On representing variation

Rich Hilliard
r.hilliard@computer.org

Updated: 18 April 2012

Abstract

Although primarily studied in the context of product lines, variability is a key fact about most systems and therefore a concern for the architectures of those systems. Thus it is essential for the Architect to have suitable tools for representing, managing and reasoning about variation. Most work on product line variation has focused on the variability of components and their connectors within an architecture. Meanwhile, Architects today often use multiple viewpoints to frame diverse stakeholders' concerns for an architecture. *How can variation be expressed within the representational paradigm of multiple viewpoints?* This paper uses a simplified model of variation reflecting current practice and explores the consequences of that model for the representation of variation as a part of architecture description, using the conceptual foundation of ISO/IEC/IEEE 42010 (the revision of IEEE 1471:2000) and poses a number of questions for discussion at the VARI-ARCH workshop.
Keywords: architecture description, concerns, variation, features

I have taken my ECSA 2010 paper for VARI-ARCH and added some of notes in Green.

Working Notes

1. Variation is a concern every architect may face; not just product lines.
2. Any architecture elements can vary; therefore representation of variation should be "orthogonal" to (representation) other concerns, but intimately related.
3. Is the "simple model" in this paper a starting point? What is missing or needs to change? What is needed to make it practical?
4. How do Features relate to Concerns? How do Features relate to AD Elements? Can we enhance a picture like this (figure 1)?
5. If each Variation Point is a place of a possible Decision (or "decision point"?), how can work on Variation utilize existing work on Architecture Decisions and Architecture Knowledge?

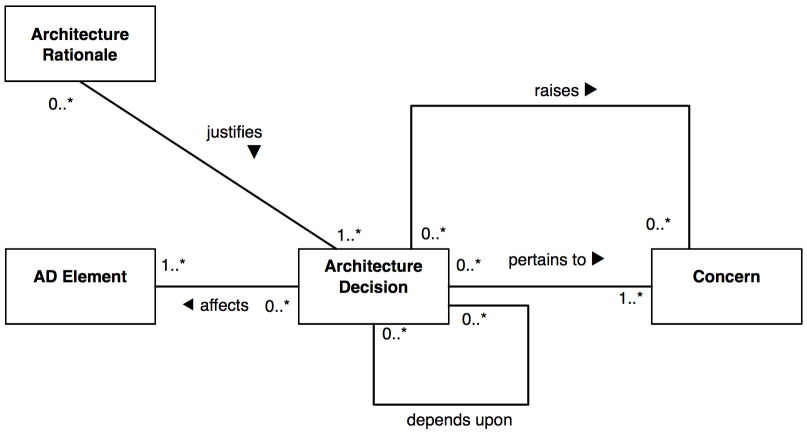

Figure 1: 42010 Model of Decisions

1 Introduction

Variation has been studied for a number of years, largely in the context of product lines and product families, however it is a key fact of *most, if not all*, systems and therefore a relevant concern for the architectures of those systems. In these systems, variation arises for the same reasons it does in product lines; including: to defer decisions about design or implementation; to support multiple deployments; to achieve system qualities such as adaptability.

If variation is widespread, then its explicit representation is probably a Good Thing. As with so many other properties of systems, we observe that identifying and managing variation early in the life cycle of a system-of-interest (whether a single system, product line, product family, system of systems, or other ensemble) is preferable to discovering and attempting to address variation in that system much later in its life cycle [11].

Given its pervasive nature, it seems every Architect should have tools for dealing with variation in her toolkit. Therefore, this paper focuses on the *representation of variation* from an architecture perspective. However, whereas most work on variation within product lines has focused on the variability of components and their connectors within an architecture [1], Architects today often use multiple viewpoints to frame diverse stakeholders' concerns for an architecture. The question I would like to pose for the workshop is:

> *How is variation to be expressed within the representational paradigm of multiple viewpoints?*

In the remainder of this paper, I explore the question of representing variation using as context the conceptual foundations of architecture description presented in ISO/IEC/IEEE 42010 (originally IEEE 1471:2000) [I have updated citations to refer to the published version of the Standard; quoted text and diagrams

have been updated to that version.] with the intent of suggesting a framework for addressing this issue at the workshop.

The next section presents a brief overview of those foundations, section 3 suggests a simplistic model of variation and section 4 is an initial exploration of ways in which variation might be captured within that model of architecture description. The Summary section poses a number of specific questions for the workshop.

2 On Architecture Description

As basis for discussion, I use the conceptual foundations of ISO/IEC/IEEE 42010, *Systems and software engineering — Architecture description* [7]. ISO/IEC/IEEE 42010 is the joint ISO and IEEE revision of IEEE Std 1471™:2000, *Recommended Practice for Architectural Description of Software-intensive Systems* [6].

ISO/IEC/IEEE 42010 attempts to codify best practices in architecture description. It does this by defining a conceptual model (or metamodel) of architecture descriptions: a content model for architecture descriptions (ADs); and, in the newest version, extends these ideas to the specification of architecture frameworks and architecture description languages (ADLs). Most of the observations in section 4 will apply to all three of these constructs.

The conceptual model in the Standard provides a general syntax and semantics of architecture descriptions, which I briefly summarize here.

An *architecture description* (AD) is an artifact specifying a particular architecture of a *system-of-interest* (which may be a software application, an individual system, a product line, system of systems, etc.).

Reflecting contemporary practice, within ISO/IEC/IEEE 42010, an AD is composed of multiple *architecture views*. A view addresses one or more system concerns relevant to some of the system's stakeholders. A view is composed of one or more models. A key requirement in the Standard is that each view in an AD must completely depict the system with respect to those concerns.

Each view is defined following the conventions of an *architecture viewpoint*. The viewpoint determines the notations and *model kinds* used for that view to enable the expression of the architecture in a manner that addresses a set of *concerns* for its audience of stakeholders.

Reflecting the breadth of what the Architect must deal with, concerns, as defined by the Standard, are meant in the widest possible sense:

> **concern** – interest in a system relevant to one or more of its stakeholders
> *NOTE:* A concern pertains to any influence on a system in its environment, including developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological and social influences.

In the latest edition, the basic ideas for specifying an architecture description have been generalized to enable users of the Standard to also specify architecture frameworks [5] and architecture description languages (ADLs) – each being a

mechanism that pairs stakeholders and concerns with viewpoints and model kinds capable of *framing* those concerns.

While the core meanings of these constructs are established by the Standard, there is great flexibility for end users of the Standard (i.e, Architects) in the identification of stakeholders and concerns, and in the formulation of specific viewpoints and model kinds suitable to particular systems, application domains or other considerations – such as how to represent variation! The Standard is defined with the expectation that its users will extend its constructs by defining new viewpoints: each new viewpoint may have its own models, viewpoint languages, notations, as needed to express critical concerns. Therefore, the exact semantics of any particular AD is dominated by semantics of the viewpoints and model kinds used therein.

The core conceptual model is illustrated in figure 2.

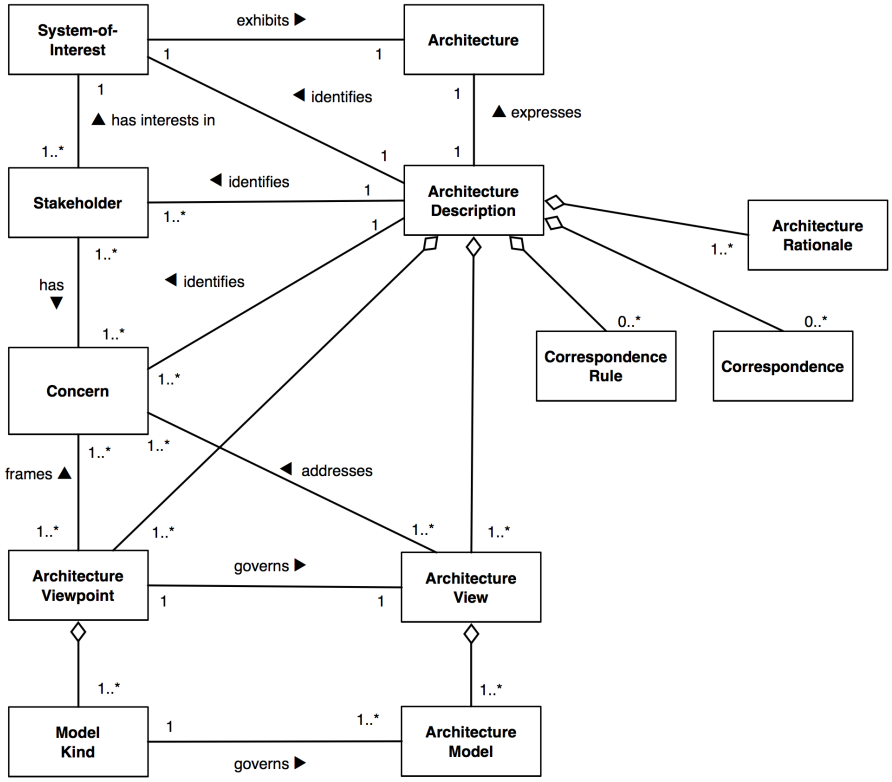

Figure 2: Content model of an architecture description (from ISO/IEC/IEEE 42010)

In addition to the constructs discussed above, ISO/IEC/IEEE 42010 adds a basic notion of *AD element*: any construct occurring in an AD; and a notion of *correspondence* and *correspondence rule* for expressing relations between such elements. Correspondences can be used to capture a wide variety of relations within an AD,

including dependency, traceability, and transformation [4].

3 On Variation

Variation has been typically studied in the context of product lines and product families. In these situations, one wants to capture *commonality*: what is the same across products and *variation*: how and when should the products be different. However, variation is not limited to product line and product family situations; the Architect encounters many situations where variation is present and must be expressed. These situations include variations due to: deferral of design decisions and the resulting choices among one or more alternatives; configuration of "single systems" for customization; multiple deployment, operation and/or maintenance scenarios; planned evolution of a system over its life cycle or even within a mode of operation; self-⋆ systems, etc. Within the product line context, a wide range of such situations have been studied and taxonomized (see for example [3, 9]).

Variation is pervasive throughout the system life cycle, its processes and resulting artifacts. It is in no way the exclusive interest of Architecture. Although this paper is focused on Architecture, specifically on ADs and the linkages to proximate artifacts related to architecture – recognizing key considerations and linkages between AD and other areas, including "upstream" variation requirements which might lead to variations captured in an AD and "downstream" AD variations leading to Design, Implementation, Operations – its observations (if useful at all!) may offer insights outside Architecture.

There are two artifacts frequently used in modeling variability: feature models and variation models.

3.1 Features and feature models

In product line engineering, variation is often characterized in reference to *features*: a feature being any stakeholder-relevant characteristic of a system. Features express commonalities and potential variations. Features and admissible combinations of features are often recorded using feature models [11]. Feature models express these combinations using trees, `AND`/`OR` connectives, mandatory/optional attributes and other constructs.

3.2 Variation points and models

Another common modeling approach uses *variation points*. A *variation point* specifies where a decision is to be made; including the how, when and where a variation may be introduced. A variation may be described by one or more variations points. Associated with the variation point will be one or more *variants*: alternatives which may be realized. Associated with each variant, one might incorporate *conditions*: what facts must hold to use this variant; and a *binding time*: when the variant may be applied [2].

Conditions may be expressed logically, as combinations of choices, values of parameters, in terms of dependencies, etc. The scope or *context* of variation is important – e.g., Bachmann and Clements make the distinction between essential versus local variation. When representing variation, capturing this context is critical to aid realization. A variant must tell us what is varied – let's call this the *variant body*. This may be an attached process description, a model or other artifact, describing the outcome or "recipe" for realizing this variant. Finally, the realization of a variant may introduce *dependencies* on other elements. This should be captured in the variation model, too.

Figure 3 summarizes this simple model – pervasive in the literature.

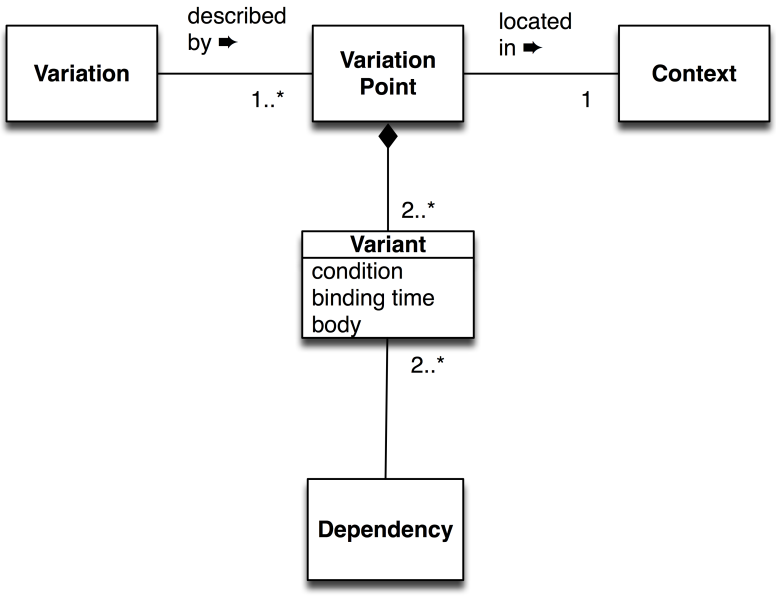

Figure 3: Simplistic model of variation

4 Variation in Architecture Description

This section looks at variation within the context of architecture description, specifically the conceptual framework offerred by ISO/IEC/IEEE 42010.

4.1 Sources of variation

The first thing to observe, is that from an Architect's perspective, *any of the constructs captured in an AD may meaningfully vary for a given system-of-interest*. One can imagine a system with varying stakeholders, varying concerns, and therefore varying viewpoints, views, models and elements.

Stakeholders may vary in kind: change of stakeholder, or in degree: e.g., casual users versus power users. Within an AD, the consequences of stakeholder variation would potentially include varying concerns, viewpoints, and models, as well as their constituents. I am not aware of approaches to variation that address this dimension of stakeholder variation.

Varying concerns is a general expression of a wide range of cases which are beginning to be studied in the product line community, such as variability among qualities, and varying levels of quality [10]. In the Standard, concern is intentionally very general to cover many familiar categories including quality attributes, non-functional requirements, developmental and operational issues, etc. Because concerns are first-class entities in the Standard, their expression is necessary.

One can discern two broad cases among stakeholders: those concerned with variation and those concerned with implementing and managing the outcome of a particular variation. Metzger et al. discuss this distinction in other terms [9]. In the first case, without getting reflective, *variation itself is a concern* for some stakeholders. While in the second case,, stakeholders are interested in implementing, overseeing or using the implementation of particular variants and others will want to reason about the consequences of those selections.

I would observe that the feature models already in use treat variation as a concern, attempting to capture all permissible variation in one place:

> This approach assumes that the feature model plays the exclusive role of gathering and representing all variability aspects of a system (or a family thereof) in one central variability view ([?]). This, however, implies that, since they must potentially accommodate a wide diversity of feature types (such as behavioral, structural or data), feature models necessarily have little to say about the meaning of the variability they represent. [10, Liaskos, et al., 18]

As an Architect, I am agnostic as to whether a feature model is an architecture view, or an input to Architecting, but will make these observations:

- For many stakeholders, a feature model would be more than enough information about an architecture.
- We should ask where the vocabulary used to name and describe features comes from. It is clearly more than just requirements, and equally clearly less than design. Taking the definition of feature seriously: any stakeholder-relevant characteristic may be a feature; therefore, I would posit that *any AD element in any viewpoint language may express a feature*. Of course, some may be much more interesting than others, and many "features" may be aggregations of AD elements. A consequence of this is that the basic partitioning of an AD by its viewpoints lends itself to creating the context for each feature.

In this regard, there is probably useful work to be done among the communities interested in goals, intentions, dimensions of variation, concern spaces and architecture concerns in the sense of the Standard. The strict structuring into trees

often used for goals and some variation models may be too strict for this general case. Experience in Architecting suggests the relations are more complex and do not typically have a single "root" due to multiple stakeholders, conflicts, etc.

4.2 Expressing Variation

Given the simplistic model of variation shown above, we can now look at possible kinds of variation.

The first thing to notice is that if we capture a variation as in figure 3 then potentially anything in an AD might be in the vocabularies for a Condition, or might occur in a Body; i.e., anything in the AD could be a source of variation or a target of variation.

The Binding time in an AD may be within the AD, i.e., the variant expresses changes to be realized within the architecture; or may be realized downstream: during Design, Implementation, Deployment, Installation, Operation (or beyond: one can imagine system retirement options, especially for critical systems). Often, the binding time will be obvious from the variation point's context, which will be expressed by the model or view to which it refers (e.g., an operational view is unlikely to contain a design-time variation point).

It is critical to capture the dependencies among variants. Within the context of architecture description, dependencies would (hopefully) already be part of each viewpoint language, and should therefore be available for expression of variant dependencies. For cross-model and cross-view dependencies, correspondences can be used.

Finally, wherever there is no variation (point) in an AD, the reader can assume there is commonality!

4.3 Implementing Variation

Given the scheme for architecture description in ISO/IEC/IEEE 42010, there are several possible approaches to implementing variation constructs within the the Standard:

1. a *variation viewpoint*, for defining a single view within an AD where all variation-related information resides;

2. a *variation model kind*, which can shared over multiple viewpoints; packaging all variation points relevant to each view into a model within that view;

3. a global, viewpoint-neutral *variation notation or annotation scheme*, (building on the body of existing knowledge of AND/OR trees, mandatory/optional features) which may be utilized wherever needed within each/any view/model;

4. Unique, idiosyncratic approaches to variation for each viewpoint or model kind where variation may arise.

At this point in time, I see no basis on choosing a single implementation approach, and the needs of various Architects and systems may lead to preferring one or another in individual cases. Analogs to each can be found in the existing literature. For example:

Thiel and Hein use a feature model and a variation point viewpoint (although they do not call it that) with one or more architecture views in the spirit of IEEE 1471 to express variation [11].

KumbangSec defines four kinds of variability, each associated with a defined viewpoint: functional, security, component and connector, and deployment variability [10, Myllärniemi, et al., 61]

More than one researcher has observed the similarity, synergy of variations and aspects [10, Kim, et al., 37ff]. In ISO/IEC/IEEE 42010, model kinds can be used for aspect-style Architecting, and for "packaging" portions of an architecture description in general.

Mens et al. show how change can be represented in a first-class manner in terms of resemblances and replacements [8]. Their language could be a candidate of the third type, a schme to be used across viewpoints and model kinds.

5 Summary

Variation is pervasive in Architecting – not just for product lines and product families. Therefore, it is essential for the Architect to have tools for handling variation in her toolkit. Common approaches to variation in the software product line literature, features models and variation points, have been "contextualized" to fit with the practices of contemporary architecture description, using the paradigm of multiple viewpoints together with hints at how they may be generalized are suggested. But the main goal of this paper is to suggest a background to explore the representation of variation within this paradigm. The following questions are posed in the context of the above presentation for discussion at the workshop:

1. Is the simple model of capturing variation described in 3 adequate in the multiple viewpoints paradigm?

2. Are there alternate models of variation suggested by this model of architecture description?

3. Which "bindings" of constructs of variation elements in Figure 2 to AD constructs make sense? I.e., practically speaking, which AD constructs can usefully vary and therefore ought to be representable?

4. Can/should variation be limited to "structural" models (modules, components or connectors) or is the notion equally useful for arbitrary architecture views?

5. Can/should feature models be generalized to multiple views?

References

[1] F. Bachmann and L. Bass. Managing variability in software architectures. *SIGSOFT Software Engineering Notes*, 26(3):126–132, 2001.

[2] F. Bachmann and P. C. Clements. Variability in software product lines. Technical Report CMU/SEI-2005-TR-012, Software Engineering Institute, September 2005.

[3] J. Bosch, G. Florijn, D. Greefhorst, J. Kuusela, J. H. Obbink, and K. Pohl. Variability issues in software product lines. In *PFE '01: Revised Papers from the 4th International Workshop on Software Product-Family Engineering*, pages 13–21, London, UK, 2002. Springer-Verlag.

[4] N. Boucké. *Composition and relations of architectural models supported by an architectural description language*. PhD thesis, Katholieke Universiteit Leuven, October 2009.

[5] D. Emery and R. Hilliard. Every architecture description needs a framework: Expressing architecture frameworks using ISO/IEC 42010. In R. Kazman, F. Oquendo, E. Poort, and J. Stafford, editors, *Proceedings of the 2009 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA/ECSA 2009)*, pages 31–40. IEEE Computer Society Press, 2009.

[6] *IEEE Std 1471–2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, October 2000.

[7] *ISO/IEC/IEEE 42010, Systems and software engineering — Architecture description*, December 2011.

[8] T. Mens, J. Magee, and B. Rumpe. Evolving software architecture descriptions of critical systems. *Computer*, 43(5):42–48, 2010.

[9] A. Metzger, K. Pohl, P. Heymans, P.-Y. Schobbens, and G. Saval. Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis. *Requirements Engineering, IEEE International Conference on*, 0:243–253, 2007.

[10] K. Pohl, P. Heymans, K.-C. Kang, and A. Metzger, editors. *Proceedings of the First International Workshop on Variability Modelling of Software-intensive Systems*, 2007. Available as Lero Technical Report 2007–01.

[11] S. Thiel and A. Hein. Modeling and using product line variability in automotive systems. *IEEE Software*, 19(4):66–72, July/August 2002.