19TH INTERNATIONAL DEPENDENCY AND STRUCTURE MODELING CONFERENCE, DSM 2017
ESPOO, FINLAND, SEPTEMBER 11 – 13, 2017

The Logic of DSM

Patrik Eklund[1], Mats Johansson[1], Jari Kortelainen[2], Vesa Salminen[3]

[1]Umeå University, Department of Computing Science, Sweden
[2]South-Eastern Finland University of Applied Sciences, Finland
[3]Häme University of Applied Sciences, Finland

Abstract: In this paper we propose a logical enrichment DSM so that design structure is described in a logical language rather than as an unstructured relation. We further show how DSM potentially can embrace languages and notations that support modelling of information and process.

Keywords: Category theory, lative logic, modules.

1 Introduction

Design structure described as a matrix (Eppinger and Browning, 2012) oversimplifies the logic of DSM. Components, people and activities in DSM are simply names, so they are logically just constants in the underlying signature of the logic, i.e., there are no operators with arity beyond zero to describe components as terms. This means that task volatility in DSM is connected only with information variability and likelihood, so DSM is not logical but statistical with respect to modeling rework and redesign.

In order to explain the logical extension of DSM, and the use of many-valuedness instead of variability, let X be the set of components. The basic design matrix is a binary relation

$$R \subseteq X \times X$$

which equivalently can be represented as a mapping

$$\rho : X \times X \rightarrow 2$$

where 2 denotes the two-pointed set $\{0,1\}$ $(or\ \{false, true\})$, i.e., representing binary (two-valued) truth. The relation has initially no properties, so it may e.g. be asymmetric indicating that the order between components is important. However, order as a structure is not explicitly recognized within the formal notation, and in fact, DSM comes with very little formal notation.

In design structures, order and many-valuedness are important. In logic it is an interesting question whether order precedes many-valuedness, or vice versa. In (Eklund, Gutiérrez García, Höhle and Kortelainen, 2017) we argue that order underlies many-valuedness. We further show how category theory as a metalanguage, and monoidal closed categories in particular, underlies logical considerations related to order and many-valuedness.

If we first extend 2 to Q, a non-commutative quantale, we have a many-valued relation

$$\rho : X \times X \rightarrow Q$$

and non-commutativity of the quantale means that aggregations will consider the order among elements in Q. DSM also deals with many-valuedness, but in a rather pragmatic way, and not using algebraic notions or logical formalism to describe it more precisely.

This is clearly seen e.g. in DSM's four types of interactions (spatial, energy, information, and materials), with a 5-scale (-2 … 2) characterizing many-valuedness for each interaction. That 5-scale can be viewed as a quantale, but the relation between respective 5-scales is not algebraically explained in DSM.

Many-valuedness and order is thus poorly explained in DSM, and for the set X must also have a more elaborate structure, otherwise the size of that unstructured set quickly grows to become very large, and application development makes no practical sense. As we indicated before, X cannot be just a *set of* elements. It has to be a *structure of* elements. As an example, if we only say 'crankshaft' as a name for a component in an automotive system-of-systems, 'crankshaft' is just a logical constant, but if we include the attributes $attr_1, \ldots, attr_n$ attached to a crankshaft it becomes a logical term. Using logical notation, $crankshaft$ is a logical constant (of zero arity), whereas $crankshaft(attr_1, \ldots, attr_n)$ is a term, with

$$crankshaft: s_1 \times \ldots \times s_n \to s$$

being an operator (of arity n) and s_i, $i = 1, \ldots, n$, and s are types (sorts).

In first order logic, $crankshaft(attr_1, \ldots, attr_n)$ may be viewed as a term or a predicate, but in *lative logic* (Eklund, Höhle and Kortelainen, 2014) we clearly separate terms from sentences, so that $crankshaft(attr_1, \ldots, attr_n)$ is an expression (term) rather than a statement or predicate (sentence). Conglomerates of sentences become part of the logical *theory* related with the design structure.

In the simplest case, components are terms, built upon a signature $\Sigma = (S, \Omega)$, where S is the set of types and Ω is the set of operators. The set of all terms (expressions) is then $T_\Sigma X$, where X is a set of variables. The design structure is then

$$\rho : T_\Sigma X \times T_\Sigma X \to Q$$

where order and many-valuedness reside in both components and the valuation of the relation between them. In this situation, T is a functor over the category of sets, so that order and many-valuedness reside in the functor structure. However, as explained in (Eklund, Galán, Helgesson and Kortelainen, 2014), T can more generally be an endofunctor over any monoidal biclosed category, so that order and uncertainty is modeled in the underlying category (metalanguage) rather than in the functor itself.

Further, the relation ρ may be constrained by properties, such as associativity. Applications typically define these properties, as well as the nature of order and many-valuedness.

We can enrich ρ even further, and this makes us realize how DSM without structure is capable of producing applications on a very general level only.

2 Substitution and value

As a starting point for discussing substitution, note how the mapping $\rho : X \times X \to 2$ can be represented equivalently as

$$\rho : X \to PX$$

where P is the powerset functor, i.e., PX is the set of all subsets of X. For $x \in X$, $\rho(x)$ is intuitively the *value* that is substituted for x. This is more evident in the substitution

$$\rho : X \to T_\Sigma X$$

which, because T_Σ as a functor is extendable to a monad $\boldsymbol{T}_\Sigma = (T_\Sigma, \eta, \mu)$, can be extended to a mapping between expressions (terms)

$$\tilde{\rho} : T_\Sigma X \to T_\Sigma X \, .$$

In a generalized view of components, people and activities, they are all members of term sets with specific underlying signatures, e.g., with Σ_C as the signature for components. We then have $crankshaft$ and $attr_i$ as operators in the signature Σ_C, so that

$$crankshaft\big(attr_1(\bar{x}_1), \dots, attr_n(\bar{x}_n)\big)$$

becomes the value in $T_{\Sigma_C}X$ of the attributed crankshaft as a component.

An expression can also be at a higher level in a system-of-systems, e.g., as for a wheel and its related suspension. In this case, we must provide the design very carefully, since we must consider the order between wheel and suspension. A malfunctioning suspension, as part of a *suspension-wheel subsystem*, will affect the wheel differently as compared to how the wheel, as part of a *wheel-suspension subsystem*, will affect the suspension.

For people, with an underlying signature Σ_P, the term for a human resource may be designed so that the value of the resource in a team is represented as a particular competence or skill. Two coworkers assigned to a task may then be ordered or unordered.

Activities are similar with an underlying signature Σ_A. They precede one another, or execute in parallel, where the overlap may not always be viewed in a binary way.

Components, people and activities can thus be seen to make use of an underlying signature $\Sigma_{C,P,A}$ embracing sorts and operations in respective scopes of information.

Monads can be composed as was shown in (Eklund, Galán, Helgesson and Kortelainen, 2014), where the composed monad $\boldsymbol{Q} \circ \boldsymbol{T}_\Sigma$ provides many-valued sets of terms. Note how it is far from clear if the suspension-wheel subsystem as a term carrying value is more suitable to have a many-valued representation as a term in $\boldsymbol{Q} \circ \boldsymbol{T}_{\Sigma_C}X$, or as a term in $\boldsymbol{T}_{\Sigma_C}X$, where $\boldsymbol{T}_{\Sigma_C}$ would appear as a monad over a monoidal biclosed category, like the Goguen category $Set(Q)$, which includes structure that enable representation of uncertainty. See (Eklund, Galán, Helgesson and Kortelainen, 2014) for detail and examples.

3 Conglomerate and truth

The fundamental difference between expression (term) and statement (sentence) is that the term functor is extendable to a monad so that substitutions can be composed within the Kleisli category of that monad. If a functor composes with the term monad so that the composed functor is not extendable to a monad, then we do not have *terms*, with which we can substitute in expressions, but sentences acting as statements. Here we can then compose further, so that we can distinguish between sentences and conglomerates, or structured sets, of sentences, the latter appearing functorially as “generalized sets” of program clauses in an inference machinery in a particular logic. The value of such sentences or conglomerates relate to qualitative truth rather than quantitative value like in the case of values of terms.

The suspension-wheel subsystem can be designed as a generalized term, whereas an engine system may need a structure as a conglomerate of statements. Note that statements and conglomerates of statements do not hide terms residing within statements. On the contrary, they are available for computation within that system. However, in this case an engine system is observed from qualitative truth point of view whereas a crankshaft is viewed as based on quantitative value considerations. Indeed, it is up to the designer to make these distinctions, and design is a much more complicated process in this logically extended view, as compared to only having matrices as in the basic DSM model.

Teams of people, and organization of teams across management, marketing, operation, subsupply, etc., is requires a similar distinction between value and truth, and an executive may wonder if ROI is quantitative value or qualitative truth.

Activities and tasks are part of processes, and then also to become formalized within process modeling notations. A notation like BPMN (Business Process Modeling Notation) involves bits of syntax, where logical considerations in this paper can be represented, but BPMN also comes short when we aim at using the syntax of our logical machinery at full strength to match the syntax of BPMN.

4 Actions

Previous sections explain how our logical extension of the DSM model of unstructured relations to quite an elaborate distinction of generalized terms and sentences provides a foundation for the *logic of DSM*. This extension is a path from the simplest DSM model

$$R \subseteq X \times X$$

to the radically extended model

$$\boldsymbol{\rho} : Q \circ T_{\Sigma_{C,P,A}} X \to Q \circ T_{\Sigma_{C,P,A}} X \, .$$

Moreover, we are dealing not just with the $X \times X$ matrix as a starting point, but also with $X \times Y$, with

$$\rho : X \times Y \to 2$$

as the functional representation. DSM deals implicitly with relations and interaction *between* elements, respectively, in X and Y. However, DSM does not provide a model about how an element in X might *act upon* an element in Y, and more generally, how the structure of X *affects* the structure of Y. Initially, this can be denoted as

$$\alpha : X \times Y \to Y$$

and the tutorial example is that of a *group action*

$$\alpha : G \times Z \to Z$$

with $(G,*)$ a group and Z a set, and α satisfying

$$\alpha\big(a_1, \alpha(a_2, z)\big) = \alpha(a_1 * a_2, z)$$

Permutation groups are typical examples of this construction. In this case, G has “more structure” than Z, and the action α is capable of providing Z with more structure. This can be generalized in a categorical framework using *modules*. For detail, see (Eklund, Gutiérrez García, Höhle and Kortelainen, 2017).

Actions enable to describe the complicated relation between interventions and evidence that interventions have desired effect. This is important in maintenance and services, and when dealing e.g. with availability and reliability, concepts that also need to be defined in far more detail than just as a percentage of risk or duration of system downtime. In (Eklund, Höhle and Kortelainen, 2017) we provide detail within the realm of health care.

5 Lative logic

Logic is a structure containing

- signatures
- terms
- sentences
- theoremata (conglomerates, or structured sets, of sentences)
- entailments
- algebras (models)
- satisfactions
- axioms
- theories
- proof calculi

and each of these have their respective many-valued extensions as based on underlying order relations. This also enables to model *communication between logics*.

The structure of logic is further *lative*, because signatures are part of terms, terms are part of sentences, and so on. Lativity restricts constructions in logic by disabling self-referentiality, e.g. like in Gödel's famous Incompleteness Theorem using his numbering of sentences. In fact, Gödel's approach is just a sophisticated use of the Liar Paradox, as pointed out by Hilbert and Bernays in the Grundlagen der Mathematik. Logic in design and structure must be lative for the same reason, namely so as not to enable self-referentiality and ad hoc constructions fragmenting the logical description of design structures. Fragmented logic only means we will eventually have fragmented data for analytics, and analytics will produce only fragmented results and observations that cannot be accommodated into a logically harmonized and unified framework. For a treatment of logic in this lative view, see (Helgesson, 2012).

6 Conclusion – The *Information & Process* view of DSM

Development of complex products and large systems is a highly interactive social process

involving hundreds of people designing thousands of interrelated components and making

millions of coupled decisions (Eppinger and Salminen, 2001). On business level, this calls for a framework for synergy management during various types of innovation in life cycle business evolution routing (Salminen, 2009).

In the logically extended view of DSM, design structure becomes potentially supported by *information and process* standards as appearing in the OMG (Object Management Group) family of languages and notations, including

- UML (Unified Modeling Language)
- SysML (Systems Modeling language)
- BPMN (Business Process Modeling Notation)
- CMMN (Case Management Model and Notation)
- DMN (Decision Model and Notation)

UML's Structure Diagram is an information model, whereas the Behaviour Diagram is a process model. In fact, UML's Behaviour Diagram is part of SysML, which is a process model expanding the process model side of UML. SysML is intended e.g. to support systems-of-systems modeling in engineering and manufacturing.

BPMN differs from SysML, and BPMN's syntax enabling data flow with underlying tokens can be provided with a semantic based on our approach to lative logic. BPMN semantics for clinical decision-making has been provided in (Eklund and Helgesson, 2012), where our suggestion for information semantics is that BPMN's Data Object coincides with documents and their structured content. Similarly, 'token' coincides with variable substitution. We are then able to extract at any point in the data flow a valid variable substitution that precisely represents an information snapshot of the process at that particular point. An activity in the BPMN sense can then be viewed as a composition of variable substitutions with the initial token or variable substitution.

On BPMN it should be noted how software implementations often tend to adopt or related to Kanban and/or Scrum styles of business process structures. Kanban is lean, whereas Scrum is agile, but lean is not always agile, similarly as agile is not always lean. Both Kanban and Scrum processes involve pools and swimlanes. Agility underlines sprints and milestones, whereas lean underlines manufacturing efficiency and just-in-time production.

Whereas the underlying logic of BPMN has been explained, the underlying logic of CMMN and DMN is in its infancy. The logic of DMN is basically a propositional logic and on a very intuitive level. Propositional and relational logic for ontology like in web and health ontology also shows how logically poor structures are unable to provide rich information modeling. Logical extension of CMMN and DMN is outside the scope of this paper, but similar techniques and approaches in this paper apply also for these extensions.

In conclusion, it is now clear that DSM can potentially embrace UML, SysML, BPMN, CMMN and DMN, and this is feasible indeed because of the logical extension of DSM.

References

Eklund, P., Helgesson, R., 2012. Information ontology and process structure in social and health care, unpublished, 2012. Appears partly as Chapter 6 in (Helgesson, 2013)

Eklund, P., Höhle, U., Kortelainen, J., 2017. Modules in health classifications, FUZZIEEE 2017.

Eklund, P., Höhle, U., Kortelainen, J., 2014. The fundamentals of lative logic, LINZ2014, 35th Linz Seminar on Fuzzy Set Theory, Linz, Austria, February 18-22, 2014.

Eklund, P., Galán, M.A., Helgesson, R., Kortelainen, J., 2014. Fuzzy terms, Fuzzy Sets and Systems 256, 211-235.

Eklund, P., Gutiérrez García, J., Höhle, U., Kortelainen, J., 2017. Semigroups in complete lattices: Quantales, modules and related topics, book under review.
Eppinger, S.D., Browning. T.R., 2012. Engineering Systems: Design Matrix Methods and Applications, MIT Press.
Eppinger, S.D., Salminen, V.K., 2001. Patterns of product development interactions, International Conference on Engineering Design, ICED ´01. Professional Engineering Publishing, Glasgow, pp. 283-290.
Helgesson, R., 2013. Generalized General Logics, Doctoral Thesis, Umeå University.
Salminen, V.K., 2009. Synergy Management by Hybrid Innovation, Doctoral Thesis, Tampere University of Technology.

Contact: P. Eklund, Umeå University, Department of Computing Science, Umeå, Sweden, +46 70 586 4414, peklund@cs.umu.se