RAIRO-Inf. Theor. Appl. **39** (2005) 361-390
DOI: 10.1051/ita:2005023

A METRIC FOR EVALUATING SOFTWARE ARCHITECTURE AND COMMUNICATION MODELS CONSISTENCY

Jean-Yves Lafaye[1] and Georges Louis[1]

Abstract. Among several alternative viewpoints for building software quality metrics, evaluating the consistency between different models in a software specification or implementation appears to be fruitful. An obvious difficulty is that different models are usually expressed by means of different concepts, and then, confronting heterogeneous representations is not straightforward. In this paper, we propose a solution for measuring the consistency between the architecture and the communication models. After some sensible transformations, the information about both models are captured trough hierarchical representations. We define and discuss a similarity measure between hierarchies, that eventually founds the software metric we propose. Lastly, we investigate how to scale and interpret the metric values and give an application example with SDL.

Mathematics Subject Classification. 68N30.

1. Introduction

There actually is a large collection of software metrics [5–7, 21], that tend to capture various features of quality as defined and classified by several authors [8, 9, 16, 26]. Object oriented programming is a strong evolution within computer science and even if the fundamental principles about software quality still remain unchanged, metric definitions have had to be revised and extended: some original propositions, specific to object oriented specification and design, have recently been made, while former ones have been reviewed [3, 15, 22, 23].

Keywords and phrases. Software metrics, software architecture model, communication model, hierarchical clustering, model consistency, SDL.

[1] Laboratoire Informatique, Image, Interaction (L3i), Université de La Rochelle, France; jylafaye@univ-lr.fr; glouis@univ-lr.fr

In this paper, we focus on consistency concerns. More precisely, a specification or an implementation, more or less explicitly embodies various models (*e.g.*: data and software architecture, communication or behaviour...). In fact, all of these models have to be consistent with one another, and their consistency should be evaluated and controlled. Added to plain syntactic and technical rules, there should exist a semantic consistency that ensures all models are compatible and express one underlying general knowledge which, in fact, is the backbone of the application.

Syntactic correctness is taken into account by all CASE tools, and syntactic consistency between models is addressed by the main ones (*e.g.*: objecteering, Rational, SPIN for UML, and TAU and ObjectGeode for SDL).

Conversely, there are very few works dedicated to semantic consistency concerns. However, "consistency" remains a key word in software quality matter. It might refer to consistency between views in distributed software development, which has been addressed for long in a large number of papers [11, 13]. In all these papers, consistency is the basis for reasoning and formal proof using model checking and various kinds of logic. The original information is captured from pieces of a specification which can originally be written in a general specification language, and is then translated into a suitable formalism before being processed with *ad hoc* tools.

It might also refer to the wide scope of database systems design, where data consistency is a major question, dealt for instance with static architecture issues, constraint checking and dynamic trigger specification.

More specifically connected with our approach, consistency between some UML models is addressed in [14]. The study is limited to the confrontation between UML StateTransition Diagrams and UML Scenarios (or Object Collaboration Diagrams). Once again, model checking and Temporal Logic are used to ensure consistency.

Whatever the context, consistency is mainly tackled as a matter of checking invariant and pre-post conditions. The solutions are found in organisation (project management, quality rules, coordination, use of CASE tools, controlled distributed software design process...) or in a technical approach (State Transition Systems, logical reasoning, model checking, graph theory, CASE tools again...).

Our point of view is different and complementary, and seems not to have been yet studied much in the literature. We assume that the specification under analysis is correct, and we are digging for awkward and inappropriate software architectures. Our aim is to point out such inadequacies – despite of asserted technical correctness – between two specification models. The goal is to increase the quality level of such important external attributes as efficiency, readability, maintainability, testability... So, it appears that our proposal should be applied from the first steps of the specification. An improved architecture (according to our consistency approach) – being easier to check – should be a better input for the general verification methods mentioned above.

One might try to imagine metrics for evaluating the consistency of a whole specification, but we shall – for now – only deal with pair-wise comparisons of models.

In this paper we only address the consistency between the software architecture model and the communication model. So, according to Fenton's classification [10], the quality measures we propose can be considered as based on internal product attributes, since architecture and communication models intimately constitute the code under evaluation. Our proposition applies either to a specification or a program, as soon as communicating modules can be identified and structured in both a hierarchical way (with respect to architecture) and a net structure (with respect to communication).

This is the case for SDL specifications [8] where a set of processes and procedures is structured according to a hierarchical organisation, namely in systems, subsystems, blocks, and nested sub-blocks. A set of channels and signal routes convey signals between processes and constitute the communication relationship.

The last section of this paper presents an example where the system specification uses SDL. Our opinion is that – at the expense of some additional reflection and work for adaptation – this could also be applied to UML specifications, where the hierarchy of packages (Class-diagram) and the communication between objects (collaboration-diagrams / sequence diagrams) are analysed. Similarly, other examples could be found in modular, object oriented or reactive programs that involve communicating objects and a hierarchical architecture.

With such an approach, the main question is to find one operational common representation that could apply to the two models under study. In Sections 2 and 3, we address this question and take advantage of graph theory [25] to advocate for using a hierarchy of processes as a super-representation that can account for both the genuine hierarchical process architecture and the net structure of the communication relationship. Section 2 is dedicated to the architecture model, which is rather straightforward, and Section 3 to the communication model which is not.

We published a previous paper on the subject, that was based on rather similar premises, but dealt with partitions of the set of processes instead of hierarchies [1]. So, the two approaches are complementary, the former is local (only one level in the implicit hierarchy is taken into account) while the latter is global (the whole hierarchy is explicitly considered).

In order to compare two hierarchies – the one for architecture an the other for communication – we discuss, in Section 4, some similarity measures definitions that are convenient for dealing with the semantics of each model. The basic ideas for constructing these similarity measures are stemming from data analysis techniques: clustering, relational and combinatorial analysis.

Of course, these rough software metrics have to be normalised and scaled in order to allow efficient interpretations and bring significant meanings [9, 10]. Section 5 specifies mathematical transformations that finally provide such an operational software metric, *i.e.* ranging within $[0, 1]$, and being free from size effects.

Before a conclusion in Section 7, Section 6 presents with a short application example using SDL.

Some important theorems that found our reasoning – especially about Co-Graphs [25] and the underlying Euclidean distance upon vertices – are cited

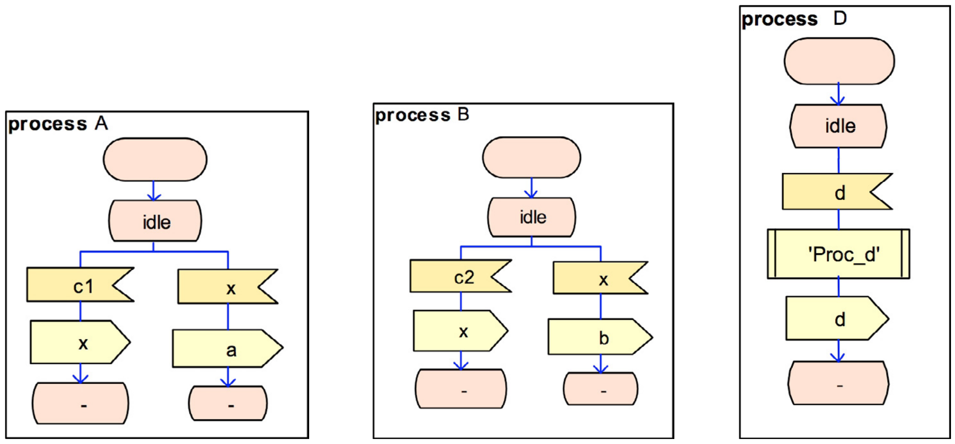


FIGURE 1. A first example: processes A, B, and D.

with references to our previous paper [1]. They could not be detailed here for sake of conciseness, so these results must be admitted or otherwise read in the references. In contrast, properties that are specific to the present approach are exposed.

Before going into the heart of the matter, it might not be pointless to sketch a toy example in order to stimulate intuition about our final aim, our inputs and outcome. The toy example below is taken from an SDL specification of a most simple system with two blocks and five processes.

We first provide a rough SDL specification of processes that deliberately hides the code of tasks, and focuses on signal exchange. Of course, code analysis will later be essential for capturing the communication pattern when dealing with real systems, but it is not in case of a toy example, where channels and routes analysis is sufficient.

Then we specify two architecture models that may implement the common process communication system. The whole specification is syntactically correct, but one is evaluated as better than the other by our quality metric (architecture and communication models consistency). We insist on the fact that the best of the alternative architectures is not necessarily the one that will eventually be implemented. In fact, in the paper we shall extensively detail in what sense and on what criteria our metric is based; other criteria can quite reasonably be put forth (for instance hardware constraints...), anyway, in case a “better” architecture is rejected in favour of an other, the choice of the latter should – in our opinion – be explicitly advocated for.

The diagrams in Figures 1 and 2 represent the specification of the five processes, namely: A, B, C, D and E. We propose two candidate architectures, namely System_1 and System_2, to support communication and achieve process distribution. Both are constituted of the same number of blocks, but this is not a constraint of our approach.

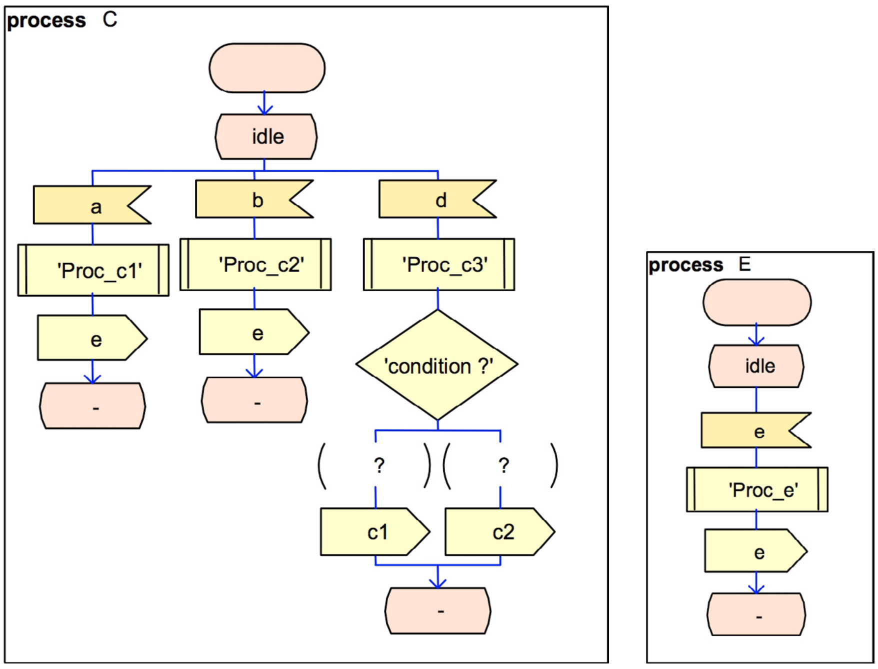


FIGURE 2. A first example: processes C and E.

The diagrams in Figure 3 specify the System_1 architecture. The architecture model for System_1 directly comes from the SDL system and blocs structure, and can be represented by a tree or a list as:

```
System_1 (
   Block_ACD (Process_A, Process_C ,Process_D),
   Block_BE (Process_B, Process_E)
)
```

Computing the quality metric for System_1 leads to a value of 0.25, which must be read as the occurrence of a random variable uniformly distributed in $[0, 1]$. This proves to be a poor quality, showing a high discrepancy between System_1 architecture and the communication model between processes.

The diagrams in Figure 4 now present with the System_2 architecture which is a far better specification according to the consistency criterion. System_2 architecture hierarchical model is:

```
System_2 (
   Block_AB(Process_A, Process_B),
   Block_CDE (Process_C, Process_D, Process_E)
)
```

The quality metric value now is 0.85. This proves to be satisfactory. As a matter of fact, 0.85 actually is the best possible value for a two blocks architecture.

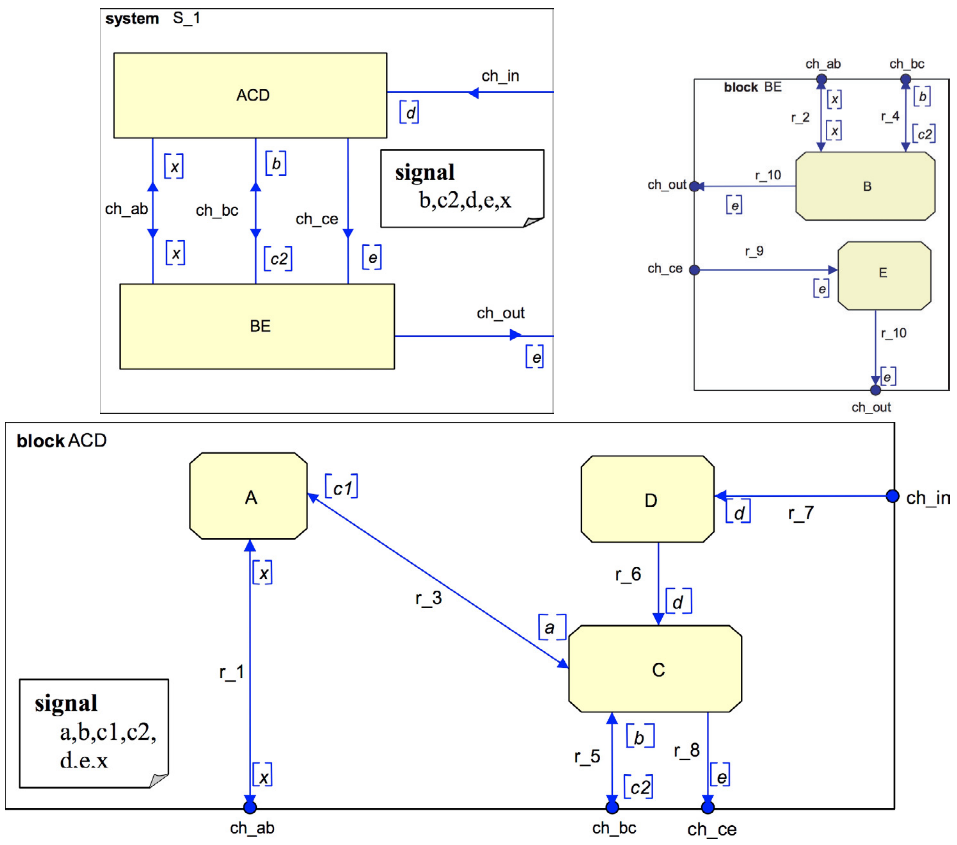


FIGURE 3. A first example: the System_1 architecture.

Better quality values might be reached by specifying three blocks. The "ideal" architecture, which is the best hierarchy of process according to our criterion (the one closest to the communication graph) would be System_*.

```
System_*(
   Block_ABDE(
      Block_AB(Process_1, Process_B),
      Block_DE(Process_D, Process_E)
   ),
   Block_C(Process_C)
)
```

It is clear that the various concepts and properties that have just been cited, need proper and rigorous definitions, proofs and discussion. This is the subject of the reminder of the paper.

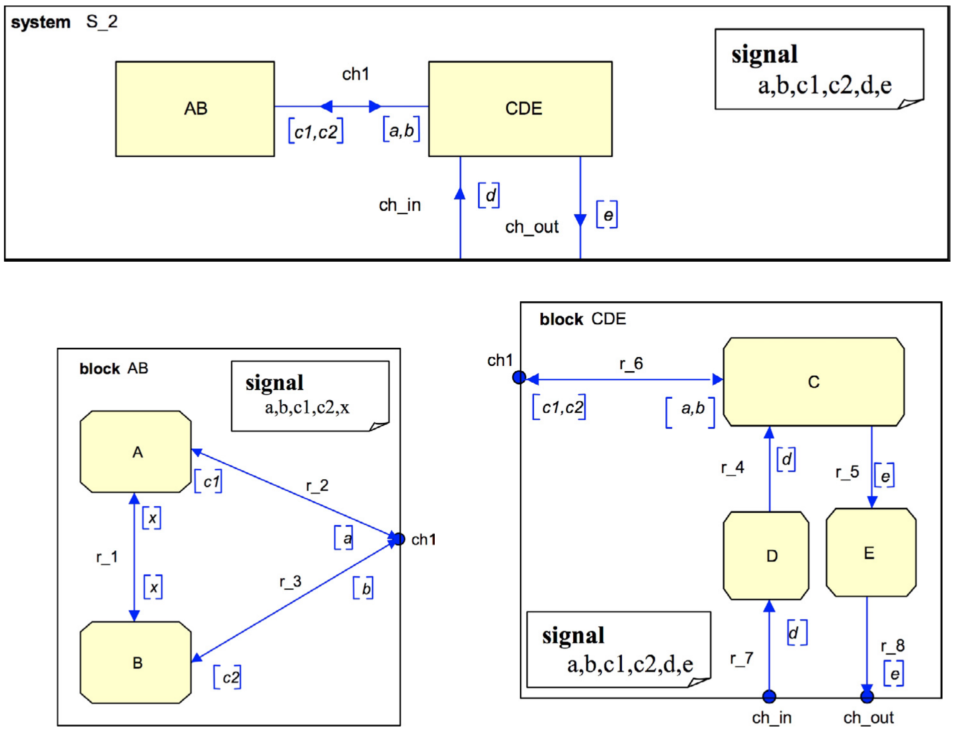


FIGURE 4. A first example: the System_2 architecture.

2. Architecture model representation

We assume that we deal with programs or specifications that are constituted by modules being clearly identified. These modules must be liable to be organised in a hierarchical way. In most cases, the architecture model is explicitly defined; otherwise, it can generally be captured from the analysis of the source code or specification. For instance, modules may be such entities as packages, classes and sub-classes, or processes and procedures, or linear pieces of code and instructions... The overall architecture is derived from the nesting relationship. When necessary, in case the resulting hierarchy only is partial, a dummy root-module can be added so as to obtain a total hierarchy.

A constraint is that no overlapping clusters are allowed, in order for the resulting structure to actually be a hierarchy.

This does not mean that no other architectural organisation than a hierarchy (*e.g.*: a net) can be found between modules (for instance, mutual visibility rules could induce another relationship than the hierarchical embedding relationship between packages in UML design). We simply assume here, that only a hierarchical architectural relationship is retained. The reasons for such a hierarchical structure upon modules, may be either a conceptual decomposition, which means that modules are semantically coherent entities (with main effects on readability, maintainability or testability...), or at an implementation level, a physical distribution of modules (with main effects on costs, transmission delays or reliability...). In both cases, complexity and efficiency are directly involved.

The hierarchy can be described in terms of the set of its leaves, and the set of its clusters, accompanied by the various levels where the clusters are constituted. Different classes may be constituted at a same level.

From now on, we shall refer to lower level modules as leaves, and to upper level modules as nodes.

We denote by P the set of leaves (P might be for "process") and assume that n is the number of leaves ($\#P = n$). Let HP be the function that associates each level (indexed by a natural number) in the hierarchy to its corresponding partition.

Let's define the following, with use of the Z Notation [24] *(with a paraphrase in natural language).*
(POW is for PowerSet)

$$\text{Partition}(X) == \{Y : \text{POW}(\text{POW}(X)) | (\forall A, B : Y \bullet A \cap B = \{\}) \wedge \bigcup\{C : Y\} = X\}.$$

(An element in partition(X) consists of a set of subsets of X that do not mutually intersect and whose union is X.)
A hierarchy on P can be defined by means of a total function HP which fulfills the following constraints. It is assumed there are h levels:

$$h : 1...(n-1)$$
$$\text{HP} : 1...h \rightarrow \text{Partition}(P) \qquad (\rightarrow \textit{ denotes a total function})$$

(each of the h levels in the hierarchy is associated – through HP – with one partition of the set of leaves)
that make a hierarchy:

$$\forall i, j : 1...h \mid i < j \bullet (\forall A : \text{HP}(i); B : \text{HP}(j) \bullet A \subseteq B \vee A \cap B = \{\})$$

(any two clusters from distinct levels either are embedded or do not intersect)
with a unique root:

$$\text{HP}(h) = \{P\}\cdot$$

Another (partial) function HC (which is a kind of HP inverse) associates each cluster in the hierarchy to the level where it actually is constituted:

$$\text{HC} : \text{POW}(P) \nrightarrow 1...h \qquad (\nrightarrow \textit{ denotes a partial function})$$

(some of the partitions of the set of leaves correspond – through HC – to one level in the hierarchy).

with the defining properties:

$$\begin{aligned}&\forall C : \mathrm{POW}(P); c : 1...h \bullet \\ &\qquad \mathrm{HC}(C) = c \iff (\forall D : \mathrm{POW}(P); d : 1...h \mid D \in \mathrm{HP}(d) \land C \subseteq D \bullet d \geq c)\end{aligned}$$

(given cluster C, being constituted at level c in the hierarchy, any other cluster D that contains C is constituted at level d higher than c)
and

$$\mathrm{dom}(\mathrm{HC}) = \bigcup\{C : \mathrm{POW}(P) \mid \exists k : 1...h \bullet C \in \mathrm{HP}(k)\}$$

(the domain of HC is made of those clusters that appear in one partition at some level k in the hierarchy).

3. Communication model representation

We consider the communication relationship between modules. The communication model must deal with those modules that appear in the architecture model. Communication corresponds to the occurrence of an event, and according to the kind of analysis being done, communication may refer to such situations as signal input/output, method invocation, procedure call or global variable sharing...

Our viewpoint is static, that is to say the communication relationship only indicates potential communication. In fact, a static analysis of code does not allow to capture the actual dynamic communication pattern. Then, we assume that two modules "communicate" as soon as they can exchange data. Such a point of view naturally leads to a symmetrical communication relationship (some developments can be added to our proposal in order to cope with unsymmetrical communication relationship, but this will not be addressed here). We also admit some non determinism, which exists in various languages: when a sender module does not precisely identify the receiver, then all possible candidates are supposed to "communicate" and hence are in relation with the sender. The object of our analysis is then an upper estimation of the actual communication model, in that the actual communication relationship is included in our communication model.

Of course, our proposal remains valid for any more accurate communication relationship, the underlying question being how and at what price the capture of this more precise information could be achieved.

3.1. Graph representation of the communication model

According to our previous definition, the communication relationship is represented by a graph $G(P, E)$. P is the set of leaves (modules) in the architecture hierarchy. E is the set of edges ($E : \mathrm{POW}(P \times P)$). An edge exists between two modules in P iff these two modules can communicate.

As said above, the communication is assumed to be symmetrical, hence G is symmetrical too. We consider that reflexive edges do not make sense for our study, then information about $\{x : P \bullet G(x, x)\}$ is considered as being irrelevant.

The $n \times n$ incidence matrix of Graph G, will also be denoted by "**G**" (boldface to avoid misinterpretation).

3.2. Hierarchical representation of the communication model

In order to allow a direct comparison between the architecture and the communication model, we intend to represent Graph G by a tree. We shall first recall some results about Co-Graphs which constitute a maximal class of graphs that can exactly be represented by trees in a simple manner [25]. Then, we present what we call the Canonical Decomposition of a graph. The Exact Canonical Decomposition (ECD) is a direct extension of the Co-Graph notion and does not provide a tree representation, but the Approximated Canonical Tree Decomposition (ACTD) does. Defining the ACTD implies a previous reflection that results in providing the set of leaves with a Euclidean structure.

We list below the main properties about Co-Graphs, ECD and ACTD. No theoretical developments nor proofs are given, but illustrative examples are. See the references for more information and proofs [1, 25].

3.2.1. *Co-Graphs*

The question of computing a maximal set of graphs that admit an accurate tree representation has been studied during the 70's. The study only considers trees that can be built through algorithms with a polynomial complexity. Three main families of solutions are known, namely Co-Graphs [25], TSP Graphs [18] and Interval Graphs [4]. When graphs are symmetrical (which is the case), Co-Graphs and TSP Graphs are equivalent. Our special interest in Co-Graph comes from the very straightforward principle of their construction: it is based on vertex grouping and edge factorisation. This proves to be most appropriate in the scope of structured system design, since it amounts to a hierarchical decomposition of the set of nodes, with similarity of communication pattern taken as the criterion for grouping.

Property_1:

For any Co-Graph, there exists a tree representation (which is unique up to simple syntactic rewriting rules). The tree representation allows a complete and exact reconstruction of the original graph.

As an example, let's consider three equivalent Co-Graph representations: a graph, a tree with labeled nodes, and a formal expression (see Fig. 5). The following definition indicates how the tree representation should be read, and then how to re-build the original relationship from the tree or the formal expression.

Definition 1 (communication model tree representation)**.** All elements in a cluster have the same relationship (communication) with other elements out of the cluster:

elements gathered at a node with label "+" are in relation (communicate with one another).

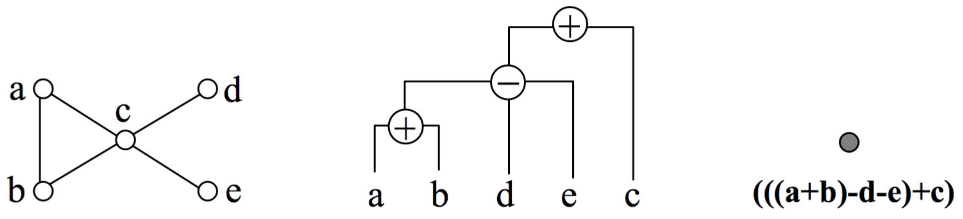


FIGURE 5. Equivalent Co-Graph representations.

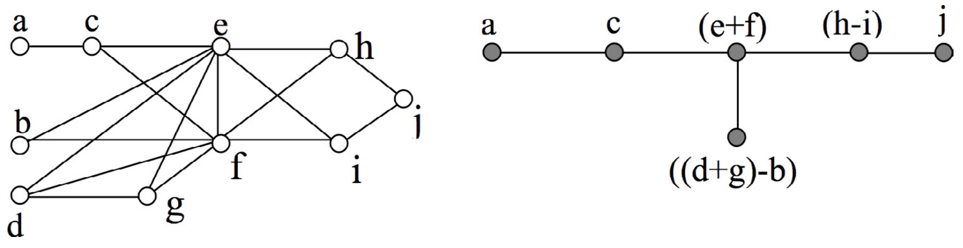


FIGURE 6. Exact Canonical Decomposition (ECD).

Elements gathered at a node with label “−” are not in relation (do not communicate with one another).

These properties clearly indicate how the original graph may be rebuilt from its Co-Graph.

The Co-Graph computation may also be viewed as a graph compression method. For instance, the previous five vertices and five edges original graph reduces to a graph with no edge and only one vertex labeled by: $(((a + b) - d - e) + c)$.

3.2.2. *Exact canonical decomposition*

The Co-Graph definition can be extended to achieve an accurate compression of any graph [1]. The result is the ECD which is a unique Exact Canonical Decomposition, that can no more be compressed and shows as a graph whose vertices are Co-Graphs. The compressed graph edges account for mutual common intercommunication between the vertices of the various Co-Graphs, while the inner structure of the compressed graph vertices is interpreted according to the usual Co-Graph semantics. So, the ECD computation appears as an optimal edge factorisation (see Fig. 6).

3.2.3. *Approximate canonical tree decomposition*

Our ultimate purpose is to associate a hierarchy to any communication graph. This not possible without going beyond the ECD. Further grouping of ECD vertices is necessary to provide the tree structure we need, but consequently, an unavoidable loss of accuracy must be accepted. This loss is minimised. The grouping

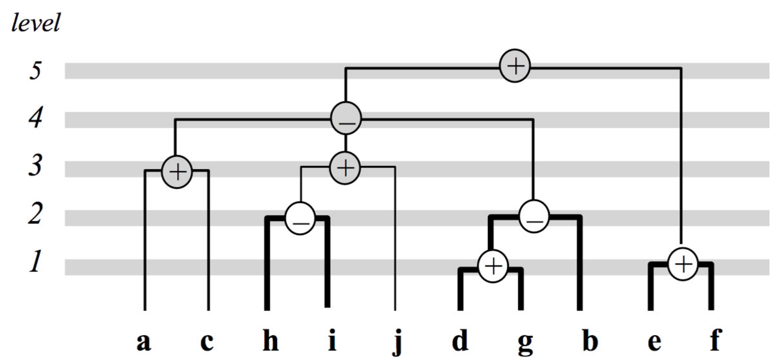


FIGURE 7. Approximated Canonical Tree Decomposition (ACTD).

process is based on clustering algorithms that exploit both a Euclidean and probabilistic structure built on P (and then extended on POW(P) in the natural way).

Let's consider P as a set of n vectors in an n-dimensional space (say: F). We consider matrix $\mathbf{G}$ as the set of coordinates of P vectors onto the canonical basis of F. Assuming that P is weighted by what we interpret as a probability measure (say: $\{p_i\}(i = 1 : n)$), there are strong arguments for using this probability to build the Euclidean distance [1].

More precisely, let M be the $n \times n$ diagonal matrix induced by the probability measure:

$$M = [\delta_{i,j}\, p_i]\ (i, j = 1...n).$$

Then, M defines a scalar product, hence a Euclidean distance on F:

$$\forall x, y : F \times F \bullet < x; y >_M = x'My \wedge d^2(x; y) = (x - y)'M(x - y) \wedge \|x\|^2 = x'Mx$$

(x' denotes x transposed).

Any hierarchical clustering algorithm [2,19] can take benefit from the Euclidean structure and achieve a wise grouping of ECD vertices, using inertia criteria, and finally providing a complete tree structure.

The Approximate Canonical Tree Decomposition (ACTD) is the ultimate tree structure; it is not an exact decomposition of the initial graph, in that it does no allow an exact reconstruction of G. Nevertheless, the ACTD is optimal, and this optimality is extensively discussed in [1]. (As a matter of fact, there is a functional equivalence between the inertia criterion and the risk of error during the reconstruction process.) We shall not recall here the algorithms nor the theory which founds the hierarchy calculus, but simply illustrate the main points and focus on the quality metric, once the hierarchy is obtained.

For instance, the example in Figure 6, would reduce to the structure (ACTD) in Figure 7. Thick lines indicate the genuine Co-Graphs in the ECD, whilst thin ones indicate the clusters that have been made to complete the ACTD. The principle is that nodes being close with respect to the Euclidean distance, will be gathered in a cluster at a lower level than nodes being more distant. This also means that clusters in the hierarchy almost have the same properties than CoGraphs have (*cf.* Def. 1).

4. Distance between hierarchies

When building the quality metric, we retain the hierarchical structure and ignore the set of node labels (*i.e.* "+" and "−"). This is a natural point of view, since we are not interested in rebuilding the original communication model, but simply aim at confronting two hierarchies. Within this scope, node labels are irrelevant. What is important is whether elements in a node do have similar communication patterns, whatever these patterns effectively are.

Then, at this point, both the architecture and the communication model actually are represented by hierarchies on the same set of leaves.

In order to define a distance between hierarchies, let us first consider the ultrametric preorder relationship which is equivalent to any given hierarchy. In a second step, we shall define a Euclidean distance on preorders, that will consequently apply to the corresponding hierarchies.

4.1. Hierarchies, ultrametric distances and ultrametric preorders

4.1.1. *Ultrametric distance*

Ultrametric preorders are based on ultrametric distances. We shall first discuss ultrametric distances and then derive the preorder definition. A common manner for representing hierarchies is to use an ultrametric distance on the set of leaves. The distance (UM) between two leaves is the rank of the lowest level where these two leaves happen to be gathered in one cluster. The knowledge of the distance is equivalent to that of the hierarchy.

$$\forall x, y : P; d : 1...h \bullet \mathrm{UM}(x, y) = d \Longleftrightarrow \\ d = \min(\{t : 1...h \mid (\exists C : \mathrm{dom}(\mathrm{HC}(C) \wedge \{x, y\} \subseteq C)\}). \quad (1)$$

The distance UM is said to be ultrametric since it verifies a stronger property than the usual triangular inequality, namely:

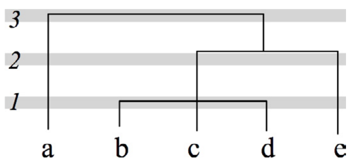
$$\forall x, y, z : P \bullet d(x, y) \leq \max(d(x, z), d(z, y)).$$

There is an equivalence between the set of hierarchies on P and the set of ultrametric distances on $P \times P$ [19].

When the elements in P are conveniently ordered, the ultrametric distance matrix comes with the special form that appears in Figure 8, *i.e.*:

- beyond the diagonal, all elements in a row are in a non descending order, and
- this side of the diagonal, all elements in a column are in a non ascending order.

UM	**a**	**b**	**c**	**d**	**e**
a	0	3	3	3	3
b	3	0	1	1	2
c	3	1	0	1	2
d	3	1	1	0	2
e	3	2	2	2	0



bc = bd = cd $\prec$ be = ce = de $\prec$ ab = ac = ad = ae

FIGURE 8. Hierarchy, ultrametric distance and ultrametric preorder.

4.1.2. *Ultrametric preorder*

A preorder is a binary relationship that proves to be both reflexive and transitive but not necessarily anti-symmetric. So, orders are anti-symmetric preorders. Consequently, a preorder is identical to a set of ordered equivalence clusters. Each equivalence cluster is made of elements verifying both $x \prec y$ and $y \prec x$ (where "$\prec$" is the preorder symbol) [19].

The set of ultrametric preorder relationships on $P \times P$ is equivalent to the set of hierarchies on P. In fact the preorder relationship definition is deduced from the ultrametric distance definition:

$$\forall x, y, z, t : P \bullet (x, y) \prec (z, t) \iff \mathrm{UM}(x, y) \leq \mathrm{UM}(z, t).$$

Any preorder relationship can be represented by its graph matrix. As an illustration, consider the hierarchy with 5 leaves together with its corresponding ultrametric distance and preorder in Figure 8, and the corresponding matrix of the preorder relationship in Figure 9.

4.2. A EUCLIDEAN DISTANCE ON PREORDERS

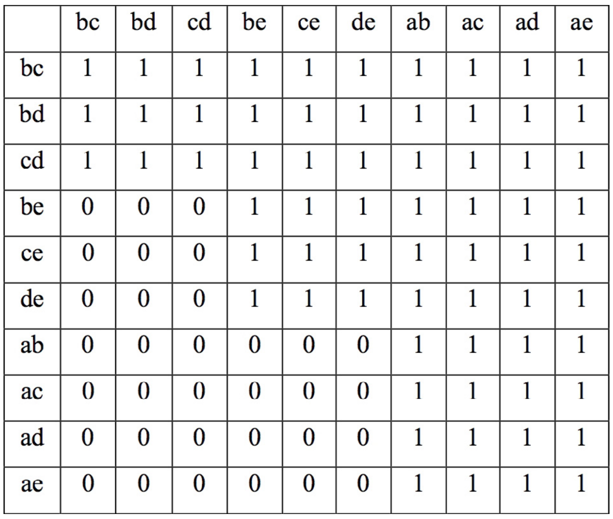
4.2.1. *Distance definition*

As announced above, we intend to derive a distance between hierarchies from a distance on the corresponding preorders. There is a simple manner to quantify the discrepancy between two preorders which is to compute the number of their inversions. One inversion occurs when two pairs of elements are ordered differently in one preorder and in the other. Then, the semantics of the discrepancy are quite explicit. Let us give some formal definitions below.

Let H_1 and H_2 be two hierarchies on P, and let O_1 and O_2 be the two corresponding preorders on $P \times P$. The preorder relationships respectively are denoted by the following symbols: $\prec_{O_1}$ and $\prec_{O_2}$.

A formal definition of the discrepancy – say "Inv" – between preorders is given below:

$$\mathrm{Inv}(O_1, O_2) = \#\{p, q : P \times P \mid p \prec_{O_1} q \wedge p \not\prec_{O_2} q\}\cdot$$



	bc	bd	cd	be	ce	de	ab	ac	ad	ae
bc	1	1	1	1	1	1	1	1	1	1
bd	1	1	1	1	1	1	1	1	1	1
cd	1	1	1	1	1	1	1	1	1	1
be	0	0	0	1	1	1	1	1	1	1
ce	0	0	0	1	1	1	1	1	1	1
de	0	0	0	1	1	1	1	1	1	1
ab	0	0	0	0	0	0	1	1	1	1
ac	0	0	0	0	0	0	1	1	1	1
ad	0	0	0	0	0	0	1	1	1	1
ae	0	0	0	0	0	0	1	1	1	1

FIGURE 9. Matrix of the preorder relationship between pairs in P.

An algebraic equivalent definition can be given, which uses the matrices of the preorders (here also denoted by bold face symbols $\mathbf{O_1}$ and $\mathbf{O_2}$):

$$\begin{aligned}\mathrm{Inv}(O_1,O_2) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{O_1}(i,j)(1-\mathbf{O_2}(i,j)) + \sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{O_2}(i,j)(1-\mathbf{O_1}(i,j))\\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}(\mathbf{O_1}(i,j)-\mathbf{O_2}(i,j))^2\\ &=< \mathbf{O_1}-\mathbf{O_2}, \mathbf{O_1}-\mathbf{O_2} >_{\mathrm{HS}}\\ &= \|\mathbf{O_1}-\mathbf{O_2}\|^2_{\mathrm{HS}}\end{aligned}$$

where $\langle X, Y\rangle_{\mathrm{HS}}$ denotes the Hilbert Schmidt scalar product upon bilinear $\mathcal{R}^n$ operators (*e.g.*: $n \times n$ matrices) [12,17]. As a matter of fact, $\langle X, Y\rangle_{\mathrm{HS}}$ equals the trace of the product of X and Y transposed: $\mathrm{trace}(X \times Y')$.

Hence comes the following:

$$\mathrm{Inv}(O_1,O_2) = \mathrm{trace}((\mathbf{O_1}-\mathbf{O_2}) \times (\mathbf{O_1}-\mathbf{O_2})')$$

(where the prime denotes matrix transposition).

Consequently, counting the number of inversions between two preorders is equivalent to defining a Euclidean distance.

Definition 2 (Euclidean distance between hierarchies)**.** The discrepancy between two hierarchies is defined as the square root of the number of inversions between the two corresponding preorders.

$$\boxed{d(H1, H2) = \sqrt{\mathrm{Inv}(O1, O2)} = \sqrt{\mathrm{trace}((\mathbf{O_1} - \mathbf{O_2}) \times (\mathbf{O_1} - \mathbf{O_2})')}}$$

This discrepancy measure actually is a Euclidean distance upon the set of hierarchies.

The whole process leading to the metric definition is summed up in the following section.

5. Quality metric, definition and scale

5.1. Quality metric definition

At this point, let us recall below the scheme of our reasoning, for providing a rough quality metric to a modular system specification or implementation.

Definition 3 (Quality measurement process)**.**

(i) We define the quality of a specification as the degree of coherence between the architecture model (Arch) and the communication model (Comm).
(ii) Both architecture and communication models respectively are accurately represented by hierarchies (H_1,H_2) and preorders (O_1, O_2).
(iii) The number of inversions (Inv) between two preorders is a Euclidean distance (d).
(iv) This Euclidean distance is the rough quality metric(Rgh_QUAL) we propose between the architecture model and the communication model.

$$\boxed{\mathrm{Rgh\_QUAL(Arch,\ Comm)} = d(H_1, H_2) = \sqrt{\mathrm{Inv}(O_1, O_2)}}$$

As a matter of fact, a good quality level is indicated by a low value of the quality metric Rgh_QUAL. It would be straightforward to propose any decreasing function to make things perhaps more natural (*i.e.* a good quality level denoted by a high metric value). We shall not discuss this point, and prefer to focus on the issue of scaling the rough metric. In fact, a rough metric is convenient for ranking several systems, but is of poor use for making an absolute opinion about the quality level of a particular specification.

A rough metric needs to be scaled before being of practical use. This scaling is addressed in the next section.

5.2. Simulation for metric scaling – Stochastic approach

The scaling we propose is obtained by a study of the probability distribution of the distance between two random hierarchies drawn from a given population.

The sampling process, the stochastic model, and possible constraints on the type of the random hierarchies are specified hereafter.

5.2.1. *Stochastic model*

The first obvious point when computing the distance between two hierarchies in the way we propose, is that operators, matrices and preorders necessarily are to be defined upon the same set of elements. Then the two hierarchies have identical sets of leaves.

Secondly, the type of a hierarchy is characterised by its number of levels, and also by the distribution of children among the nodes of each level.

The most simple stochastic model assumes a uniform distribution of the number of levels (ranging in $2...(n-1)$), and a uniform distribution of the children among the nodes. Under such conditions, stochastic arguments may advocate for assigning an asymptotic Gaussian model to the square distance between two random hierarchies having an identical set of leaves.

This stochastic model is made more precise below. We use it in two different ways.

On the one hand, we refer to the stochastic model to simulate a set of random hierarchies and then observe empirical estimates for the distribution of the quality metric (empirical distribution model and its estimated parameters) between these random hierarchies. These parameters and the empirical distribution provide us with likelihood estimates that eventually allow to scale the rough metric. This is a so-called distribution-free approach, since no assumptions are made upon the random variables distribution.

On the other hand, we assume a Gaussian model for the metric distribution. In this case, the simulation process only is of use for estimating the mean and standard deviation of the Gaussian random variable, and then build confidence intervals, critical regions or likelihood estimates that finally found the decision process.

5.2.2. *Simulating process*

The simulating process embodies the stochastic model above. More precisely, in order to detail the operational way for providing random hierarchies, we shall proceed in two steps:

- generate a random *binary* hierarchy on the set of leaves;
- at each level, nodes are provided with the uniform probability distribution, and the two elements that are to be joined are randomly drawn from the set of candidate nodes;
- condense the binary tree at random levels.

A possibly constrained number of levels (according to the stochastic model the analyst has chosen) is randomly selected, and the corresponding actual levels where the hierarchy will be condensed also are randomly drawn from the set of existing levels); from this process, results the final random hierarchy.

Let $\mathbf{H}(n, l_1, l_2)$ be the set of all hierarchies on n leaves, with a number of levels ranging within $l_1...l_2$. Of course the following constraints must be verified: $1 < l_1 < l2 < n$. $\mathbf{H}(n, l_1, l_2)$ – as well as the $l_1...l_2$ interval – are provided with the uniform distribution.

5.2.3. *Quality scaling – Normalised quality index*

Let Harch (resp. Hcomm) be the hierarchy that embodies the actual architecture model (resp. communication model) in the specification under analysis. Then let us assume that the rough metric, *i.e.* the observed distance between Harch and Hcomm effectively is d(Harch,Hcomm). Low values of d(Harch,Hcomm) indicate a high quality level for the specification.

Let Hrand$_1$ and Hrand$_2$ be two non dependant random hierarchies in $\mathbf{H}(n, Constr)$. Hierarchies on n leaves, with possible additional constraints ($Constr$), for instance expressing that Hrand$_1$ and Hrand$_2$ respectively are of types similar to those of Harch and Hcomm.

We shall assume here that the type of a hierarchy is the number of levels. Stronger or weaker constraints could be imposed instead.

The Scaled Quality Metric: Scl_QUAL(Harch, Hcomm) is defined as the probability that two such random hierarchies should not happen to be closer to one another (with respect to the Euclidean distance d) than Harch and Hcomm actually are.

Definition 4 (Scaled quality metric)**.** (Simulation approach.)

$$\boxed{\begin{aligned}&\mathit{Scl\_QUAL(Harch,\ Hcomm)} =\\ &\quad \mathit{Prob(\ d(Hrand_1,\ Hrand_2) > Rgh\_QUAL(Harch,Hcomm)\ )}\end{aligned}}$$

Once the formal definition of the Scaled Quality Metric is given, operational estimates and computational methods should be given too. This is done by analysing simulation samples. A first approach is distribution-free, and another refers to the Gaussian distribution.

*Distribution-free approach: Scl_QUAL^*

Let D_{sim} be the random variable that equals: d(Hrand$_1$, Hrand$_2$).

Let $S = \{d_{\text{sim}_i}\}(i : 1...s)$ be the sample of the s observations of D_{sim}, resulting from a non exhaustive random drawing process from $\mathbf{H}(n, l_1, l_2)$. Where l_1 and l_2 are the constraints on the number of levels such that:

$$l_1 = \#\ \text{dom HP(Harch)} = \#\ \text{dom HP(Hrand}_1)$$

(we assume l_1 is less than l_2, otherwise indices should be permuted)

$$\text{and}\quad l_2 = \#\ \text{dom HP(Hcomm)} = \#\ \text{dom HP(Hrand}_2).$$

The rate of occurrence of the event $[d(\text{Hrand}_1, \text{Hrand}_2) > \text{d(Harch, Hcomm)}]$ is an estimate of Scl_QUAL(Harch,Hcomm). This estimate will be denoted by: Scl_QUAL^ (Harch,Hcomm).

In order to judge of the significance of d(Harch,Hcomm), we shall also compare it to characteristic values such as: min(S), mean(S) and max(S).

In addition to the previous point estimates, asymptotic confidence intervals may be given for more precise information.

Definition 5 (Scaled quality estimate)**.**

$$\boxed{\begin{array}{l} \textit{Scl\_QUAL}\hat{}\,(\textit{Harch, Hcomm}) = \\ \qquad \textit{Freq}(\ d(\textit{Hrand}_1,\ \textit{Hrand}_2) > \textit{Rgh\_QUAL}(\textit{Harch,Hcomm})\ ) \end{array}}$$

Where Freq is for "observed frequency", and where Scl_QUALˆ is the estimate for Scl_QUAL.

*Assuming that d^2(Harc,Hrand) is Gaussian): Scl_QUAL**

When a Gaussian model is assumed for the square distance – say $N(\mu, \sigma)$ – its parameters are estimated as usual, from the simulation sample of size s, and Scl_QUAL can then directly be computed with a reference to the Student law with $s - 3$ degrees of freedom.

Definition 6 (Scaled quality metric)**.** (Gaussian model.)

$$\boxed{\textit{Scl\_QUAL*}(\textit{Harch, Hcomm}) = \textit{Prob}(T_{s-3} > (d^2(\textit{Harch, Hcomm}) - m)/S)}$$

Where T_{s-3} represents the Student distribution, and m (resp S) the empirical estimates for μ (resp. σ). In case s is important ($s > 50$), the Student law can be approximated by a standard Gaussian distribution $N(0, 1)$.

Let us note that the metric Scl_QUAL* exactly refers to the same event as Scl_QUALˆ does, namely $[d(\mathrm{Hrand}_1, \mathrm{Hrand}_2) > \mathrm{d}(\mathrm{Harch}, \mathrm{Hcomm})]$, but in order to use the Gaussian hypothesis, both terms in the inequality have been squared and normalised, so that the final Student random variable appears in the LHS of the inequality.

6. Application example: SDL

The following example is about the "Production Cell" case study, which is a well known benchmark, among the formal specification community [20]. What is presented below is a simplified version, since our aim only is to demonstrate our proposal on an example, and in no way to achieve an actual and complete study about the quality of the Production Cell specification.

The original SDL specification has been achieved using Verilog's GEODE tool. For our purpose, all information about the production Cell architecture and communication models have been captured by use of a custom parser applied on SDL-PR sources. SDL-PR is a textual counterpart of the usual graphical SDL-GR specification. SDL-PR is generated on demand by both of the main SDL specification environments: Verilog's GEODE and Telelogic's TAU.

For analysing the quality of the Production_Cell specification, the analyst is free to select a convenient granularity: he may deal with elementary processes

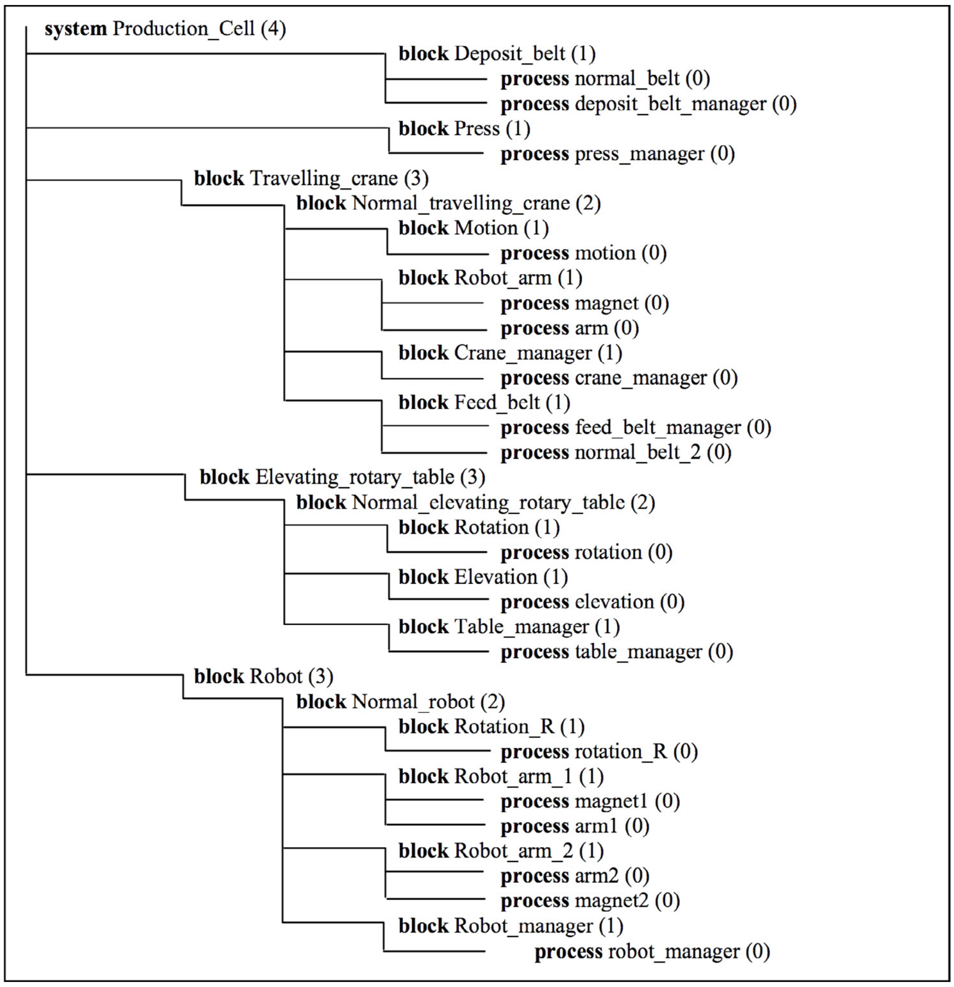

system Production_Cell (4)
block Deposit_belt (1)
process normal_belt (0)
process deposit_belt_manager (0)
block Press (1)
process press_manager (0)
block Travelling_crane (3)
block Normal_travelling_crane (2)
block Motion (1)
process motion (0)
block Robot_arm (1)
process magnet (0)
process arm (0)
block Crane_manager (1)
process crane_manager (0)
block Feed_belt (1)
process feed_belt_manager (0)
process normal_belt_2 (0)
block Elevating_rotary_table (3)
block Normal_elevating_rotary_table (2)
block Rotation (1)
process rotation (0)
block Elevation (1)
process elevation (0)
block Table_manager (1)
process table_manager (0)
block Robot (3)
block Normal_robot (2)
block Rotation_R (1)
process rotation_R (0)
block Robot_arm_1 (1)
process magnet1 (0)
process arm1 (0)
block Robot_arm_2 (1)
process arm2 (0)
process magnet2 (0)
block Robot_manager (1)
process robot_manager (0)

FIGURE 10. Cell_0 and Cell_1 common System Overview Diagram.

and procedures or conversely choose to deal with prior aggregates (*i.e.* blocks or sub-blocks). Here, for sake of simplicity, we chose to consider the processes as elementary modules.

6.1. CASE STUDY: PRODUCTION CELL

We thank Prof. Lindner [20] who provided us with a complete SDL specification of the Production Cell.

There are two specifications: Cell_0 and Cell_1. Both do share the same architecture model, but the communication models are different. That is to say Cell_0 and Cell_1 are associated to a sole hierarchy (namely Cell.arc) that embodies the

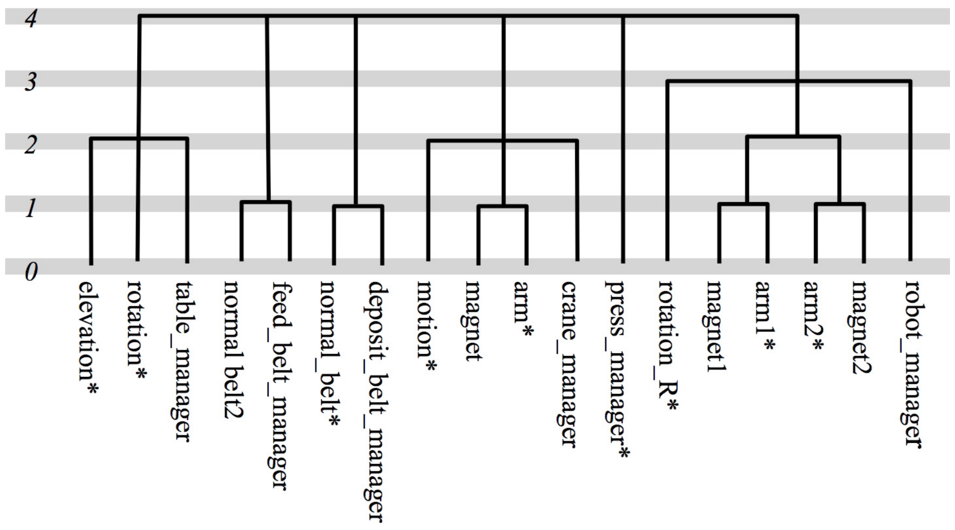


FIGURE 11. Cell_0 and Cell_1 common Architecture Model (Harch).

specification architecture. Figure 10 is a copy of the System Overview Diagram (SOD) which is provided by the SDL design tools as a global architecture graphical representation. The SOD is also described by its textual specification in SDL_PR. We have designed an analyser that computes our architecture model from the textual specification of the SOD. Figure 11 shows the resulting hierarchy, where the processes are the leaves, and where the Cell_0 SDL System is the root node.

In contrast, Cell_0 and Cell_1 communication models are different. Here, we shall present only Cell_0 communication model in detail (*cf.* Fig. 12 for Cell_0.com). Information and results about Cell_1 will be mentioned as complementary examples.

6.2. Production cell architecture model

According to the System Overview Diagram (SOD), the Production Cell system consists of 18 processes which are the leaves of Harch hierarchy in Figure 11. Both SOD and Harch have 5 levels. The nodes of Harch correspond to blocks and subblocks in the SDL specification. For instance: Block Robot, Block Press, Block Crane, Blocks Deposit and Feed Belt, and Block Table Manager, that can easily be identified in Figure 11 at level 3. The concrete semantics of the leaves and nodes are not really important here, and only the node names are given.

There are in fact several options about the way to translate a SOD into an architecture model (*e.g.*: which nodes to be selected as leaves? whether applying a top-down or a bottom-up propping up of levels?). All this is not discussed here, since it does not interfere with the grounds of the method.

6.3. Cell communication model (ACTD)

Extracting the communication model from SDL sources is not as straightforward as extracting the hierarchical structure from the SOD. We do not intend to give the extensive formal definition of how communication is specified and captured in this problem, but simply provide basic indications and rules to outline our point of view. (The complete formal definitions are given in our tool user's manual.)

- A communication between two blocks is acknowledged in case inner elements do communicate.
- Two processes communicate in case they are linked by a path through channels and signal routes, with at least one signal exchanged.
- Procedures are taken out of their parent module to constitute isolated modules.
- A procedure communicates with its parent.
- A process communicates with the procedures it calls.
- Procedures in a same process communicate if they share global variables.
- ...

All of these pieces of information, about communication between modules are drawn from SDL-PR source, by parsing SDL channels and signal route declarations, signal input and output, procedure calls and variables names within SDL processes and procedures. However, it is clear that dynamic aspects such as process creation and output with Pid expressions cannot be completely handled in a deterministic manner. We propose several alternative choices according to what effort is accepted to be paid for code analysis. An ultimate attitude would be to scan traces of system behaviour for achieving a full capture of dynamic communication.

As said above, the whole SDL specification is necessary to build the communication model. So it is not feasible here to present with all the system, block, sub-block and process diagrams in order to detail the correspondence between the SDL specification and the resulting communication graph between processes, which captures the communication model. So, only the final communication graph Cell_0.com is presented in Figure 12.

The first step in the analysis of the communication model Cell_0.com is to compute the Exact Canonical Decomposition (ECD) (Fig. 13) which allows to recognise three non trivial Co-Graphs, with only one level and labeled by "–". The set of all Co-Graphs (*i.e.* including trivial singleton Co-Graphs) is of size 10.

Then, computing the Approximated Canonical Tree Decomposition (ACTD) leads to an 8 level hierarchy. The Hcomm hierarchy in Figure 14 is the best one for representing the communication model.

6.4. Results and discussion

The question is to estimate the quality of Cell_0 specification, and according to our approach, the point is to determine whether the architecture model Cell_0.arc

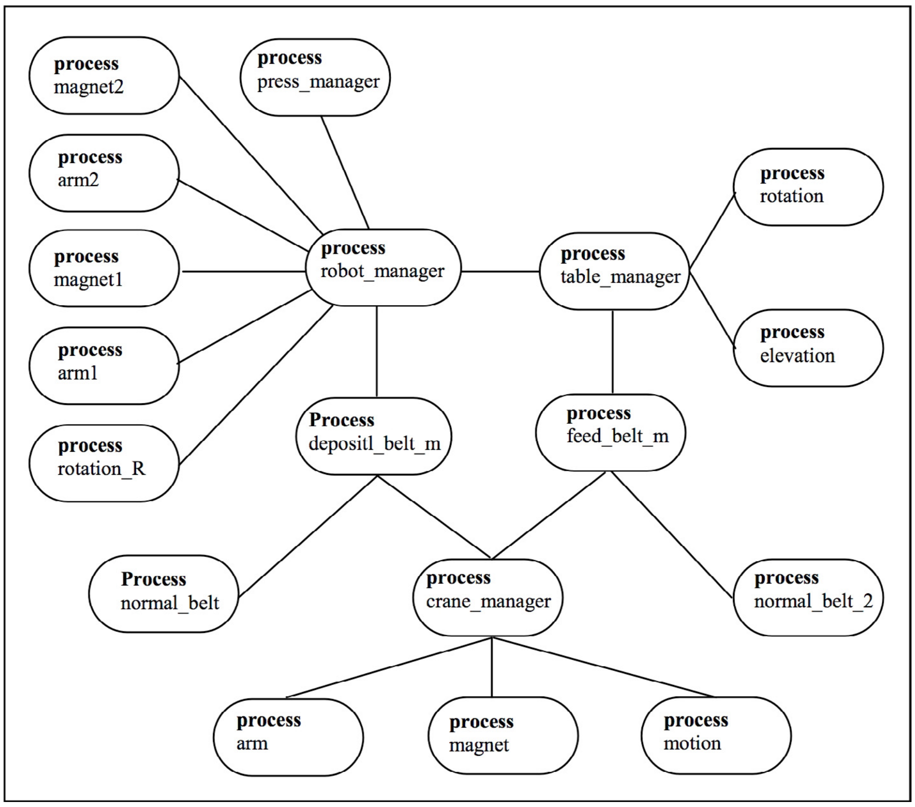


FIGURE 12. Cell_0.com: Process Communication Graph
(as captured from SDL_PR source specification).

and the communication model Cell_0.com are consistent. Harch and Hcomm hierarchies respectively account for Cell_0.arc and Cell_0.com

The following numerical results have been computed for Harch and Hcomm.

6.4.1. *Rough metric*

Rgh_QUAL0 computes the Euclidean distance between the architectural and the communication models. It is in fact the square root of the number of inversions between the two preorder which are equivalent to Harch and Hcomm respectively.

$$\begin{aligned}\text{Rgh\_QUAL\_0} &= \text{Rgh\_QUAL (Cell.arc, Cell\_0.com)}\\ &= \text{d(Cell.arc, Cell\_0.com)}\\ &= \mathbf{70.35}\end{aligned}$$

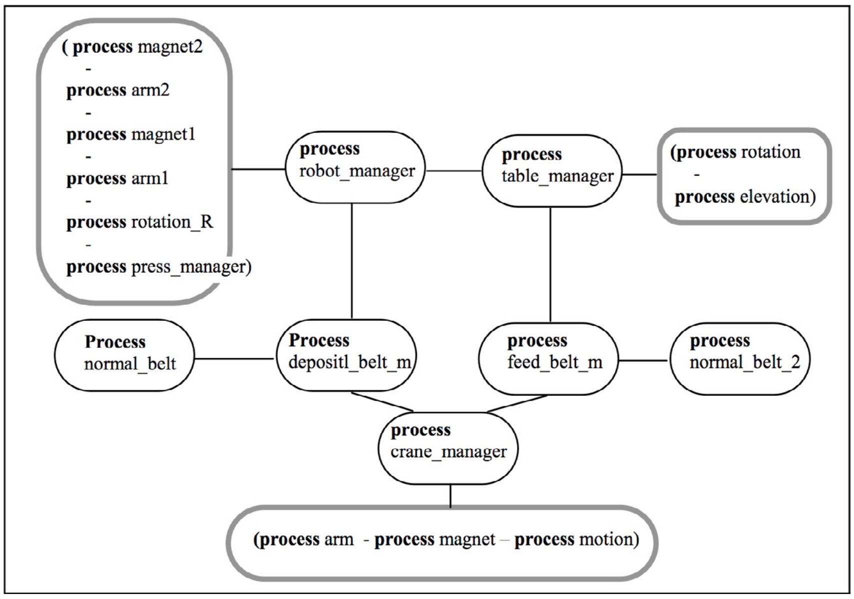


FIGURE 13. Exact Canonical Decomposition (ECD) of the Communication Graph for Cell_0.

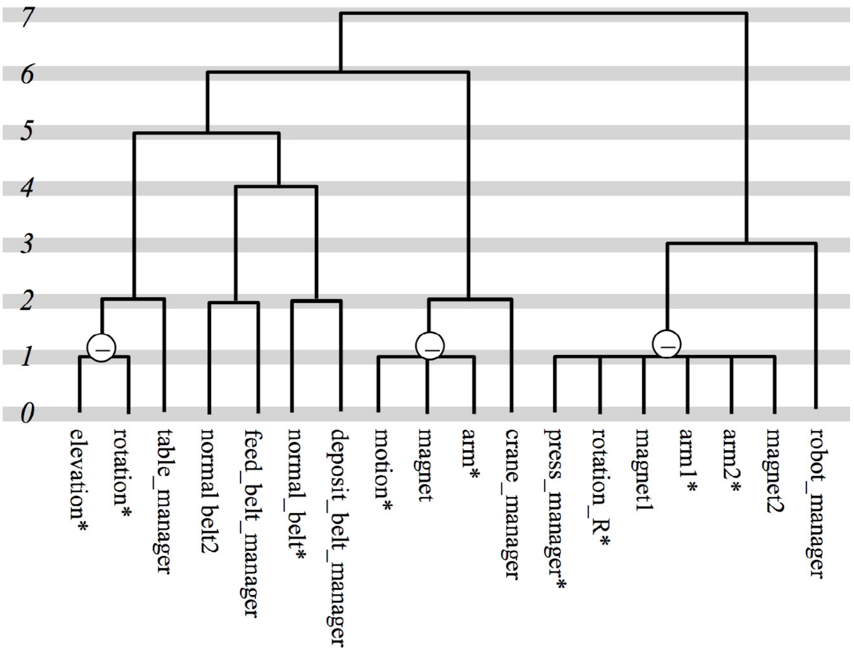


FIGURE 14. Approximate Canonical Tree Decomposition (ACTD) of the Communication Graph for Cell_0 (*i.e.*: Hcomm).

I.e.: 4949 observed inversions out of at most the $46665 = 18{\times}17{\times}(18{\times}17-1)/2$ "possible" ones for hierarchies with 18 leaves. This leads to a 0.106 rough ratio. As a matter of fact, there is no real convincing argument to discuss such a raw value. Of course, the Rgh_QUAL_0 ratio ranges within $[0, 1]$, but the distribution of expected ratio values for random hierarchies is not uniform, hence interpreting the rough ratio value is not accurate. Scaling is the solution, and scaled metrics examples are given hereafter.

6.4.2. *Scaled metrics*

Distribution free approach

The first scaling presupposes no assumption on underlying statistical models, *i.e.* we adopt a distribution free approach. The following results have been obtained by simulating 1000 occurrences of a random pair of hierarchies – namely (Hrand_1, Hrand_2) – in $\mathbf{H}(18, 5, 5) \times \mathbf{H}(18, 8, 8)$, *i.e.*: 18 leaves, and respectively 5 and 8 levels according to the types of the architecture Harch and communication HComm hierarchies. For all these random pairs, we computed the Rgh_QUAL metric, and so obtained the empirical Rgh_QUAL(Hrand_1, Hrand_2) distribution. We then computed the rate of occurrence of a simulated value exceeding the observed one: Rgh_QUAL_0.

This leads to the Scl_QUALˆ scaled metric value:

Scl_QUALˆ (Cell.arc, Cell_0.com) = **0.973**
CI(95%): [0.963; 0.983]
(simulated sample size: s = 1000)

The computation of the empirical distribution of Rgh_QUAL moreover allows us to compute a – say 95% – confidence interval in addition of the point estimate Scl_QUAL.

Here we are provided with a most interesting information. In fact, we may ensure that less than three times out of one hundred, two random non dependant hierarchies happen to be closer to one another than the two hierarchies under study: Harch and Hcomm.

That is to say Harch and Hcomm are very similar, and hence Cell_0.arc and Cell_0.com are very consistent. The Cell_0 specification shows a very high quality level as regards architectural and communication models consistency concerns. It is obvious that the analyst has wisely and strongly worked in order to build a consistent specification with no haphazard features. Such a precise piece of information was hidden by the sole rough metric.

The minimal, mean and maximal values for the simulated Scl_QUALˆ bring some additional information: the observed Rgh_QUAL_0 value is far under the mean value for random hierarchies.

$$\min\{d(\text{Hrand}_1, \text{Hrand}_2)\} = 61.37$$
$$\text{mean}\{d(\text{Hrand}_1, \text{Hrand}_2)\} = 91.67$$
$$\max\{d(\text{Hrand}_1, \text{Hrand}_2)\} = 107.67$$

Gaussian model assumption

The above simulating process can be lightened, provided a prior probabilistic (*e.g.*: Gaussian) model is assumed for the squared Rgh_QUAL random variable, in case the two hierarchies in hand are themselves assumed random and non dependant.

With such an approach, its only necessary to obtain estimates for the mean and standard deviation of the Rgh_QUAL variable for random hierarchies. The results are the following:

$$\textit{mean estimate for } d^2(\textit{Hrand}_1, \textit{Hrand}_2) = 8469.2$$
$$\textit{standard deviation estimate for } d^2(\textit{Hrand}_1, \textit{Hrand}_2) = 1725.5$$

$$\textbf{Scl\_QUAL*}(\textit{Cell.arc}, \textit{Cell\_0.com}) = \text{Prob}(T_{15} > -2.04) = \mathbf{0.970}$$

One can check that the two approaches above lead to scaled metrics values that are very close to one another. More precisely, the Scl_QUAL* value clearly lies within the 95% confidence interval computed for Scl_QUALˆ, so the observed discrepancy between the two estimate actually is statistically not significant.

We carried out a similar analysis on the alternative Cell_1 specification. As complementary information, we present the results we obtained without going into the specification details.

Rgh_QUAL_1 = **Rgh_QUAL**(Cell.arc, Cell_1.com)
= d(Cell.arc, Cell_1.com) = **71.23**

Scl_QUALˆ (Cell.arc, Cell_1.com) = **0.969**
with CI(95%): [0.958; 0.980]

$\textbf{Scl\_QUAL*}(\textit{Cell.arc}, \textit{Cell\_1.com})$
$= \text{Prob}(T_{15} > -1.98) = \mathbf{0.967}$

d(Cell_0.com, Cell_1.com) = 29.66

It can be seen, (due to the wide intercept of confidence intervals for the quality level), that there is no significant difference between the quality of Cell_0 and Cell_1, both are excellent. As a confirmation, the distance between the two communication models Cell_0.cm and Cell_1.com has been computed as the number of inversions between the preorders respectively associated to Hcomm_0 and Hcomm_1. The observed result of 29.66 is extremely low. The probability for obtaining such a pair of close hierarchies by chance is under 1/10 000. So, other quality items but consistency, should be put forth in order to decide which is better , between Cell_0 and Cell_1 specifications.

However, it is important to understand that the small number of inversions between the hierarchies associated with Cell_0.com and Cell_1.com does not necessarily imply that the communication models are similar. This strictly means that the hierarchies are close to one another. In fact the underlying architectures, that organise and group together processes that have similar communication patterns are nearly the same, but as we decided not to retain the labels ("+" and "−") in the communication models (since they have no architectural counterpart), it should not be deduced that the communication patterns are equal in Cell_0 and Cell_1 (and they actually are not). Direct equality between communication graphs is out of concern for quality, the point actually is that, for a good quality level, groups of processes that appear in an architectural hierarchy should have similar communication patterns, whatever these patterns are, and this precisely is the case in our approach. We eventually conclude that if communication models that are similar, naturally have similar quality levels, on the other hand, distinct quality models may be of equivalent quality, *i.e.* may be both consistent with a given architecture.

7. Conclusion

All we have proposed in this paper treats of consistency, considered as an aspect of the quality in software specifications. We addressed a special case of consistency, which concerns the software architecture and the communication models. All items necessary to carry on the quality analysis can be automatically captured from the source documents (specification or code). The basic constraint is that a hierarchical architecture model makes sense. This is the case in modular specifications and programs; this can also be the case in an object oriented approach, by means of an embedded package organisation.

The rough quality metric is computed on mathematical representations (preorders and hierarchies) that strictly are either equivalent to the initial information (The ECD is equivalent to the communication graph) or else optimal estimates (ACTD is the best tree approximation of the communication graph). The number of inversions between preorders, which is the metric we propose, obviously is a quality factor that directly accounts for the consistency between the architecture and the communication model.

The simulation process which is founded on explicit stochastic hypothesis allows scaling the rough metric. The result is an absolute metric, in that it depends neither on the size factors nor on other contextual parameters. Its semantics are expressed in terms of the probability that a given quality level could have been obtained by a blind specification process. In the Production Cell example where Scl_QUALˆ is 0.973, this means that among 1000 random architectures, the software engineer chose one among the 27 best (*i.e.* among the 27 closest to the communication model, and hence among the 27 most consistent). This seems to appear as a convincing quality assessment. It might be objected that the stochastic estimates we presented are in some sense subject to a bias, in that we consider all possible hierarchies for drawing the random samples, and do not limit ourselves to hierarchies that could actually be implemented. Anyway, our strategy could easily be adapted in order to take any additional constraint into account during the simulation process, as soon as this constraint is explicitly formulated.

In case the quality is very high, the distribution-free estimates became not precise enough, unless very important samples are used. In fact, very low probabilities for rare events are to be estimated. Then referring to a Gaussian model is a solution which only needs estimates of the mean and standard deviation of the distance between random hierarchies. Under a Gaussian hypothesis, accurate estimates can be obtained even on small samples.

The aim of this paper not only is to present with a particular metric and its theoretical basis, but also to suggest that what we propose is not a stand alone result. This is a part of an integrated strategy for quantifying and improving the consistency of a specification. Let us give below three examples of such possible developments.

– Here, only global aspects have been addressed. No attention has been paid to possible improvements of the specification under analysis (one must grant that the quality of the Production Cell specification example already is high). It is clear that a local analysis can be achieved, in order to point out what precise parts of the hierarchy are responsible for a high number of inversions. Each level in the hierarchy provides with a partition of the set of leaves, and analysing the quality of such partitions – in a completely consistent way with the one presented here – was the subject of our referenced previous works.
– We only discussed one stochastic model, for sake of simplicity (*e.g.*: reference to a pair of random hierarchies of given types). Alternative approaches could be attempted. For instance, one could assume that the communication model and its hierarchy are given, and that only one hierarchy is random among the set of hierarchies having the same type as that of the architecture model. A homologous approach could set the architecture model hierarchy while the communication model is assumed to be random.

Constraints on the types of partitions could be varied (either the number of levels being assigned to a special value or ranging in an interval, setting several distributions of the nodes among the levels). Hierarchies could be squashed, possibly prohibiting gaps between consecutive node levels...

All such alternatives must not be seen as prejudicial to the method nor bringing looseness or imprecision. On the contrary, since models can explicitly be specified, they do increase the control by the analyst, and allow to judge of the robustness of the results.

– The rough metric we propose proves to be equivalent to a Euclidean distance. This important point has not been exploited here. The matter is that a Euclidean distance is liable to provide optimal planar layouts (multidimensional scaling), where the mutual distances between points on the diagram are proportional to the actual distances between the items under study. A motivating application for this, is that of analysing the quality of a set of specifications. Since the metric is absolute, even heterogeneous specifications can be confronted. Representing the set of specifications with their mutual distances on one diagram, allows to reach synthetic conclusions, by pointing out different clusters of specifications having a similar quality level.

Moreover, analysing the set of mutual distances by means of factor analysis [12,17] is a mean to exhibit underlying exogenous indirect quality factors which explain that the within cluster quality is homogeneous while the between cluster quality is heterogeneous.

At last, we believe that the way we defined our quality metric for comparing the architecture and the communication model may be considered as a pattern for creating quality metrics for other consistency aspects.

The whole process that founds the quality metrics Rgh_QUAL and Scl_QUAL could be adapted in other contexts. A major difficulty is to find one mathematical super-representation that accurately embodies both aspects of the specification, the consistency of which is to be quantified. Another is to find a reasonable distance on elements of this super-representation that has clear semantics, and can be easily computed. The final scaling process (through simulation) is straightforward and directly adaptable, as soon as the stochastic models have been clearly specified and founded.

References

[1] F. Ammar-Boudjelal, J.Y. Lafaye and G. Louis, Evaluating, Comparing and Improving the Quality of System Structure, During the Specification Process. Application Example with SDL. *Software Quality J.* **7** (1998) 195–222.

[2] Barthelemy J.P. and A. Guénoche, *Les arbres et les représentations des proximités.* Masson, Paris (1988).

[3] V. Basili, L.C. Briand and W.L. Melo, A Validation of Object-Oriented Design Metrics as Quality Indicators. *IEEE Trans. Software Engineer.* **22** (1996) 412–421.

[4] K.S. Booth and G.S. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity using PO-Tree algorithms. *J. Comput. Syst. Sci.* **13** (1976) 335-379.

[5] S. Chidamber and C. Kemerer, A Metric Suite for Object Oriented Design. *IEEE Trans. Software Engineer.* **20** (1994) 476–493.

[6] S. Chidamber, D. Darcy and C. Kemerer, Managerial use of metrics for object oriented software: an exploratory analysis. *IEEE Trans. Software Engineer.* **24** (1998) 629–639.

[7] T. Demarco and B.W. Boehm, *Controlling Software: Management, Measurement and Estimates.* Prentice-Hall (1998).

[8] J. Ellsberger, D. Hogrefe and A. Sarma, *SDL, Formal Object-oriented Language for Communicating Systems.* Prentice-Hall (1997).

[9] N. Fenton, R. Whitty and Y. Lizuka (editors), *Software Quality Assurance and Measurement: A Worldwide Perspective.* Chapman and Hall (1996).

[10] N.E. Fenton and S.L. Pfleeger, *Software Metrics: A Rigorous Approach.* PWS Publ. 2nd edition London (1996).

[11] A. Finkenstein, D. Gabbay, A. Hunter, J. Kramer and B. Nuseibeh, Inconsistency Handling in Multi-Perspective Specifications. *IEEE Trans. Software Engineer.* **20** (1994) 569–578.

[12] T. Foucart, *Analyse factorielle de tableaux multiples.* Masson, Paris (1984).

[13] P. Fradet, D. Le Métayer and M. Périn, Consistency Checking for Multiple View Software Architectures, in *Proc. of European Software Engineering Conference, FSE'99,* Springer-Verlag (1999) 410–428.

[14] P. Inverardi, H. Muccini and P. Pellicione, Automated Check of Architectural Models Consistency using SPIN, in *Proc. of Automated Software Engineering conference, ASE'2001* (2001) 322–330.

[15] INSM, *New Approaches in Software Measurement,* in *Proc. of 10th international Workshop INSM'2000,* LCNS, Springer-Verlag (2000)

[16] ISO/IEC 9126, *International Standard Information Technology – Software Product Evaluation. Quality Characteristics and Guideline.* ISO (1991).

[17] C. Lavit, *Analyse conjointe de tableaux quantitatifs.* Masson, Paris (1988).

[18] E.L. Lawler, Graphical algorithms and their complexity. *Mathematical Center Tract* **81** (1976) 3–32.

[19] I.C. Lerman, *Classification et analyse ordinale de données.* Dunod, Paris (1981).

[20] C. Lewerentz and T. Lindner, *Formal Development of Reactive Systems: Case Study Production Cell,* 2nd edition, Springer-Verlag. *Lect. Notes Comput. Sci.* **891** (1995).

[21] M. Lorentz and J. Kidd, *Object Oriented Metrics.* Prentice-Hall (1994).

[22] G. Poels and G. Dedene, Modelling and Measuring Object-Oriented Software attributes with Proximity Structures, in *Proc. of 3rd International ECOOP WQAOOSE*, Lisbon (1999) 1–22.

[23] G. Poels and G. Dedene, *Measuring Event-Based Object Oriented Conceptual Models,* L'Objet, logiciel, bases de données, réseaux, Vol. 7. Hermès, Paris (2001).

[24] J.M. Spivey, *The Z Notation: A Reference Manual.* Prentice Hall International, 2nd edition, (1992).

[25] L. Stewart, *Co-graphs, a class of tree representable graphs.* Ph.D. Dpt of Computer Science, TR 126/78, University of Toronto, Canada (1978).

[26] S.A. Whitmire, *Object Oriented Design Measurement.* J. Wiley (1997).

Communicated by Y. Métivier.
Received March 4, 2003. Accepted February 20, 2004.