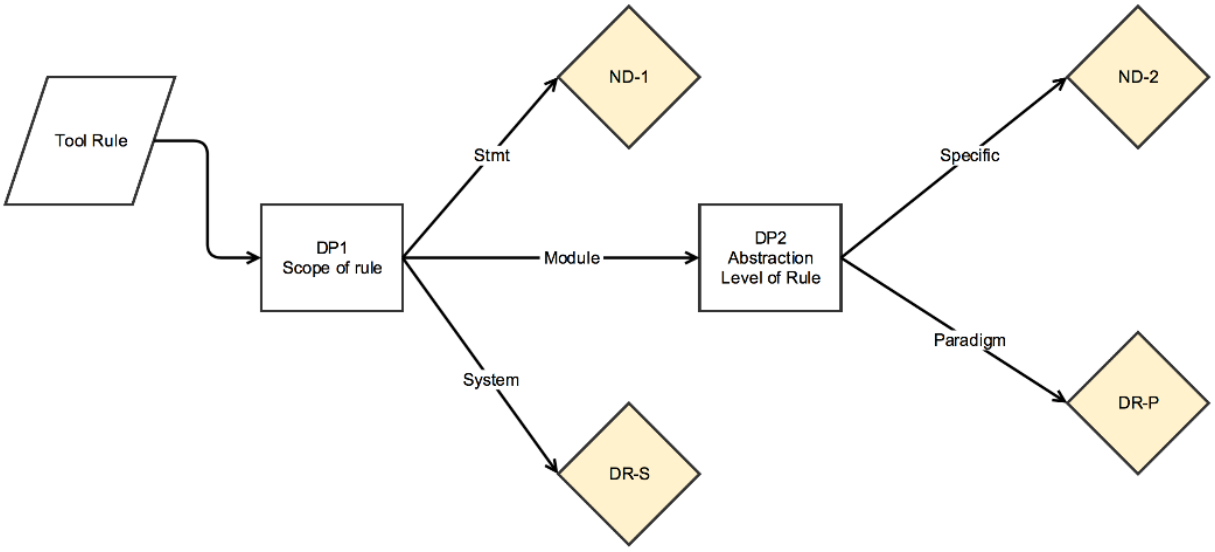
Software Engineering Institute
Carnegie Mellon University

DESIGN RULE CLASSIFICATION RUBRIC AND GUIDANCE

Neil A. Ernst, Stephany Bellomo, Ipek Ozkaya, and Robert L. Nord

April 2017

Design Rule Classification Rubric



Design Rule Classification Guidance

Quality attribute type is an orthogonal classification and not relevant to this rubric.

DP1: Scope

- **Statement (ND1)**
  - Almost always code problems and not design.
  - Not about accumulation of multiple violations of the rule.
  - Internal to method (switch, case, if/else, expression)
  - Empty methods, dead stores, fit here.
- **Module (go to DP2)**
  - A class, component, Module, object.
  - The lowest level of “design” we might draw on paper.

  - A group of statements (file) or a language construct (method, class) that can be executed independently, reused, tested, and maintained or is a composition of other modules.
- **System (DR-S)**
  - Problems detected cross boundaries between languages and/or architectural layers. For instance, it involves both the application code and the data access layer (CAST)
  - Metric thresholds (e.g., McCabe complexity > X, Component Balance < 7) typically fall here.
  - Allocation view specific details (e.g., package/file naming violations) may belong here.

DP2: Abstraction Level

- **Specific (ND2)**
  - Module level, syntax specific violations.
  - We cannot translate into another language or paradigm easily.
  - Might see keywords or reserved words or concepts, but can't be translated to generalizable concept.
  - E.g., Violating Spring naming conventions is a rule with no obvious commonalities in other frameworks.
- **Paradigm (DR-P)**
  - Covers a paradigm (OO, Functional, Imperative, ...), Architectural Style (Concurrent, Pipe/Filter, PubSub, MVC, ...), or Design Pattern (Exception Handling, ...).

Additional Resources

This material supplements the research paper Bellomo, S.; Nord, R.; Ozkaya, I.; & Popeck, M. “What to Fix? Distinguishing Between Design and Non-Design Rules in Automated Tools,” in *Proceedings of the IEEE International Conference on Software Architecture (ICSA 2017)*.

This study is part of a wider SEI effort on technical debt.

Contact Us

Software Engineering Institute
4500 Fifth Avenue, Pittsburgh, PA 15213-2612

Phone: 412/268.5800 | 888.201.4479
Web: www.sei.cmu.edu | www.cert.org
Email: info@sei.cmu.edu

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

DM-0004376