matching concepts between specifications, or just getting an understanding of the scope of an implementation effort is a significant challenge and can be achieved using modelling over different abstraction levels [1] and domains [2]. An example system discussed in this paper lies in the digital hand over of design documents to the operation and maintenance

* Rebecca Morgan was supported by the Australian Government Research Training Program Scholarship and by Lockheed Martin Australia.

area in the integrated energy industry using specific standards. The information contained in these standards is very complex, often spanning several parts with hundreds of pages containing several hundred concepts and relationships. Understanding the specifications is extremely challenging and time consuming, even for domain experts like engineers in the energy industry who need to approve what kind of information is passed between information systems and that the information is semantically correct and represented by those standards. Visualization of the standard specifications can be used to reduce the time taken to gain this understanding.

While engineers in this space are reasonably fluent in data modelling and data mapping concerns they are not IT experts or developers who are required to create customised visualizations. Currently, customised visualizations are often hard coded at a low level requiring domain, IT and programming expertise. The solution cannot be reused for other visualizations and is a black box where it is not clear how the visualization and interactivity is specified. If commercial visualization tools are used instead, significant difficulties can be faced in terms of interoperability and extensibility, communication of visualization design between users, product lifespan and support.

The goal of our research is to address these issues by developing a visualization tool based on model-driven engineering principles that creates interactive visualizations quickly for end users to explore and understand complex semantic structures. In this paper, we describe a *platform-independent* and extensible modelling *language*, VizDSL, which allows non-IT experts to describe, model and create interactive visualizations, quickly and easily.

The paper uses a case study from the energy industry based on our ongoing project work which motivated this research and also provides realistic large scale industrial datasets [3]. Lockheed Martin Australia is now supporting this project because of similar requirements in the visualization of complex system architecture frameworks.

The contribution of this paper is threefold: (1) compared to [4] in which we introduce VizDSL, we describe in this paper the detailed metamodel of VizDSL, (2) discuss the implementation and (3) evaluate the implementation based on the visual patterns identified by Heer et al.[5]. In the following, we discuss first the background of the VizDSL approach and related work and then describe the metamodel and the evaluation in more detail.

2 Background

Ensuring interoperability in the face of increasingly large and complex enterprise systems is challenging and expensive. Model driven engineering (MDE) is a way of addressing the difficulties inherent in dealing with complex system architectures. MDE focuses on the model as the primary development driver, rather than the code [6]. Interoperability is assisted by the model being independent of the implementation platform, as well as the provision of customised and reusable software. In MDE, the code is generated directly from the model, as opposed to

traditional code-driven development, where the model is referred to separately as an information source during the primary process of writing the code. In MDE, Domain Specific Languages (DSLs) are used to represent domain models in a relatively concise and efficient way [7] leading to greater productivity and reduced costs. In this context, DSLs are used in conjunction with transformation engines/code generators to generate required artifacts such as source code [8]. DSLs with modelling support are used as part of domain specific development, for effective code generation from the model artifacts.

IFML An example of a DSL with modelling support is the Interaction Flow Modeling Language (IFML)[3]. IFML uses a graphical notation to create visual models of content, user interaction and controls for front end applications. IFML provides a modelling language which sits between a user interface and the code. IFML has been recently accepted as a standard by the OMG for front-end design and it is well established through its predecessor WebML. A commercial implementation of IFML is available through WebRatio[4]. IFML can be extended through its specification written in UML.

Visualization Visualization amplifies cognition [9] and facilitates knowledge sharing between different user groups with different levels of expertise and experience, without requiring extensive background knowledge or training [10–15]. There is a growing need to support information visualization in a way which is compatible with MDE. Visualization techniques are used as part of MDE to visualise the code, the problem domain and the models used to describe the domain. Models are frequently represented graphically, as a formal specification using graphical syntax, or in an informal sense as a quick means of communicating key features and ideas. Knowledge of underlying semantic structures is of fundamental importance in MDE. This knowledge is collected and shared by visualising semantic information, such as hierarchical, relational or entity-based semantic information, or some combination of these [16].

Although there has been significant research conducted in the areas of visualization tools and techniques, there are no existing visualization tools which can be used to create, share and implement visualizations using model driven techniques and which are based on accepted standards.

To address these gaps, we have developed a graphical DSL which facilitates the creation of interactive visualizations called VizDSL [4]. In [4], we introduced the concept of VizDSL without providing extension details. Since VizDSL is platform-independent and extensible through its UML profile, it is important to provide IFML extension details, to aid implementation in the community. In this paper, we describe the metamodel with the visual syntax of VizDSL and its implementation using the Domain Modelling Environment (DoME) and the

[3] http://www.ifml.org
[4] https://www.webratio.com/site/content/en/home

Agile Visualization package[5]. Further it includes an industry application and evaluation of the framework.

VizDSL takes a model-based approach rather than a procedural approach to the design process, to meet integration and interoperability requirements in the context of MDE. Since VizDSL has been designed with the aims of MDE in mind, strong support for semantic visualization is provided, including the ability to design interactive visualizations with a hierarchical navigational structure. Before we describe VizDSL in more detail, we discuss its role in the engineering of information systems within an oil & gas industry project.

2.1 Industry Application

To provide context for the application of the proposed DSL, the following section discusses a real life case study which provided the motivation for developing VizDSL. The Open Industrial Interoperability Ecosystem (OIIE)[6] provides a supplier neutral interoperability ecosystem, aimed at asset intensive industries such as the defense industry and the integrated energy industry. The OIIE framework uses system-of-systems interoperability and employs a portfolio of standards for components of representative business processes. These standards are selected according to industry requirements, adoption and community engagement. One instance of the OIIE is found in the Oil and Gas Interoperability (OGI) Pilot[7]. The OGI Pilot is a public collaborative interoperability test bed for the OIIE within the oil and gas industry. Participating organisations in the OGI Pilot include BP, SAP and Yokogawa Electric.

An area of active interest in the OGI Pilot lies in digital information handover, for example, the transfer of operations and management related information from Engineering, Procurement and Construction (EPC) systems to systems that owner/operators are using. In order for safe plant operation and production to occur, operations and management systems must be populated with structural information about the plant, including operating parameters, product data and serialized assets. Businesses in the design domain generally use the ISO 15926 standard for information models, while businesses in the operations and maintenance domain generally use MIMOSA.

For interoperability, the standards must be mapped to each other as part of the digital information handover process. Due to the complexity of the standards information, it is necessary to identify and visualize the hierarchical structure obtained from formal standard specifications, quickly and efficiently, so that this mapping can be established.

2.2 Industry Requirements

From our previous work [4] and from the identified organizational problem as discussed in Section 2.1, the following requirements have been identified:

[5] `http://agilevisualization.com`
[6] `http://www.mimosa.org/open-industrial-interoperability-ecosystem-oiie`
[7] `http://www.mimosa.org/oil-and-gas-interoperability-ogi-pilot`

R1 Semantic visualization using MDE techniques: Although there are a significant number of visualization solutions available, these solutions tend to rely on procedural programming techniques, with or without graphical user interfaces [4]. Procedural programming techniques are not as well suited to representing information in terms of entities, relationships and hierarchical structure as the model driven approach due to the lower level of abstraction.
Requirement: There is a need for a graphical DSL which can be used to model and implement interactive visualizations. This DSL should be able to visualise complex information such as that contained in the OIIE standards and should be able to visualise semantic structures as well as data content.
R2 Creating complex interactive visualizations without programming experience: With respect to the motivating example given above, the information contained in the standards is currently represented using static visualizations which have been hard coded by software engineers. This approach is problematic. The size and complexity of the information being presented means that static visualizations are not very effective in matching concepts. The structure of the visualization, what it will represent and how it will be represented, has to be inferred directly from the code, which leads to difficulties in communication for non-IT experts, such as design engineers. It would be preferable, in this case, if they are able to create their own visualizations without the need to rely on a third party with necessary programming skills.
Requirement: There is a need for a visualization DSL which can be used by non-IT experts to quickly and easily create highly interactive visualizations.
R3 Support for standards-based interoperability: The problems faced in information handover in the OGI Pilot are further complicated by the fact that the OGI Pilot, as part of the OIIE, employs system-of-systems interoperability. This implies that a visualization solution must remain independent while able to act in concert with other systems used in the OGI Pilot. Standards are employed in interoperable systems to ensure a common grounds for information exchange.
Requirement: There is a need for a visualization DSL which can be employed as part of a heterogeneous system of interoperable components, and which should be based on a standard to increase its adoption.

3 Related Work

There has been significant interest in the field of visualization in recent times. In the following section, we discuss Model Driven Visualization, a framework for the creation of visualizations using concepts from MDE. VizDSL fits into the Model Driven Visualization framework as a DSL which describes software visualizations. We give a brief overview of DaisyViz, another model-based approach to information visualization and briefly discuss CloudMap and RALph, as examples of graphical DSLs which follow a similar structure to VizDSL, although they are not related to visualization as such.
Model Driven Visualization: One proposed approach to visualization in MDE is Model Driven Visualization (MDV) [17–19]. MDV is a model based,

platform independent approach to creating visualizations which uses the MDE concepts of metamodels and transformations to describe the structure of visualization tools. One of the open problems identified in [18] was the lack of a suitable transformation language; support was requested to help build a suitable DSL for describing software visualization.

The MDV reference architecture recommends meta-models for the domain and the viewers. Bull [19] noted that there was a need for a DSL for describing software visualization, which can be used for describing translations between domain and viewer meta-models. In [19], it was noted that the work on MDV had focused primarily on the static data structures behind the specification of a view model and that modelling languages could support a broader range of options for view control and behaviour which is addressed in part by VizDSL.

In comparison to presented work, MDV provides a framework for visualization in the context of MDE but cannot be used to create visualizations. VizDSL provides a language to model and execute model-driven visualization.

DaisyViz: DaisyViz [20] is a model-based user interface toolkit for the development of domain-specific information visualization applications without programming. DaisyViz uses three declarative models: data, visualization and control. The data model uses relational database schema to manage data, the visualization model is used to define the visual representation of data and the control model is used to describe the information visualization tasks and techniques used in the views. While DaisyViz has a textual syntax, VizDSL uses a visual modelling language, which improves understandability for non-IT experts.

CloudMap: CloudMap [21] presents a visual notation for representation and management of cloud resources. Notational constructs include entities, links, probes and control actions. These notation constructs were identified and subsequently specified after user surveys. CloudMap implements an interactive mindmap visualization for the navigation of cloud resources, detection and display of events (event management system) and manual and automated actions. The end results were promising, with significantly improved efficiency. CloudMap focuses solely on the visualization of cloud resources whereas VizDSL can be used to visualize any schema or structured data.

RALph: RALph [22] presents a graphical notation for visual modelling of resource selection conditions in process models. The underlying semantic concepts were identified on previous experience and case studies. RALph was integrated into a platform which uses BPMN and provides a graphical editor based on Oryx. Evaluation of RALph is not currently available, however, the authors intend to evaluate the understandability and learnability of RALph in the future, using the Physics of Notation [13]. As with CloudMap, RALph is restricted to a particular domain, in this case, resource assignments in business processes. VizDSL is general purpose in that it is not limited to any particular information domain.

4 VizDSL

In this section, an overview of VizDSL, the graphical notation for VizDSL and discussion of the proposed VizDSL extensions to IFML is given.

VizDSL takes a model driven approach to visualization in the form of a DSL which can be used to model interactive visualizations with a focus on representation of semantic information. VizDSL takes the form of a graphical language since there are cognitive benefits to using graphical languages [23] and graphical languages are more effective in conveying information to non-IT experts [13]. To ensure standards-based interoperability, VizDSL makes use of existing standards. A natural candidate to form the basis of VizDSL is the Information Flow Modeling Language (IFML) released by the OMG: (1) it is a standard, (2) follows MDE principles and (3) can be used to specify user interactions. However, IFML originated from the WebML language aiming at model-driven Web engineering and not the development of interactive information visualizations. Hence, the first step in the development of VizDSL was to identify how to extend IFML with appropriate concepts such as *Data* and *Interaction* for modelling interactive visualizations of data specifications. The goal of this research was to develop a modelling language that is independent from the implementation of a visualization and can be executed by different visualization libraries.

For testing the execution of the language the *Agile Visualization* framework was chosen because it is open source, highly customisable and has strong out of the box support for interactions, animations and web browser integration. It is flexible enough to create completely new visualization techniques. The first step and also challenge in the design of VizDSL was (1) to identify the gap between IFML and Agile Visualization concepts and (2) the integration of both. In some cases the IFML metamodel could be extended with the visualization concepts in a straight forward manner, e.g., place the *ViewContainer* from Agile Visualization as subclass of *ViewElement* in IFML. However, in some cases it was not trivial, e.g., IFML has no or only limited support when it comes to direct representation of data sources including layouts and themes and how they should be associated with other concepts in the metamodel. Below we discuss the extensions of IFML which represent the VizDSL metamodel:

In Fig.1, the core visual notation for VizDSL is shown. The visual notation is based on the IFML visual notation (shapes in grey), with new notations for the proposed extensions (shapes in white).

In Fig.2, the class diagram for VizDSL is shown. The VizDSL classes from top left to bottom right which can also be found in IFML are *Parameter*, *ParameterBinding*, *ParameterBindingGroup*, *ViewElement*, *ViewComponentPart*, *Event*, *Action*, *ViewComponent* and *ActionEvent*. These classes have been extended in VizDSL with additional attributes. New classes introduced by VizDSL from top left to bottom right in Fig.2 are *Data*, *Layout*, *Theme*, *Menu*, *Shape*, *MenuItem*, *Element*, *Edge*, *Interaction*, and *Animation*. Remaining classes are

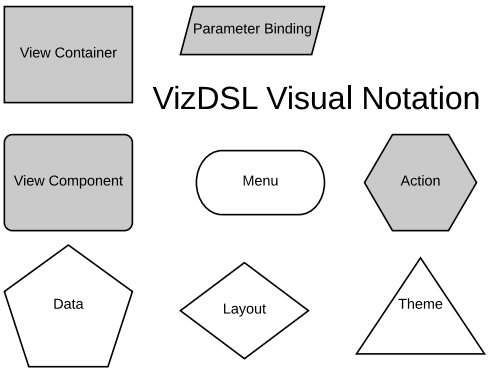

Fig. 1. VizDSL visual notation

VizDSLObject and *ViewContainer* are IFML classes which have been modified for VizDSL [8] . Below some of the classes are described in more detail.

VizDSLObject: This is the VizDSL root class. Like in IFML, which has the root class *InteractionFlowElement*, a root class is required in VizDSL that captures generic attributes that are inherited by subclasses, e.g., *parameter* relationship attribute which maps data to visual objects (see also below about the *Data* concept).

ViewContainer: *ViewContainer* is present in IFML and was extended to represent complex structures by adding new subclasses *Structure*, *Relationships*, *Comparison* and *Detail*. Generic concepts *Elements* and *Edges* are rendered on *ViewContainers*, so a relationship was required to model this. Further, *ViewContainers* are associated with *Layouts*, a *Theme*, *Menus*, *Interactions* and *Animations*.

Data: One of the significant extensions to IFML is the *Data* class. Although IFML provides means of describing content and data, it does not contain an explicit concept to describe them. This is required to directly represent data source as part of the visualization design process, rather than having to create another data model using a different language. The *Data* class is associated with the IFML *Parameter* class, which provides a way of mapping data to visual objects represented using the *VizDSLObject* class.

Layout and **Theme**: These two classes were introduced to provide users with the capability to use different layout algorithms (*Layout*) and to create a consistent style for multiple views (*Theme*) as there is no way for defining layouts in themes in IFML.

Menu and **MenuItem**: Interactive menus were not directly supported by IFML but are required for interactive visualizations. As with *Themes*, *Menus* are reusable and can be associated with multiple *ViewContainers*.

[8] In a color-printed version of this paper the IFML classes with extended attributes are in black, new introduced classes are in green and modified classes are in red.

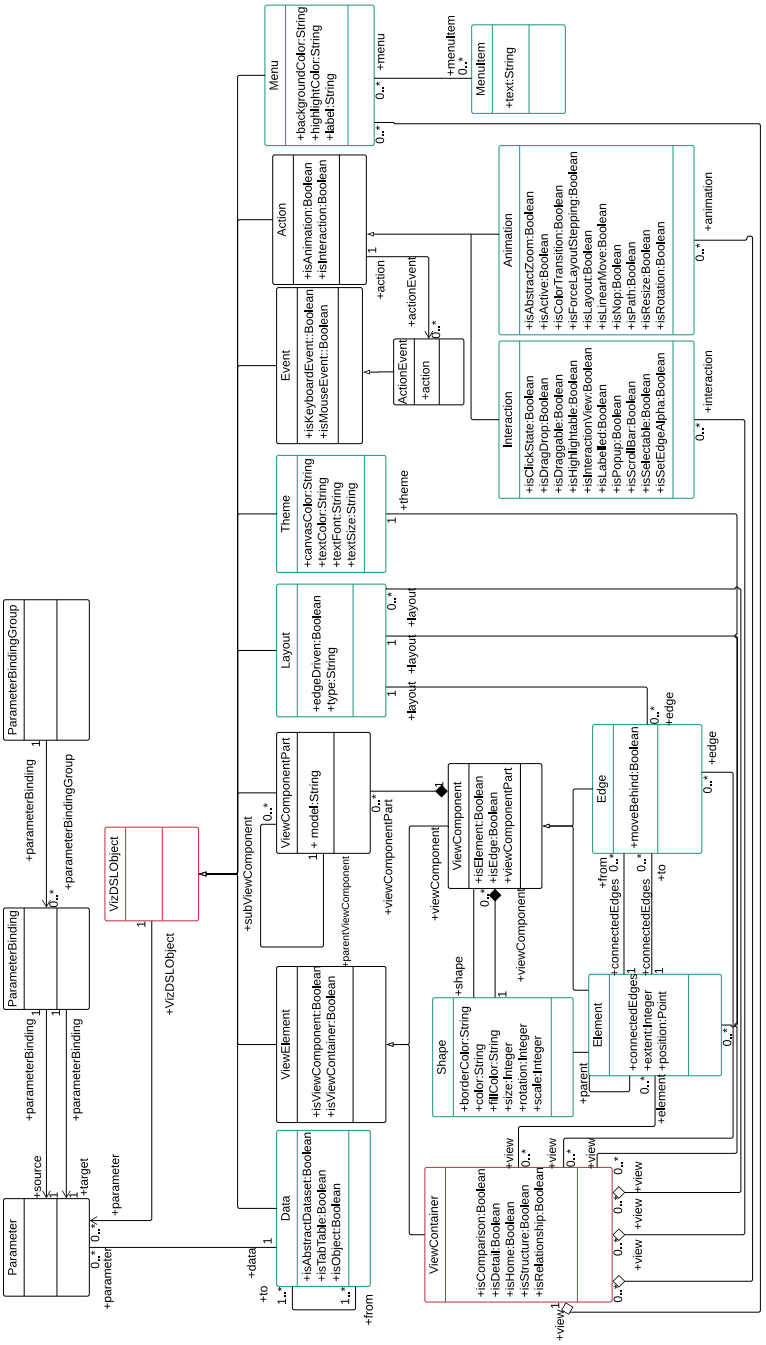

Fig. 2. VizDSL class diagram

Shape: The *Shape* class was introduced as a means of defining the visual attributes of the primary visual components for the proposed VizDSL *ViewComponent* classes: *Elements* and *Edges*. Each *Element* and *Edge* is associated with a *Shape*, which is used to map visual attributes to the final display.
Element: The *Element* class was introduced to provide visual representations of entities, which was identified as a requirement for the VizDSL language in Section 2.2. *Elements* can be associated with multiple *ViewContainers*, *Edges* and with other *Elements*, by nesting.
Edge: Together with *Elements*, *Edges* make up the fundamental visual components of a VizDSL visualization. *Edges* represent relations between entities, which is one of the requirements for VizDSL as identified in Section 2.2. Each *Edge* is from one *Element* to another *Element* and is associated with a *Shape*, which provides the visual representation of the *Edge*. As with *Elements*, *Edges* can be associated with multiple *ViewContainers*. Each *Edge* is associated with a *to* and *from* element.
Interaction: The *Interaction* class is a child of the IFML *Action* class. *Interactions* are triggered by IFML *Events*. *Events* are occurrences which affect the state of the application, such as clicking on the mouse. When an *Event* occurs, an *Action* is triggered. The *Interaction* class was introduced in VizDSL to represent required user interactions, such as selection, drag and drop, dragging, highlighting, labelling, popups, scrollbars and zooming.
Animation: As with the *Interaction* class, the *Animation* class was introduced as a child of the IFML *Action* class. Animations are well supported in Agile Visualization and can provide a greater level of engagement with visualizations. *Animations* are triggered by an *Event* and include abstract zoom, color transitions, layout transitions and stepping, resizing and rotation.

5 Implementation

In the following section, details are given for a prototype implementation of VizDSL using DoME in conjunction with the Roassal visualization engine that implements Agile Visualization. For this particular implementation, the Roassal2 package was used in the VisualWorks development environment. Roassal also works in Pharo, an open source Smalltalk environment; VisualWorks was chosen because DoME is only available in VisualWorks.
Domain Modelling Environment (DoME): We used the DoME for a prototype implementation of VizDSL. DoME is an integrated set of model-editing, metamodelling and analysis tools. We are using DoME to extend IFML and create an editor for our proposed VizDSL. In this editor we can create models that are executable and generate interactive visualizations in Agile Visualization. DoME is available through VisualWorks[9], the commercial implementation of Smalltalk. The VizDSL metamodel was defined using DoME, in order to create interactive model editors which can be used to define VizDSL diagrams.

[9] `http://www.cincomsmalltalk.com/main/products/visualworks`

Agile Visualization: The Agile Visualization package was used to render visualizations. Agile Visualization is written in Smalltalk and is available in Pharo, the open source Smalltalk implementation, or in VisualWorks, the commercial Smalltalk implementation. Agile Visualization was originally developed for the visual analysis of software systems and takes an object-oriented approach to visualization. Agile Visualization was chosen because it supports the identified requirements except for providing a graphical language (for which we can use IFML) and it provides basic building blocks to create the majority of existing visualizations.

6 Evaluation

We have evaluated the framework against the requirements mentioned in Section 2.2 and against software design patterns in information visualization. Before we discuss this in more detail, the industry application is demonstrated on the visualization of ISO 15926.

6.1 Industry Application: Visualization of ISO 15926

To illustrate the capabilities of VizDSL, we have used VizDSL to model and implement an interactive visualization of the ISO 15926 standard. As mentioned in Section 2.1, the ISO 15926 standard is generally used by the EPC domain. The information contained in ISO 15926 is complex and visualization of the hierarchical structure is necessary for interoperability. The ISO 15926 standard describes 320 classes with associated attributes and relationships. The relatively high number of classes means that static visualizations of the hierarchical structure are cluttered and difficult to understand. In this case, it is more useful to create an interactive visualization with drill-down navigation, since this eliminates unnecessary visual clutter and allows users to focus more effectively on areas of interest.

VizDSL was used to model and implement an interactive visualization of the ISO 15926 hierarchy (shown in Fig.3). In Fig.3 the *Data* source refers to the ISO 15926 classes and subclasses, which are assigned to a collection named *classes* through a parameter binding group. There is one *View Container* for this visualization, containing *Elements* on the *classes* collection, colored according to the number of subclasses, with *Edges* from each *Element* to its superclass. Interactions include element selection and details on demand. Menus include node expansion for drilling down, *Theme* and *Layout* selection and element finder.

6.2 Evaluation against requirements

R1 Semantic visualization using MDE techniques: From the example implementation described above, it is evident that VizDSL meets this requirement, as it was successfully used to model and execute an interactive visualization of the complex information contained in the ISO 15926 standard.

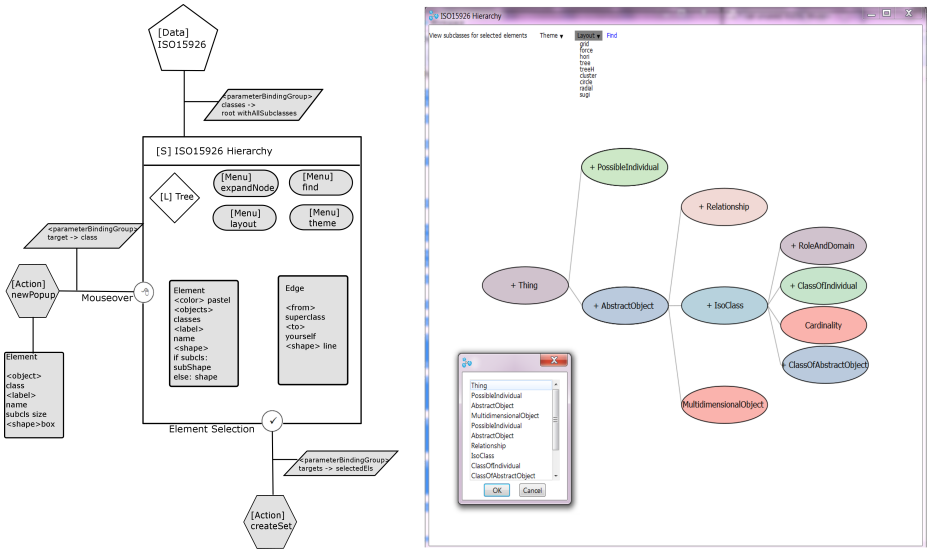

Fig. 3. VizDSL model for an interactive visualization of the ISO 15926 hierarchy

R2 Creating complex interactive visualizations without programming experience: It is important to find a balance between understandability and expressiveness when developing a domain specific modelling language. One of the primary drivers behind the development of VizDSL is the need for non-IT experts to quickly and easily create complex interactive visualizations and as such, understandability is of fundamental importance. IFML, on which VizDSL is based on, was designed for the web development community, which is multidisciplinary and covers a broad spectrum of IT and non-IT skills [24], similar to our target audience. The design principles behind IFML which support model usability and understandability (conciseness, extensibility, implementability, reusability, default modelling patterns and details) is transferred to VizDSL as an extension of IFML.

One of the primary advantages of MDE is the ability to represent complex systems at a relatively high abstraction level, which increases understandability. Again, the example described above highlights that this was possible without writing a single line of code and cannot be achieved with any framework mentioned in the related work section. A comparable visualization to the example above in the popular D3.js JavaScript library is the *d3.layout.tree* [10] which requires 170 lines of code. A non-IT expert is not able to write this and that particular D3 example does not even support menus as does the VizDSL example above.

R3 Support for standards-based interoperability: VizDSL, as an extension of the IFML standard, takes advantage of an existing and widely accepted

[10] http://mbostock.github.io/d3/talk/20111018/tree.html

language for modelling user interaction and navigation. As an extension of IFML, a VizDSL model can be connected to other models describing the structural aspects of systems. For example, a VizDSL model can be connected to a UML class diagram to visualize software system structure.

6.3 Software Design Patterns for Information Visualization

Further, we have chosen to evaluate VizDSL using the software design patterns for information visualization as identified by Heer and Agrawala [5], since this is a comprehensive list of patterns. Heer and Agrawala identified a series of twelve design patterns for the domain of information visualization: *Reference Model*, *Data Column*, *Cascaded Table*, *Relational Graph*, *Proxy Tuple*, *Expression*, *Scheduler*, *Operator*, *Renderer*, *Production Rule*, *Camera*, *Dynamic Query Binding*. In the following section, we discuss how VizDSL applies to a selection of these interaction patterns.

Reference model *Separate data source from visual attributes.*

VizDSL separates the data source from the visual attributes by using the *Data* class to represent the data, the *ViewComponent* subclasses *Element* and *Edge* to represent visual components, the *ViewContainer* class to represent the view and the presentation classes *Layout* and *Theme* for the visual style.

Renderer *Rendering of visual items performed by dedicated, reusable models which map visual attributes into pixels.*

VizDSL uses the *Shape*, *Layout* and *Theme* classes to render attributes.

Operator *Decompose visual data processing into a series of composable operators, enabling flexible and reconfigurable visual mappings.*

VizDSL supports visual data processing using operators by means of the various types of *Layout*, *Interaction*, *Animation* and *Shape* classes. For example, the *Layout* class can be used as a basis for composing a custom hybrid layout using a combination of layout types, element groups and edges.

Dynamic query binding *Allow data selection and filtering criteria to be specified dynamically using direct manipulation interface components.*

VizDSL supports the use of dynamic query binding using callbacks in the *Event* and *Action* classes. As an example, users can select elements from the visualization by clicking on them, which will add the element model to a filter group. This filter group can be used as the focus for the next view when navigating through the visualization.

Operator *Decompose visual data processing into a series of composable operators, enabling flexible and reconfigurable visual mappings.*

VizDSL uses a series of composable builders throughout the visualization pipeline, by the use of classes which are mapped to data structures, visual attributes and control mechanisms such as interactions.

7 Conclusion

In this paper, we have proposed a novel approach for the creation and implementation of interactive visualizations by the development of a visual modelling

language (VizDSL). VizDSL extends IFML using its UML profile which facilitates interoperability between models. VizDSL can be used to design and create highly interactive visualizations which improve understanding of both data content and underlying semantic structures.

Evaluation of the quality of modelling languages presents some challenges in the absence of an established modelling language quality evaluation framework [25]. In [26], an empirical framework to evaluate the usability of modelling tools in terms of *satisfaction*, *efficiency* and *effectiveness* is presented; we will be evaluating usability in this sense by means of user studies with users taken from the OGI Pilot. Future work will also include using the feedback obtained from the usability studies to refine and extend the VizDSL metamodel, with more work on code generation.

References

1. Grossmann, G., Igamberdiev, M., Stumptner, M.: Benefits and Challenges of Multi-Level Modelling for Ecosystem Interoperability. In: Proc. of BDI4E Workshop at I-ESA. (2016)
2. Grossmann, G., Jordan, A., Muruganandha, R., Selway, M., Stumptner, M.: Enabling Information Interoperability through Multi-domain Modeling. In: Proc. of Working Conference on Practice-Driven Research on Enterprise Transformation, (PRET), Springer (2013) 16–33
3. Selway, M., Stumptner, M., Mayer, W., Jordan, A., Grossmann, G., Schrefl, M.: A conceptual framework for large-scale ecosystem interoperability and industrial product lifecycles. Data Knowl. Eng. **109** (2017) 85–111
4. Morgan, R., Grossmann, G., Stumptner, M.: VizDSL: Towards a graphical visualisation language for enterprise systems interoperability. In: Proc. of Symposium on Big Data Visual Analytics (BDVA), IEEE (2017)
5. Heer, J., Agrawala, M.: Software Design Patterns for Information Visualization. IEEE Transactions on Visualization and Computer Graphics **12**(5) (2006) 853–860
6. Brambilla, M., Cabot, J., Wimmer, M.: Model-Driven Software Engineering in Practice : Second Edition. Morgan & Claypool Publishers (2017)
7. Jones, C., Jia, X.: Using a Domain Specific Language for Lightweight Model-Driven Development. In: Proc. of ENASE 2014, CCIS 551, Springer (2015) 46–62
8. Schmidt, D.C.: Guest Editor's Introduction: Model-Driven Engineering. Computer **39** (Feb 2006) 25–31
9. Fill, H.G.: Visualisation for Semantic Information Systems. 1 edn. Gabler Verlag (2009)
10. Howse, J., Stapleton, G., Taylor, K., Chapman, P.: Visualizing ontologies: A case study. In: Proc. of ISWC. Volume LNCS 7031., Springer (2011) 257–272
11. Kocbek, S., Kim, J.D., Perret, J.L., Whetzel, P.L.: Visualizing ontology mappings to help ontology engineers identify relevant ontologies for their reuse. In: Proc. of 4th International Conference on Biomedical Ontology. (2013)
12. Burgstaller, F., Stabauer, M., Morgan, R., Grossmann, G.: Towards customised visualisation of ontologies. In: Proc. of the Australasian Computer Science Week Multiconference (ACSW), ACM Press (2017) 1–10
13. Moody, D.: The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. IEEE Transactions on Software Engineering **35**(6) (2009) 756–779

14. Aranda-Corral, G.A., Borrego-Diaz, J., Chavez-Gonzalez, A.M.: Repairing conceptual relations in ontologies by means of an interactive visual reasoning: Cognitive and design principles. In: Proc. of the 3rd IEEE International Conference on Cognitive Infocommunications (CogInfoCom), IEEE (2012) 739–744
15. Voigt, M., Pietschmann, S., Meißner, K.: Semantic Models for Adaptive Interactive Systems. Human-Computer Interaction (2013) 1–25
16. Nazemi, K., Burkhardt, D., Ginters, E., Kohlhammer, J.: Semantics Visualization - Definition, Approaches and Challenges. In: Procedia Computer Science. Volume 75., Elsevier (2015) 75–83
17. Bull, R.I., Favre, J.M.: Visualization in the Context of Model Driven Engineering. MDDAUI (2005)
18. Bull, R.I., Storey, M.A., Favre, J.M., Litoiu, M.: An Architecture to Support Model Driven Software Visualization. In: Proc. of the 14th IEEE International Conference on Program Comprehension (ICPC), IEEE (2006) 100–106
19. Bull, R.I.: Model Driven Visualization: Towards a Model Driven Engineering Approach for Information Visualization. PhD thesis (2008)
20. Ren, L., Tian, F., Zhang, X., Zhang, L.: DaisyViz: A model-based user interface toolkit for interactive information visualization systems. Visual Languages and Computing **21**(4) (2010) 209–229
21. Weerasiri, D., Barukh, M.C., Benatallah, B., Jian, C.: CloudMap: A Visual Notation for Representing and Managing Cloud Resources. In: Proc. of CAiSE. Volume LNCS 9694., Springer (2016) 427–443
22. Cabanillas, C., Knuplesch, D., Resinas, M., Reichert, M., Mendling, J., Ruiz-Cortés, A.: RALph: A Graphical Notation for Resource Assignments in Business Processes. In: Proc. of CAiSE. Volume LNCS 9097., Springer (2015) 53–68
23. Sendall, S., Kozaczynski, W.: Model Transformation: The Heart and Soul of Model-Driven Software Development. IEEE Softw. **20**(5) (2003) 42–45
24. Brambilla, M., Fraternali, P.: Interaction Flow Modeling Language: Model-Driven UI Engineering of Web and Mobile Apps with IFML. 1 edn. Morgan Kaufmann (2015)
25. Giraldo, F.D., Espana, S., Giraldo, W.J., Pastor, O.: Modelling language quality evaluation in model-driven information systems engineering: A roadmap. In: Proc. of 9th IEEE Conference on Research Challenges in Information Science (RCIS). (2015) 64–69
26. Condori-Fernandez, N., Panach, J.I., Baars, A.I., Vos, T., Pastor, O.: An empirical approach for evaluating the usability of model-driven tools. Science of Computer Programming **78**(11) (2013) 2245 – 2258