Electronic Communications of the EASST
Volume 25 (2010)

Proceedings of the Workshop
Visual Formalisms for Patterns
at VL/HCC 2009

Patterns as Abstractions of Spatial Axes

Jens Gulden

12 pages

Guest Editors: Paolo Bottoni, Esther Guerra, Juan de Lara
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Patterns as Abstractions of Spatial Axes

Jens Gulden

jens.gulden@uni-duisburg-essen.de
University Duisburg-Essen, Germany

Abstract: The decision of how to model patterns as elements of formal systems is a yet sparsely covered research topic. The present article introduces an approach which understands patterns as non-linguistic carriers of formal semantics in models. The notion of patterns is embedded into a theory which links spatial orientation and navigation to the constitution of semantics in human understanding. Inside this framework, the concept of patterns is treated conform to the notion of spatial axes on a shared higher level of abstraction. A formal model is presented which expresses the introduced notion of patterns in a practically applicable meta-modeling language.

Keywords: Model, diagram, visualization, patterns, space

1 Pattern-based semantics in models and scientific theories

In order to communicate and express knowledge, linguistic or non-linguistic devices of expression are used. Meaning – or, as a synonym, semantics –, is not an inherent feature belonging to these means of expression, it does not have the status of an ontological entitity that is attached to things [Put75]. Instead, meaning is a result of interactions carried out by cognitive beings in the world [Noë04]. The "final interpretant" [Pei31] are the actions performed by participants involved in language use.

In order to communicate semantics and constitute understanding, different kinds of means can be consulted. Traditional reflections on the notion of semantics have primarily focused on linguistic symbol systems, i.e., spoken and written languages in a narrow sense. Languages, both formal and natural, are indeed suitable means for communicating and sharing meaning, and a great extent of everyday communication and scientific reflection is carried out successfully in spoken and written language.

However, the phenomenon of understanding is not limited to settings in which language-based communication takes place. It is as well typical for human communication to transfer meaning by non-linguistic carriers, e.g., pictorial symbols, diagrams, gestures, movements or just a combination of colors and locations[1]. Subject-specific communication in different scientific disciplines is also not restricted to linguistic devices of expression. Almost any scientific branch has developed means for expressing subject-related knowledge of the discipline in the form of diagrams, tabular structures, symbolic markings etc. Using non-linguistic carriers of meaning in addition to written and spoken language allows to exploit a much wider range of cognitive

[1] An example of transferring meaning by a combination of colors and locations, together with other contextual aspects, is, of course, the use of traffic lights.

resources to create common understanding among participants in a communication setting and to construct and describe complex systems. While language is bound to a linear structure and words cannot easily be used to point out subtle differences between concepts [Goo68], a wide range of techniques exist to express scientific knowledge in a non-logocentric way, in the first place visualization techniques [Ber84, Tuf83].

A wider notion of semantics incorporates any spatial-temporal constellations and processes in space and time as being capable of constituting meaning. Since the upcoming sections of the present paper focus on scientific means of communication via diagrams, a light will be put on static spatial constellations and non-moving visual appearances as carriers of meaning. The proposed approach, however, could without frictions be extended by including movements and temporal patterns accordingly.

The following Section 2 sketches the current state-of-the-art in semantic theory research. It also lays out why traditional approaches have grown insufficient for methodical reflection on sciences and the languages and models they use to communicate. Section 3 presents new philosophical approaches conquering the desiderata resulting from the traditional view's deficiencies. Based on these works, the requirements for incorporating pattern-based semantics into formal modeling methods in computer science and other disciplines are outlined, and in Section 4, a formal meta-model which fulfills the identified requirements is developed. The meta-model includes the formalization of the term “Pattern” as an abstract kind of spatial axis. The article ends with Section 5 which summarizes the presented work and sketches possible future steps of improvement.

2 In search of a semantic theory able to cope with patterns

Since non-linguistic carriers of meaning become increasingly important when large and complex systems are described, the use of patterns as non-linguistic devices should be reflected when providing scientific support for formal system design. This task, however, has only been sparsely investigated yet. Only since about the 1980s years, philosophical and linguistic underpinnings have been developed to capture a notion of semantics beyond a purely logocentric view. Conjoint research of philosophers and cognitive scientists has resulted in conceptualizations which allow for broadening the notion of semantics as constituted by pattern-like constituents of meaning. The concepts of metaphorical constituents of meaning [JL80] and image schemata [Joh87, Gal05] have provided the methodical tools for acquiring a notion of semantics which can explain phenomena of human understanding and cognition on a level required for coping with patterns.

These approaches are commonly subsumed under the label “embodied cognition” [Wil02]. The main idea of the embodied cognition view is to explain cognition, understanding and the use of symbols as emerging from cognitive processes bodily beings perform when physically interacting in the world. Spatial orientation and navigation of the body are fundamental cognitive operations of all bodily beings which bind cognition to a system of possibilities and limitiations offered by the physical world. According to the embodied cognition view, they form the basis for the development of high-level cognitive operations such as the use of languages and other semantic means of expression.

The theoretical groundings developed by the embodied cognition view provide a thorough foundation for reconstructing means of expression in computer science and information systems science. Capturing knowledge about highly interrelated systems is especially relevant in managing large-scale technological and socio-technical systems, e.g., in the field enterprise modeling [Fra02]. However, the transfer of these results into formal modeling research has yet to be performed. Modeling theories currently applied by computer science's research often merely refer to an objectivist account of semantics, which is embedded into a narrow notion of models inspired by graph theory. This notion cannot be sufficient for being applied in computer sciences and information systems to describe complex technological and socio-technical systems [Gul08], because it is blind towards expressing, e.g., non-linear, interrelated contextual semantics at a glance.

3 Incorporating pattern-based semantics into formal modeling techniques

The efficiency and effectiveness in using models for complex formal system design and management is expected to be raised when modeling languages are opened up to incorporate new means of expressing semantics, e.g., by offering pattern-based visual language elements. A theoretically better supported notion of models and appropriate model visualizations than provided by traditional approaches can be developed on the basis of semantic theories from the embodied cognition approaches, introduced in the previous section.

One place to incorporate a wider notion of semantics is the concrete syntax of a modeling language. A concrete syntax describes the visual appearance of modeling language constructs in a diagrammatic representation. Traditionally, this includes, e.g., the element symbols on a diagram plane, line-styles for connections, or variants of arrow-symbols at connection ends.

Aspects of pattern-based semantics are not taken into account by state-of-the-art concrete syntax descriptions. They get flattened by traditional notation models, which treat positions of graphical elements in a diagram simply as numerical coordinates. From this point of view, a diagram is nothing more than a set of absolutely positioned elements combined with a set of edges that describes relationships between the elements. This view on diagrams originates from traditional graph theory and provides strong restrictions on current conceptualizations of concrete model syntax.

To overcome this limitation, the responsibility for locating elements can be taken over by a more precise semantic description of “what it means to be positioned somewhere”. This does not mean to mix the traditional distinction between semantic language description and purely syntactic graphical notation. It rather means to reclaim elements, which have traditionally been treated as simple syntactic aspects in a notation model, back to a level of semantics.

To achieve this, the idea of what a model notation generally is needs to be reconsidered. This is done in the following by defining a notation meta-model which expresses an alternative idea of concrete syntax notation. The meta-model is completely developed from scratch and uses concepts of structured space and spatial locations as means for visually expressing semantics. It conceptualizes a mapping technique between the semantic concepts of a formal model on the one hand, and its visual appearance on the other hand. Such a mapping associates the semantic content of a model with specific ways of visualization, and separates the notion of knowledge captured by a model from the way a visualization is created to communicate this knowledge and make it cognitively accessible. After performing this methodical division between semantic model and mapping to visualization, it becomes possible to concentrate on generally describing visualization techniques for models in an abstract way, independently from specific concepts reflected by models.

A few approaches for conceptualizing mapping techniques for model-visualization have yet been proposed (e.g., [BEL$^+$07, ESK07]). For practical purposes, the Graphical Modeling Framework (GMF, http://www.eclipse.org/modeling/gmf/) contains a widely used pragmatic mapping approach in form of the `gmfmap` language.

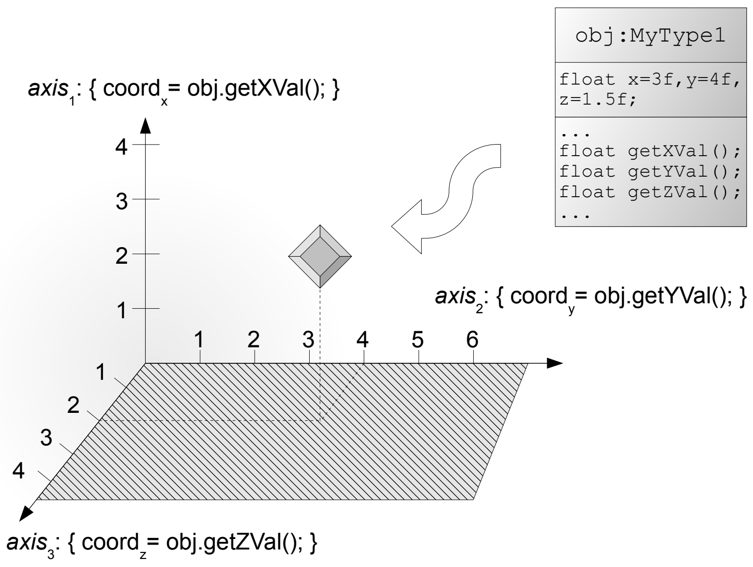
The mapping technique developed here understands describing model-visualizations as constructing abstract spaces into which symbolic objects are placed according to specific matching rules. This way, semantics is expressed by symbols occurring in spatial locations. Each spatial occurrence of a symbolic object which expresses semantics is called an *Allocation* in the proposed approach.

In such a spatial setting, traditional distinctions made on the meta-type level, such as the notion of entities versus relationships, become expressable by multiple spatial alternatives. Entities may straightforwardely be represented by symbols in space, allocated at locations related to axis-intercepts that represent features of the entities. Relationships can be expressed in the same way, if axis-intercepts represent other entities instead of feature values. Nesting view-spaces inside each other (see Sect. 4.7) is another way for representing relationships. As a third option, a relationship may be explicated by placing a symbol via an *Allocation* in space, if the *Allocation* references at least two abstract axes with intercepts refering to entities. It is worth noting that by reconsidering meta-types such as “Entity” or “Relationship” in spatial terms, the dichotomy between both concepts blurs, since both become conceptually exchangeable in the way they are handled and expressed by means of *Allocations*.

The idea of space in the model is kept as abstract as possible and is not restricted to Euclidian, homogeneous, continuous 3D spaces. Instead, spaces are modeled to be spawned by an arbitrary number of *AbstractAxes*. An *AbstractAxis* is any construct that maps the state of a given semantic element (e.g., an attribute value or a combination of values of an object instance) onto *axis-intercepts*, which in turn can be transformed to physical spatial coordinates in a *view-space* (in German: *Anschauungsraum*). A *view-space* is a 1-dimensional to 3-dimensional Euclidian space, addressable by real-number coordinate vectors in well-known Euclidian geometrics.

In the most simple case, a spatial mapping from semantic model elements to Euclidian coordinates is performed by a *NumericAxis* which reads a numeric value from a semantic element's attribute and, without further transformation, directly places this real number into one coordinate component of a view-space's coordinate. Figure 1 gives an example of such a view-space configuration in a notation model. Using 2 or 3 axes of this kind allows for simple, direct spatial visualization of real-numbers provided by the semantic element.

Another simple case are symbolically ordered axes. A *SymbolicAxis* associates string values from a semantic element with discrete axis intercepts in the conceptual space. It then can convert these discrete symbolic values to real-number values by calculating coordinate-positions from axis-intercepts of the symbolic values. An example of using a combination of a *SymbolicAxis* together with *NumericAxes* in a model notation's view-space is shown in Fig. 2. When combined with other types of axes as part of a *CompositeAxis*, the resulting view-space dimension can



obj:MyType1
float x=3f,y=4f,
z=1.5f;
...
float getXVal();
float getYVal();
float getZVal();
...
$axis_1$: { $coord_x$ = obj.getXVal(); }
$axis_2$: { $coord_y$ = obj.getYVal(); }
$axis_3$: { $coord_z$ = obj.getZVal(); }
1
2
3
4
5
6

Figure 1: Example of using *NumericAxes* in a spatial model-notation for direct coordinate mapping

provide additional structure, such as non-equally sized axis-intercepts according to additional cirteria.

More complex situations are possible where one abstract axis influences more than one real-number coordinate component in the view-space, or multiple interdependent abstract axes cooperate to commonly generate one view-space coordinate entry. The number of abstract axes in an abstract space, and the number of Euclidian axes in a view-space, thus are not necessarily equal. A more complex example of using conceptual axes is shown in Fig. 3. Here, one conceptual axis which represents a *CenterPeripheryPattern*, maps onto 2 physical *Dimensions* in the view-space. The *CenterPeripheryPattern* conceptually behaves as any *AbstractAxis*: it maps a semantic element's state to a physical location in the view-space. In the conceptual space, thus patterns and axes are treated equally. They only differ with respect to the way they are later visualized in a view-space. Besides using values from the semantic element to calcluate the view-space coordinate, the *CenterPeripheryPattern* may also apply internal rules for equally distributing symbols inside the "center" area or the "periphery" area. Such an active behaviour of patterns is part of their ability to invoke understanding through spatial constellations.

Figure 4 gives another example of the use of a pattern as *AbstractAxis*. The *GroupPattern* locates semantic element according to a set of distinguishable attribute values, and groups together elements with identical values, while keeping those with distinct values apart from each other.

In order to utilize the results on conceptualizing model notations for novel kinds of modeling languages and tools, the proposed concepts for expressing notations are now to be incorporated into an overall formal notation meta-model. The notation meta-model prepares the development of corresponding software modeling tools for pattern-based model visualizations.

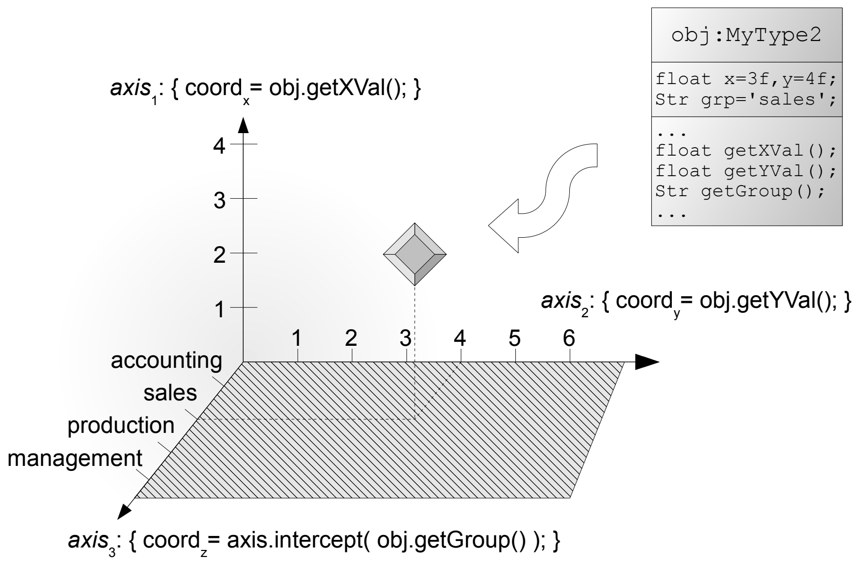


Figure 2: Example of using a *SymbolicAxis*, combined with two *NumericAxes*

4 A formal meta-model for model visualizations based on spatial concepts

Based on the previously elaborated concepts, an overall spatial notation meta-model has been formalized. It consists of three main parts specifying *Views*, *Axes* and *Allocations*, which are described in the following.

4.1 Views

Elements in the first group, *Views*, each specify entire model notations, e.g., to be displayed inside one editor window of a software modeling tool. Each *View* consists of 1 to 3 physical *Dimensions* which are the Euclidian axes of the view-space. Every location of this Euclidian space is described with vectors of real number values as coordinate-vectors. By mapping conceptual abstract axes onto the physical dimensions of the view-space, a structure is induced in the view-space, into which symbols can be placed meaningfully. When the view is rendered, this structure gets filled with *Allocations* derived from the concrete model instance displayed.

4.2 Axes

The declaration of axes happens in an independent *Axes*-section, which is the second main part of the overall *ViewMapping* model. By separating the declaration of axes from the declaration of *AxisMappings*, which are specific to individual *Views*, axes can be reused in multiple *Views* throughout the whole *ViewMapping* model.

Conceptual axes metaphorically spawn a conceptual space inside which knowledge is represented by identifying locations based on semantic features, i.e., by specifying *Allocations*. To

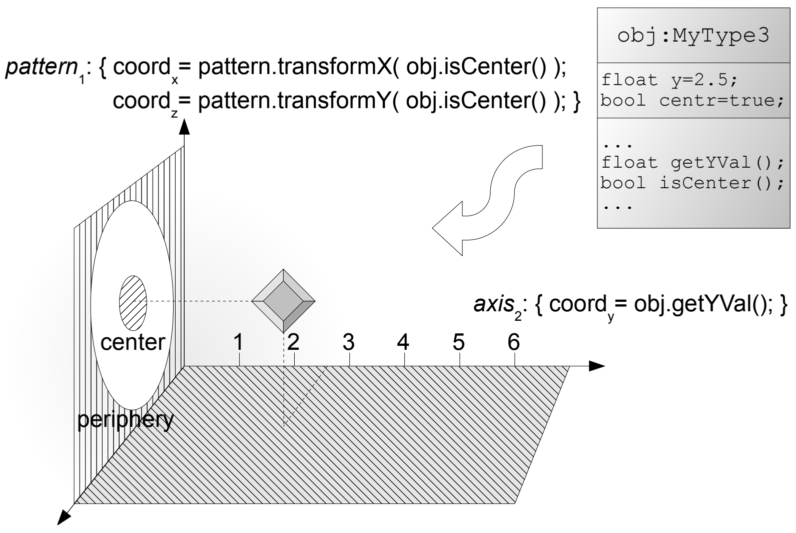

pattern$_1$: { coord$_x$ = pattern.transformX(obj.isCenter());
coord$_z$ = pattern.transformY(obj.isCenter()); }
obj:MyType3
float y=2.5;
bool centr=true;
...
float getYVal();
bool isCenter();
...
axis$_2$: { coord$_y$ = obj.getYVal(); }
center
periphery
1
2
3
4
5
6

Figure 3: Example of using a *CenterPeripheryPattern* as conceptual axis, combined with one *NumericAxis*

display elements from the conceptual space inside a physical view-space, *AxisMappings* are specified. They bind conceptual *AbstractAxes* to physical *Dimensions* in a view-space. It is up to the algorithmic implementation of an *AbstractAxis*' `mapToViewBounds()`-method how to relate to physical *Dimensions* with real value coordinates. The relationship between a logical *AbstractAxis* and *Dimensions* is to be understood as general as possible, which means that one *AbstractAxis* may, if desired, be mapped onto more than one physical *Dimension*. The mapping can also be specified in any possible way. This means, any *AbstractAxis* may in principle modify the real-value components of a view-space coordinate-vector in any aspect, no matter which *Dimensions* are assigned to the *AbstractAxis*.

4.3 Atomic Axes

Multiple subtypes of *AbstractAxis* can be distinguished. An *AbstractAxis* can be an *AtomicAxis*, which represents a semantic element's feature that is to be expressed in spatial terms. When expressing semantics using *Allocations*, symbols are placed at locations relative to intercepts of *AtomicAxes*. This is done through *AllocationMappings*, see below. Any *AtomicAxis* is either a *BasicAxis*, which is an axis that can be mapped to exactly one physical *Dimension*, or a *Pattern*. *Patterns* carry all conceptual features of an *AtomicAxis*, but potentially map onto any arbitrary number of *Dimensions*.

Three concrete realizations of *BasicAxes* are proposed by the model, which are described in the following.

Numeric Axis

A *NumericAxis* maps a numeric feature from a semantic element, or the result of an OCL statement which returns a numeric value, onto a single *Dimension* of a view-space. It can either

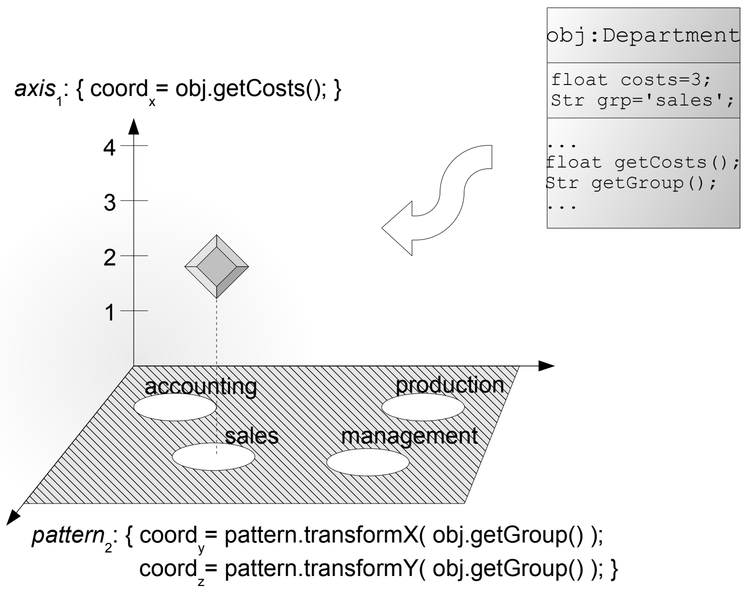

axis1: { coordx = obj.getCosts(); }
4
3
2
1
accounting
production
sales
management
pattern2: { coordy = pattern.transformX(obj.getGroup());
coordz = pattern.transformY(obj.getGroup()); }
obj:Department
float costs=3;
Str grp='sales';
...
float getCosts();
Str getGroup();
...

Figure 4: Example of using a *GroupPattern* as conceptual abstract axis, combined with one *NumericAxis*

be configured to represent a continuous, real-value addressable, homogeneous 1-dimensional number line, or a sequence of distinct intercepts representing integer values. When configured to operate with real-values, the semantic notion of a *NumericAxis* in conceptual space matches directly the notion of a Euclidian physical *Dimension* in the view-space.

Symbolic Axis

SymbolicAxes represent an ordered sequence of distinct intercepts which hold symbolic string values. Coordinate values for each intercept are calculated via the `mapToViewBounds()`-method by multiplying the visual intercept size with the index position of the intercept.

Entity Axis

An *EntityAxis* represents a set of entities with its intercept values. Any set of objects in a model, e.g., any multi-value collection, can be chosen to be lined up as intercepts of an *EntityAxis*. The set of values is specified using the *interceptExpression* attribute. Since the notion of a sequence of intercepts may require to explicate a sorting order of elements, an *EntityAxis* can carry a reference to a *Comparator* class, which may define the sorting order.

4.4 Composite Axes

CompositeAxes act as combinations of other axes. A *CompositeAxis* derives its intercept values, either numerical or symbolical intercepts, from other axes and combines them. Concrete subtypes of *CompositeAxis* describe different combination strategies. Three concrete subtypes are initially suggested by the notation meta-model. This set of combination strategies can be extended, if required.

The first kind of suggested *CompositeAxis* is the *Concat* axis. This combines an arbitrary number of axes sequentially. Another axis composition is *Join*, which resembles the traditional notion of a join in relation theory, i.e., represents a complete one-to-one pairing of each members of the joined axes. A *Mix* composition treats all intercepts of combined axes as one flat set and sorts them sequentially on a combined axis in the order specified by a *Comparator* description.

The combination of axes can be recursively applied, i.e., *CompositeAxes* themselves can again be part of compositions of axes, which makes *CompositeAxes* a highly expressive concept for describing the structure of conceptual spaces for model notations.

4.5 Patterns

Patterns in the proposed notation meta-model behave like axes in the sense that they locate semantic elements in conceptual spaces, and are able to transform these conceptual locations to physical real-number coordinates. Patterns constitute a conceptual space [Gär00] in which meaning is expressed by orientation and navigation in the same sense as axes do. This is achieved because patterns provide meaningful places. Spatial axes do the same, they unfold a space and provide means of orientation and navigation by consisting of intercepts which are associated with meaning through semantic *Allocations*.

In order to demonstrate the potential uses of the *Pattern* concept specified by the model, a set of concrete subclasses of the *Pattern* class is suggested within the notation meta-model. They are briefly explained in the following. This is a heuristic collection of initially chosen patterns and not limited to the proposed examples. An elaboration of the list of patterns may draw upon theoretical work from diagrammatic reasoning research (e.g., [CFO93]).

Group Pattern

One suggested pattern is the *GroupPattern*, which clusters semantic elements according to attribute values. Elements with identical values are grouped together, while keeping a distance between elements with different values. It is up to heuristics of the implementation how to choose coordinates to display grouped symbols in a view-space and how to spread clusters of symbols across the available space.

Spread Pattern

The *SpreadPattern* serves for equally distributing multiple elements in the view-space to make them distinguishable. It does not map a specific attribute of a semantic element to a location in space, but operates on all semantic elements that are mapped via an *AllocationMapping* onto the *SpreadPattern*. Semantically, the *SpreadPattern* allows for expressing a general notion of overview and distinctness, in opposition to notions of detail and a fine-grained perspective.

Center-Periphery Pattern

Sometimes, the notion of a central place with a surrounding area helps to express unbalanced oppositions such as major/minor dichotomies or whole/part relationships. The *CenterPeripheryPattern* serves for placing elements according to a boolean condition into the notion of a "center" place, or allocates them to a "periphery" area. The pattern may typically be mapped onto 2 physical dimensions to allow a circular representation of the center area and a surrounding periphery. However, it can as well be mapped onto one single dimension, where a center part and two surrounding periphery ranges are possible to displaye. The notions of center and periphery can also be expressed in a 3-dimensional sphere-structure.

Sequence Pattern

By applying a *SequencePattern*, semantic elements are aligned into one direction at equidistant positions in the view-space. The typical use-case is to bind this pattern to exactly 1 physical dimension, however, the implementation of the pattern might also offer to visualize sequences in 2 or 3 dimensional space, which would allow for bind the *SequencePattern* to any number of possible view-space dimensions.

Repetition Pattern

A *RepetitionPattern* indicates a cyclic constellation in a model. This semantic device may be realized by diverse graphical representations, e.g., by circular shapes.

Random Pattern

The *RandomPattern* places elements randomly in the view-space. The elements are spread across the subspace spawned by those *Dimensions* onto which the *RandomPattern* is mapped. The number of physical dimensions that can be mapped onto the pattern ranges from 1 to 3, since naturally any number of dimensions can be chosen to be randomly set.

4.6 Allocations

The third main section in the notation meta-model is the *Allocations* branch. *Allocations* map the state of semantic elements to locations in the conceptual space spawned by *AbstractAxes*, and place a symbol or a nested view (see Sect. 4.7) at this location.

By relating the state of semantic elements' features to abstract axis intercepts, *Allocations* describe an abstract notion of locations in space without having to refer to numeric coordinate-vectors. Locations in the conceptual space can be understood as spatial means for expressing semantic elements' states, and thus are not limited to fulfilling structural features such as homogenicity or continuity, which are typically associated with the notion of a three-dimensional Euclidian space.

4.7 Nested Views

Allocations can optionally nest *Views* inside each other. They thus provide a way to recursively apply the concept of mapping *AbstractAxes* to physical *Dimensions* via *AxisMappings* in a nested subspace. If the `innerView` relationship is specified for an *Allocation*-instance in a notation model, the *Allocation* will not directly map semantic elements to graphical symbols at the place described by the *Allocation*, instead, it will place a nested inner *View* at this location.

Figure 5 shows the part of the notation meta-model which defines the discussed axes concepts. The model has been created using the Ecore modeling language, which is part of the Eclipse Modeling Framework (EMF, http://www.eclipse.org/modeling/emf/).

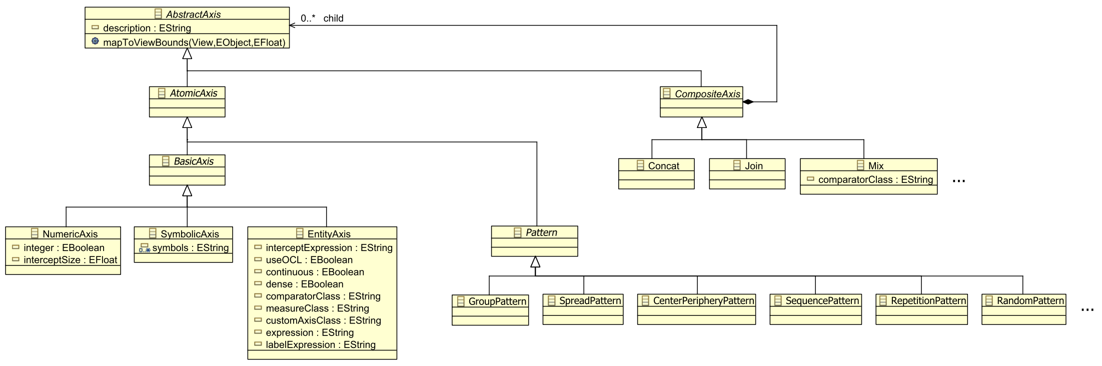


Figure 5: Partial notation meta-model suggested by the presented approach

5 Summary and future perspectives

This article has presented a formal language for describing visual model-notations, which incorporates the notion of patterns as language elements in formal visualization descriptions. This research provides the basis for further elaborating a modeling method which uses spatial means of semantic expression to model complex interdependent systems, as they are being dealt with in computer science and information systems. Besides providing the notation model language and a description of the process to be applied, the method should also be supplemented with prototypical software tool support. An implementation of the presented language, including an editor for notation models and a visualization generator which operates on these models, is planned to be developed on top of the Eclipse Modeling Framework.

References

[BEL+07] S. Buckl, A. Ernst, J. Lankes, F. Matthes, C. M. Schweda, A. Wittenburg. Generating Visualizations of Enterprise Architectures using Model Transformation (extended version). *Enterprise Modelling and Information Systems Architectures* 2(2):3–13, 2007.

[Ber84] J. Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, Madison, 1984.

[CFO93] E. Clementini, P. D. Felice, P. van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *Proceedings of the Third International Symposium on Advances in Spatial Databases*. Pp. 277–295. Springer, London, 1993.

[ESK07] S. Eicker, T. Spies, C. Kahl. Software Visualization in the Context of Service-Oriented Architectures. In *Proceedings of the 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis (Vissoft2007)*. Pp. 108–111. Alberta, Canada, 2007.

[Fra02] U. Frank. Multi-Perspective Enterprise Modeling (MEMO): Conceptual Framework and Modeling Languages. In *Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS-35)*. IEEE Computer Society Press, Honolulu, 2002.

[Gal05] S. Gallagher. *How the Body Shapes the Mind*. Oxford University Press, Oxford, 2005.

[Gär00] P. Gärdenfors. *Conceptual Spaces*. MIT Press, Cambridge, 2000.

[Goo68] N. Goodman. *Languages Of Art. An approach to a Theory of Symbols*. Harvester Press, Sussex, 1968.

[Gul08] J. Gulden. Semantik in visuellen Modellen: Räumliche Regularitäten und körperliche Erfahrungsmuster als Bedeutungsträger visueller Modelle. In Reichle et al. (eds.), *Visuelle Modelle*. Wilhelm Fink Verlag, München, 2008.

[JL80] M. Johnson, G. Lakoff. *Metaphors We Live By*. University of Chicago Press, Chicago, 1980.

[Joh87] M. Johnson. *The Body in the Mind*. University of Chicago Press, Chicago, 1987.

[Noë04] A. Noë. *Action in Perception*. MIT Press, Cambridge, 2004.

[Pei31] C. S. Peirce. *Collected Papers of Charles Sanders Peirce*. Volume 3. Harvard University Press, Cambridge/Mass., 1931.

[Put75] H. Putnam. The meaning of “Meaning”. In Gunderson (ed.), *Language, mind and knowledge*. Pp. 131–193. University of Minnesota Press, Minneapolis, 1975.

[Tuf83] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Connecticut, 1983.

[Wil02] M. Wilson. Six views of embodied cognition. *Psychonomic Bulletin & Review* 9(4):625–636, 2002.