A Design Pattern Metamodel and Use Mechanisms for Systems Engineering

François Pfister, Vincent Chapurlat, Marianne Huchard, Clémentine Nebut, Jean-Luc Wippler

A Design Pattern Metamodel and Use Mechanisms for Systems Engineering

Francois Pfister (francois.pfister@mines-ales.fr), Vincent Chapurlat(vincent.chapurlat@mines-ales.fr), Marianne Huchard (huchard@lirmm.fr), Clémentine Nebut (clementine.nebut@lirmm.fr), Jean-Luc Wippler (jean-luc.wippler@cassidian.com)

In facing repetitive classes of problems during their projects, engineers need to combine practitioners' experience in design with solutions that are already capitalized, approved, and standardized. Sharing, interpreting, and applying this experience and these solutions allows engineers to improve their performance (including comprehensiveness and relevance) and their reliability (since they are using proven solutions that have been justified and argued contextually), while also raising their economic value (through time savings). In this way enterprises can capitalize on their engineers' experience.

The idea of using such "design patterns" is intended to help an engineer improve the nonfunctional features, quality of service, and "ilities" (Manola 1999) of a system under design. A design pattern is a simple and small artifact, linking a model of a problem noticed in a given context with a model of a well-known solution that has been already used to solve the problem in another but quite similar context on which the interest of the solution has been validated. This solution must then be imitated and adapted to another context. This approach is currently used in several engineering fields, such as traditional architecture (Alexander et al. 1977), software engineering (Gamma et al. 1994; Coplien and Schmidt 1995; Harrison 1999; Fowler 1996), and process management (Appleton 1997; Van der Aalst et al. 2003). More recently it has been applied to systems engineering (Barter 1998; Haskins 2003, 2005; Schindel and Rogers 2005; Cloutier 2006; Cloutier and Verma 2007). Despite this literature, however, the design-pattern concept remains poorly formalized for the systems engineering domain. Our research promotes a formalized metamodel for design patterns.

A System Pattern Metamodel

The proposed metamodel is a domain-specific language defined in ecore format shown in figure 1 and built taking into account a global systems engineering metamodel. So, the whole metamodel includes all entities needed to support a systems engineering process, as specified in the ISO 15288 standard.

The main concepts of the metamodel concern system-of-interest models and a catalogue of existing design patterns called system patterns. A system pattern is designed as a parameterized functional microarchitecture; in other words, a function graph in which some elements play given roles (pattern roles) linked by a parameter metaclass to roles (concrete roles) played by elements belonging to the model under study.

A system pattern identifies and describes a solution that addresses a problem in a given context. More precisely, a system pattern is characterized by a unique identifier, a short but evocative name, alternative aliases, a creation date, a textual description, and an author.

In the terms of the system pattern, a Problem describes the specified design problem that is motivating the system pattern. It is characterized by an informal description, a Feature to optimize, a set of competing Forces, and a use-case Model showing an elementary functional or organic architecture. A Force is a competing constraint which, when put in conflict with another constraint, is the cause from which the Problem arises. So the decision to apply a system-pattern depends from arbitration between the Forces. A Force is described by a challenge, a constraint and a Problem Type (which might include Fluid, Field, Structure, Security or others).

A Feature is an extra functional characteristic identified as an "ility". A Solution holds a Pattern Model, which is parameterized system architecture. It configures a design

solution as a response to a Problem considering the given Context. There is only one solution for one pattern, but one Problem may have many solutions through several patterns by using equivalent patterns or related patterns.

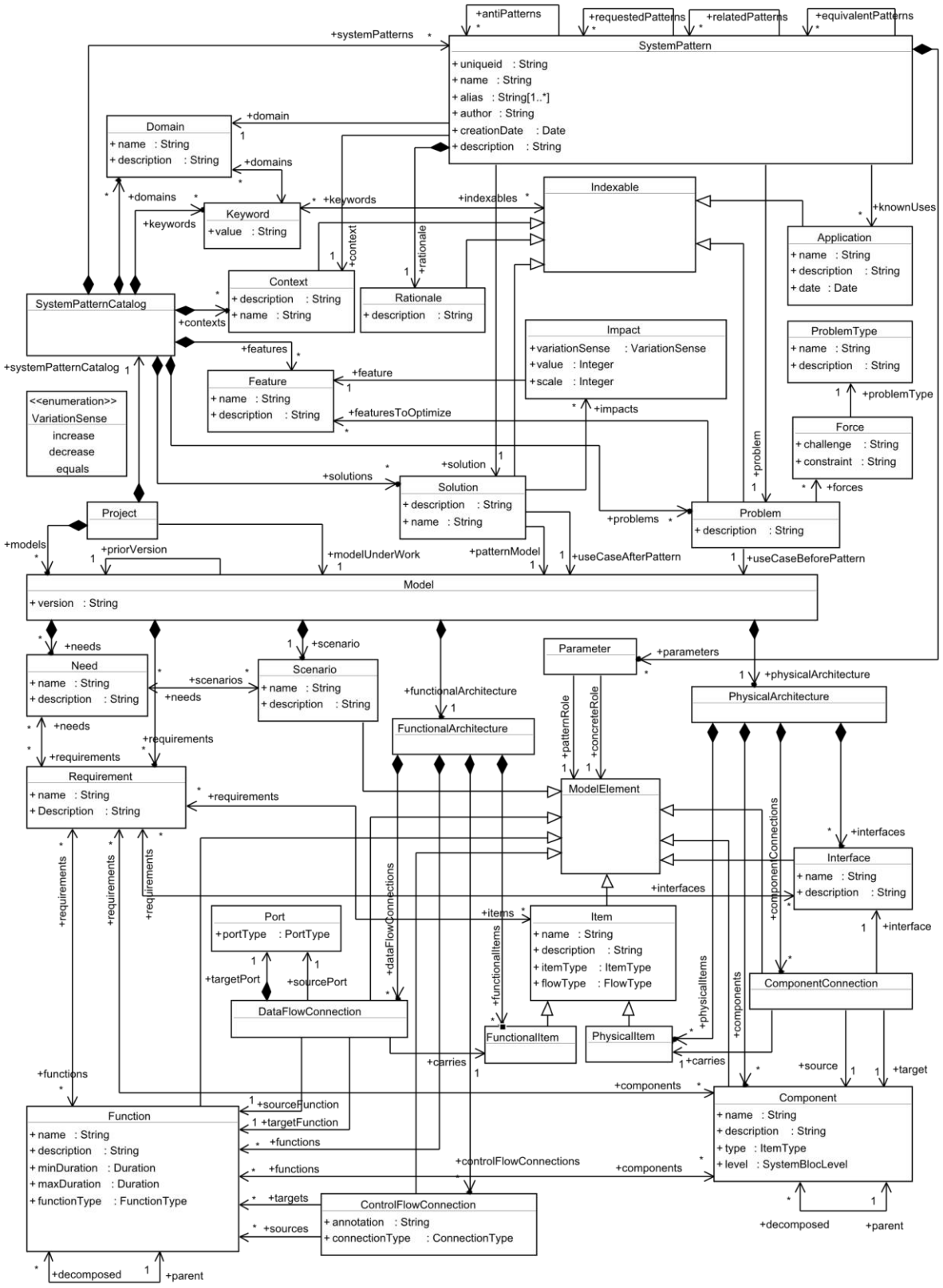

Figure 1. A system-pattern metamodel

A Solution is illustrated by a use case showing a more relevant architecture and an Impact quantified by a Variation Sense (increase, decrease, equals) and a value on a scale. Impact is measured on a feature and allows one to quantify the influence of a system pattern on a system-of-interest model by detecting the optimized and degraded Feature. The Context is interpreted as a set of pre-conditions that define in what cases and in which conditions the System Pattern may be applied.

The Rationale justifies the system pattern by an explicit description and the associated argumentation; thus the rationale allows one to assume the system-pattern can be applied or not. This approach is different from a statistical observation which inventories known-uses in several Applications. Last Problems, Solutions, Contexts, Applications, and Rationales are indexable objects, described by Keywords.

A system pattern has a Domain that identifies a specific area in which a system pattern can be applied or is relevant, such as mechanics, electronics, software, civil engineering, organization and service, security, or pedagogy. System patterns are related to each other by several relationships: requested, related, and equivalent patterns, and antipatterns. Each system pattern references well-known cases (known uses).

Conclusion and Outlook

The proposed metamodel contributes to INCOSE's Model-Based Systems Engineering Initiative (Estefan 2008) by describing the required language that allows one to implement a catalog of systems engineering design patterns. This metamodel is currently under validation. The goal is now to provide mining techniques and models of alignment mechanisms to identify applicable design patterns in a given context. We have designed a software to support this meta model. The editor has to be fully interoperable with the main tools used by system architects. Mining and alignment mechanisms (based on model transformations) are under development.

References

Alexander, C., S. Ishikawa, and M. Silverstein. 1977. *A Pattern Language: Towns, Buildings, Construction.* New York, NY (US): Oxford University Press.

Appleton, B. 1997. "Patterns for Conducting Process Improvement." Paper presented at the fourth annual conference of Pattern Languages of Program Design, Monticello, IL (US), September 3-5, 1997.

Barter, R. H. 1998. "A Systems Engineering Pattern Language." Paper presented at the Eighth Annual International Symposium of INCOSE, Vancouver, BC (Canada), 26–30 July.

Cloutier, R. J. 2006. "Applicability of Patterns to Architecting Complex Systems." PhD diss., Stevens Institute of Technology, Hoboken, NJ (US).

Cloutier, R. J., and D. Verma. 2007. "Applying the Concepts of Patterns to Systems Architecture." *Systems Engineering* 10 (2): 138–154.

Coplien, J. O., and D. C. Schmidt. 1995. *Pattern Languages of Program Design*. Reading, MA (US): Addison-Wesley.

Estefan, J. A. 2008. "Survey of Model-Based Systems Engineering (MBSE) Methodologies." Revision B. Paper for INCOSE MBSE Initiative. Available online at http://www.omgsysml.org/MBSE_Methodology_Survey_RevB.pdf.

Fowler, M. 1996. *Analysis Patterns: Reusable Object Models*. Menlo Park, CA (US): Addison-Wesley.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA (US): Addison-Wesley.

Harrison, N. B. 1999. “The Language of Shepherding: A Pattern Language for Shepherds and Sheep.” Paper presented at the seventh conference of Pattern Languages of Program Design, Monticello, IL (US), August 13-16, 2000.

Haskins, C. 2005. “Application of Patterns and Pattern Languages to Systems Engineering.” Paper presented at the Fifteenth Annual International Symposium of INCOSE, Rochester, NY (US), 10–15 July.

Haskins, C. 2003. “Using Patterns to Share Best Results: A Proposal to Codify the SEBOK.” Paper presented at the Thirteenth Annual International Symposium of INCOSE, Crystal City, VA (US), 29 June–3 July.

Manola, F. 1999. “Providing Systemic Properties (Ilities) and Quality of Service in Component-Based Systems.” Draft. Object Services and Consulting, Inc. http://www.objs.com/aits/9901-iquos.html.

Schindel, W. D., and G. M. Rogers. 2000. “Methodologies and Tools for Continuous Improvement of Systems.” *Journal of Universal Computer Science* 6 (3): 289–323.

Van der Aalst, W. M. P., et al. 2003. “Workflow Patterns.” *Distributed and Parallel Databases* 14 (3): 5–51.