A Tool for Entropy-based Analysis of Design-time and Runtime Models

Reiner Jung[1] Robert Heinrich[2] and Wilhelm Hasselbring[1]
[1]Kiel University {reiner.jung,hasselbring}@email.uni-kiel.de [2]KIT heinrich@kit.edu
Visit us at github: **https://github.com/rju/architecture-evaluation-tool**

Introduction

Models play an important role in software development and system operation. They are used for many different purposes including architecture models, adaptation plans, and runtime models. Their size, complexity, coupling, and cohesion has an impact on the evaluation of runtime models, and on the evolution and adaptation of software. Therefore, it is necessary to measure these attributes and assess the model quality.

Many approaches try to identify these attributes based on counting measures applied to the original artifacts. This has two downsides:

(a) Measurements are not comparable between metamodels.

(b) The measurement does not reflect the cognitive and computation effort necessary to process models.

Allen et al. (2007) introduced a information theory approach focusing the entropy of systems as the central measure for models. We present an extensible analysis tool realizing this measure and support a wide variety of input models.

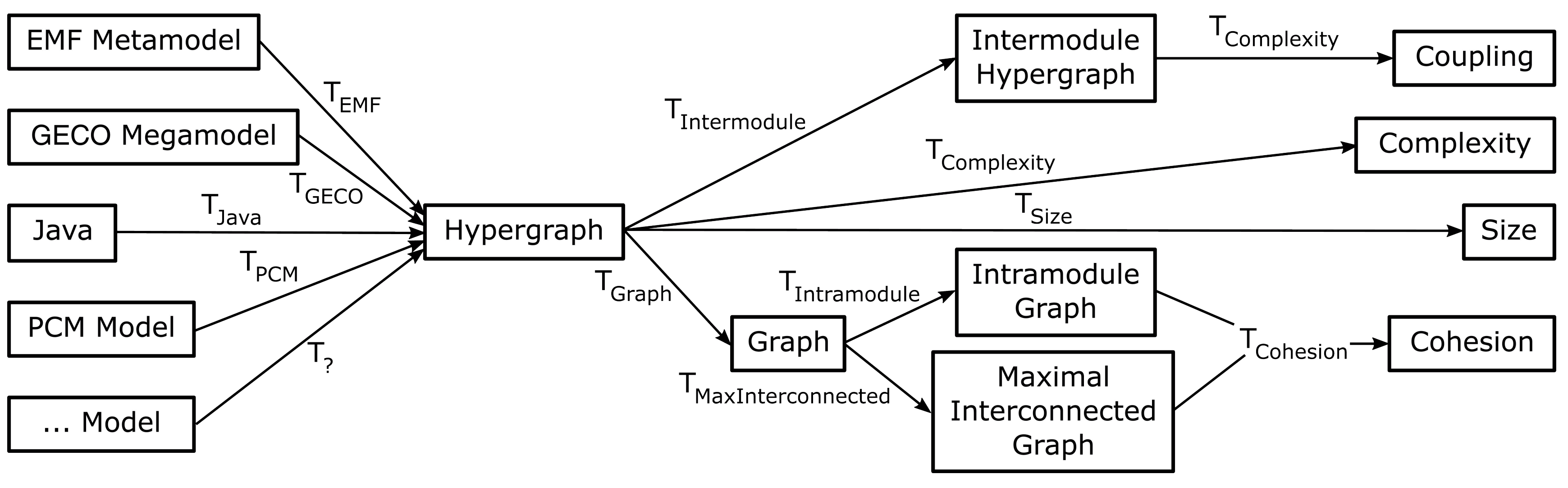
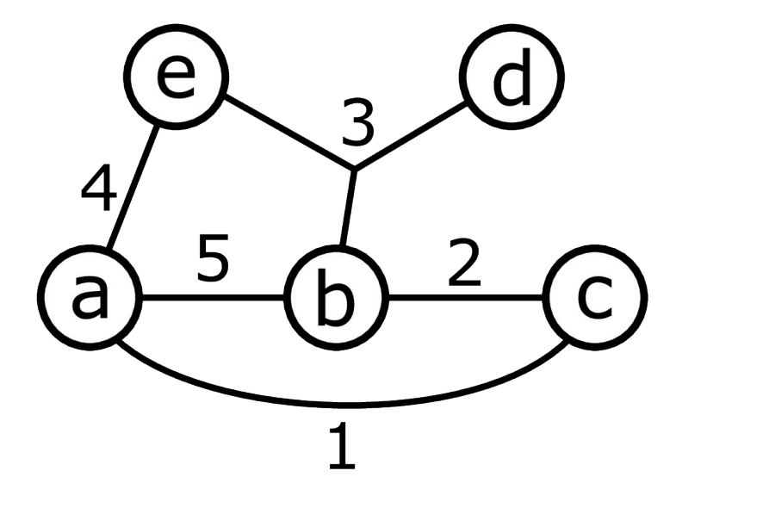


Figure 1 Megamodel of the analysis framework with input models for Java, EMF, PCM and other model types

Foundation

The analysis tool implements entropy-based measures operating on a hypergraph for size, coupling, and complexity, and relies on a derived-graph for cohesion (see Figure 1).



$$Size(\mathbf{S}) = \sum_{i=1}^{n} (-\log_2\ p_{L(i)})$$

Size(S) = 9.6094 bit

Node	Hyperedges 1 2 3 4 5	p_l
a	1 0 0 1 1	1/5
b	0 1 1 0 1	1/5
c	1 1 0 0 0	1/5
d	0 0 1 0 0	2/5
e	0 0 1 0 0	

The computation emanates from the nodes and the hyperedges they are connected to. For each node a pattern is computed containing 1s and 0s to represent connected and not connected hyperedges, respectively. For example, the sequence 01101 of a node implies its connection to hyperedge 2, 3 and 5, but not to 1 and 4. This results in as many pattern as there are nodes, making the probability of each pattern 1/n with n being the number of nodes. However, it is possible that multiple nodes have the same pattern, resulting in higher probabilities for those patterns. The function p_l returns the probability of the l-th hyperedge pattern for the metric of Allen et al. (2007). The function L(i) returns the pattern index corresponding to the i-th node.

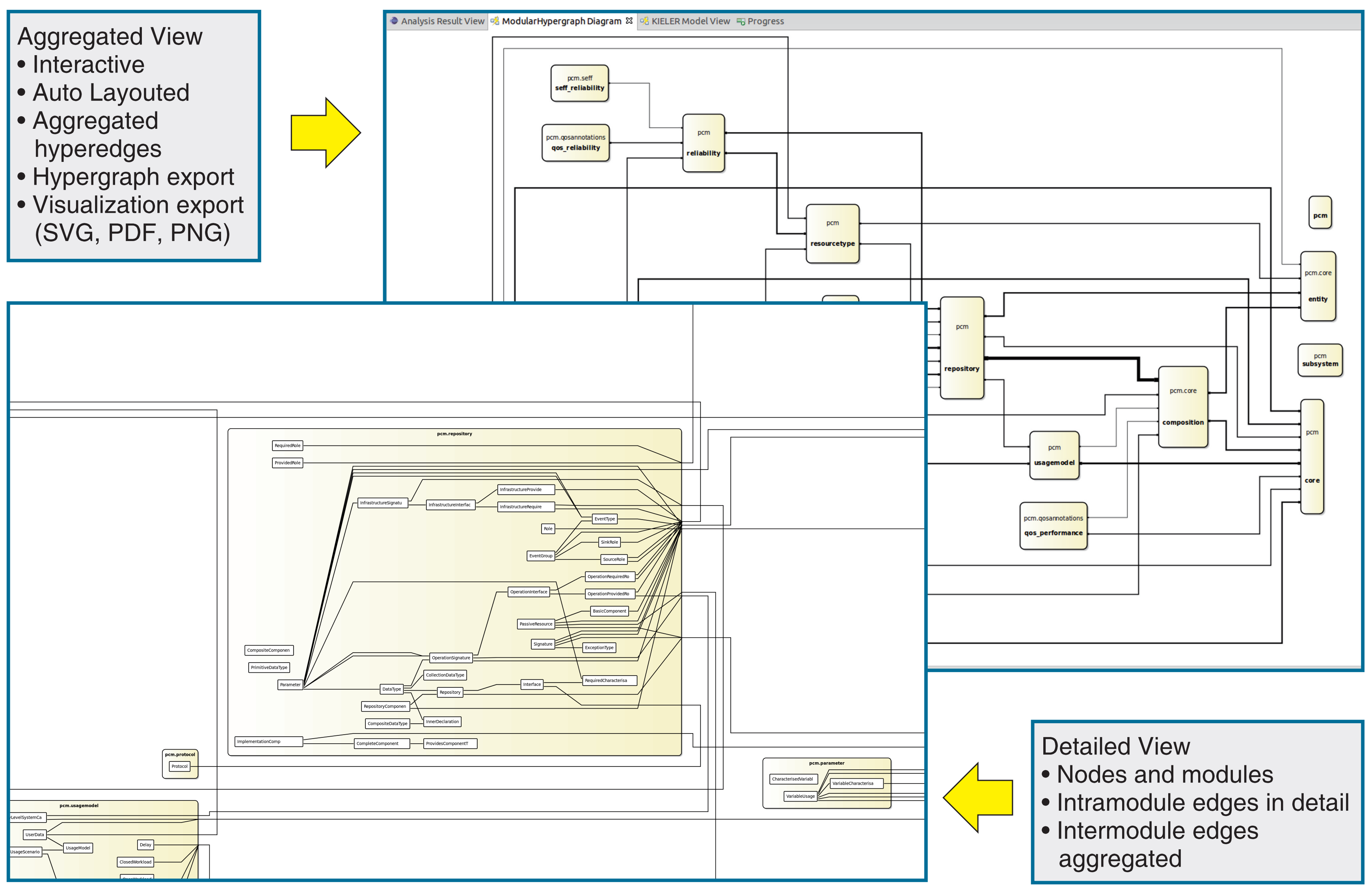


Figure 2 Visualization of the hypergraph generated for the Palladio Component Model, depicted in the aggregated view and an exceprt of the detailed view

Extensible Framework

The framework allows to add transformations for any number of input, illustrated in Figure 1 through $T_?$. Provided transformations are:

- EMF metamodel analysis
- GECO megamodel analysis
- Configurable system analysis for Java up to version 8 (configuration of system boundary and data types)
- PCM model analysis prototype for behavior models

Measuring & Visualization

The analysis tool provides three different visualizations for hypergraphs and a view for computed results:

- Aggregated hyperedges view for modular hypergraph
- Modular hypergraph view depicting all modules and nodes
- Plain hypergraph view with nodes and hyperedges
- Computation result view size, complexity, coupling, and cohesion

Example Results

	Modules	Nodes	Edges	Size	Complexity	Coupling
PCM	20	147	283	983.73 bit	578.79 bit	186.87 bit
Kieker A.	311	956	1169	8019.73 bit	2623.10 bit	2056.35 bit

References

E. B. Allen, S. Gottipati, and R. Govindarajan. Measuring size, complexity, and coupling of hypergraph abstractions of software: An information-theory approach. Software Quality Journal, 15(2):179–212, 2007.

Robert Heinrich, Eric Schmieders, Reiner Jung, Kiana Rostami, Andreas Metzger, Wilhelm Hasselbring, Ralf H. Reussner, Klaus Pohl: Integrating Runtime Observations and Design Component Models for Cloud System Analysis. MoDELS@Run.time 2014: 41-46

This work was supported by the DFG (German Research Foundation) under the priority program SPP 1593: Design For Future - Managed Software Evolution (grants HA 2038/4-1, RE 1674/7-1) and the Helmholtz Association of German Research Centers.