

Some observations on the application of software metrics to UML models



Jacqueline McQuillan and James Power
Principles of Programming Research Group
Department of Computer Science
National University of Ireland
Maynooth

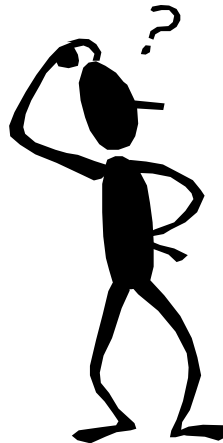
<http://www.cs.nuim.ie/research/pop/>

Outline

- Motivation
- Three categories of challenges for model metrics
 - technical
 - conceptual
 - practical
- Our research
- Summary

1 Motivation

- Object-oriented software metrics
- Many based on analysis of source code
- Not always clear how to apply them at the early stages of the software development process



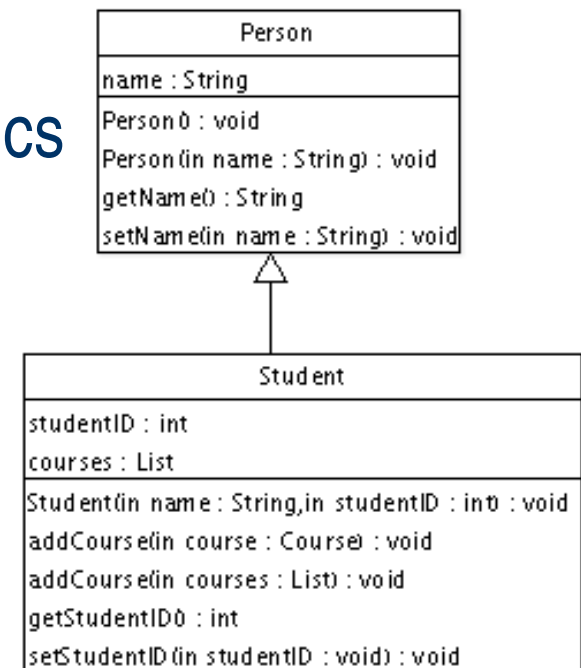
- Important to be able to measure these metrics from both models and source code
 - assessing quality at early stages of the software life-cycle
 - identify parts of the implementation that deviate from its design
 - evaluation of round-trip engineering tools

2.1 Technical Challenge

- Defining metrics
 - Reusing metrics
 - Comparing metrics
- 

Observation 1. Defining model metrics is a metamodeling activity

- Difficult to interpret and understand the exact definition of software metrics
- Difficult to compare and evaluate metrics
- Model elements being measured
- Define metrics in terms of this model
- Metrics are defined on the metamodel



Number of Methods? 9 5

 3 4


Observation 7. Metric definitions should be re-usable

- Measure metrics at both design (UML) and implementation (Java) level
- Define certain concepts once and adapt to each level
 - e.g. DIT, NOC
- Same concepts are being measured at each level

Observation 6. Differences between metric values are themselves metrics

- Design models and implementation are not synchronised
 - reflect properties of the evolution of the system
- Design models and implementation are synchronised
 - model level metrics could be used to specify boundary values for the implementation
 - differences could capture the level of additional complexity added by the implementation process
 - could also have use in reverse engineering

2.2 Conceptual Challenge

- Model metrics
 - Metrics from from partial descriptions of models
- 


Observation 3. Defining metrics is (almost regrettably) easy

- Easy to define metrics
- Difficult to correlate with external quality attributes
- Many metrics count things
- New set of elements that can be counted
- Avoid trap of proposing metrics that count these elements without offering evidence that such counts are useful

Observation 5. Models can represent partial and/or overlapping information

- Measuring a single metric from a UML model
- UML diagrams represent a view of a system rather than an overview
- Many different diagrams can contain the same elements
 - a class may appear in a number of different class diagrams
 - a sequence diagram provides details of a usage scenario
- Need to integrate these different sources of information before metric calculation

2.3 Practical Challenge

- Gathering metric values
 - Comparing metric values
 - Interpreting metric values
- 

Observation 4. We can “lift” code metrics to the model level

- Metrics at implementation level can be measured from models
- Need to consider all diagrams in UML model (and OCL)
- UML 2.0 provides 13 different diagrams
- Challenge of measuring a single metric from *all* diagrams
- Chidamber and Kemerer metrics suite
 - Baroni et al. use class diagrams
 - Tang and Chen use class, activity and collaboration diagrams

Chidamber and Kemerer Metrics

	<i>WMC₁</i>	<i>WMC_{cc}</i>	<i>DIT</i>	<i>NOC</i>	<i>CBO</i>	<i>RFC</i>	<i>LCOM</i>
Class diagram	✓	X	✓	✓	P	P	P
Component diagram	X	X	X	X	X	X	X
Composite Structure diagram	X	X	X	X	P	X	X
Deployment diagram	X	X	X	X	X	X	X
Object diagram	X	X	X	X	P	X	X
Package diagram	X	X	X	X	X	X	X
Activity diagram	X	P	X	X	P	P	P
Communication diagram	X	P	X	X	P	P	P
Interaction diagram	X	P	X	X	P	P	P
Sequence diagram	X	P	X	X	P	P	P
State Machine diagram	X	P	X	X	P	P	P
Timing diagram	X	X	X	X	X	X	X
Use Case diagram	X	X	X	X	X	X	X

Observation 9. Standardisation is multi-faceted

- Aspects for the comparison and evaluation of metrics
 - Benchmark suites
 - important in software engineering
 - select programs and models for use in metric studies
 - Data sets
 - *PROMISE Software Engineering Repository*
 - Non-code artifacts
 - lack of UML models and other design artifacts
 - many open source projects
 - require design artifacts to be made available

3 Our Research

- Define reusable metrics at meta-level
 - OCL and language metamodels
 - single metamodel and transformations, transform definitions
- Evaluate various Java metamodels
 - Dagstuhl Middle Metamodel, Netbeans, Kollmann, jnome
- Explore differences between metrics at different levels of abstraction

4 Summary

- Several observations relating to model metrics
- Partition these into three categories of challenges
 - technical
 - conceptual
 - practical
- We are concerned with the technical challenges of defining reusable metrics at the meta-level

Thank You!

- URL: <http://www.cs.nuim.ie/~jmcq>
- Thanks to
 - James Power, CS Department, NUI Maynooth
- Any questions?