SDMetrics

The Software Design Metrics tool for the UML™
UML Design Quality Measurement

- Calculate design metrics for UML designs, e.g.,
  - Coupling between components
  - Size of packages
  - Complexity of classes
  - Etc.

- Check UML design rules
  - Detect incomplete, incorrect, redundant, or inconsistent design
  - Find style problems such as circular dependencies, violation of naming conventions, and more
Design Properties: Coupling

- The degree to which an element in the design is connected to other elements.

- Distinction: import coupling and export coupling
  - Aka afferent/efferent coupling, fan-in/fan-out

- Impact of high coupling on system quality
  - Decreased maintainability
  - Decreased testability
  - Decreased reliability
  - Decreased reusability
Example Coupling Metrics

• Class coupling metrics
  – Outgoing/incoming associations or dependencies
  – Class used as parameter type of an operation
  – Messages sent/received by instances of the class

• Package coupling metrics
  – Associations or dependencies from or to classes/interfaces outside the package

• Component coupling metrics
  – Required interfaces, provided interfaces
Design Properties: Size

- Size of a design element: the number or size of elements it contains.
  - Number of attributes, operations in a class
  - Number of classes, interfaces, in a package
  - Number of states in a state machine
  - ...

- Good indicators of effort

- Large elements indicate poor design
  - E.g. “God” or “Blob” classes
Design Properties: Complexity

- The connectivity between the elements of a design unit ("count of the edges in a graph")
  - Number of method calls between lifelines in a sequence/communication diagram
  - Number of transitions in a state chart
  - Number of object/control flows in an activity diagram

- Impact of high complexity on system quality
  - Decreased understandability, testability
  - Increased fault-proneness
Further Examples of Metrics

- **Class inheritance metrics**
  - Depth of the class in the inheritance hierarchy
  - Number of children or descendent classes

- **The “Martin Metrics” for packages**
  - Abstraction (A), instability (I), distance from main sequence (D)

- **Metrics for use cases**
  - Included or extended use cases
  - Size of a use case in terms of the size of its sequence/activity diagrams
Design Rule Checking (1)

- Completeness
  - Unused or unreachable elements
  - Underspecified elements: unnamed, missing type, ...

- Correctness
  - Well-formedness rules of the UML
  - E.g., circular inheritance, fork state does not target orthogonal states, ...
Design Rule Checking (2)

• Style
  – Circular dependencies between classes/packages
  – Controversial design practices, e.g.:
    • Multiple inheritance
    • Use of association classes, n-ary associations
  – Big classes, long parameter lists

• Naming
  – Conventions for capitalization, prefixes, ...
  – Use of programming language keywords, ...
Benefits of Design Measurement

- Identify potential design problems early on
- Better focus of review and testing efforts
- Increase system quality and quality assurance effectiveness
- Refine your LOC or effort estimates for implementation, testing, maintenance
From Model to Metrics with XMI (1)

UML Tool
(RSM, MagicDraw, Together Designer, Poseidon, ...)

UML Model as XMI File

SDMetrics
From Model to Metrics with XMI (2)

- **XMI**: XML Metadata Interchange
  - XML-based representation of a UML Model
  - Standardized by OMG; key technology of the MDA
  - Supported by all major UML modeling tools
  - Different versions of XMI: 1.0/1.1/1.2/2.0/2.1

- SDMetrics supports all versions of XMI
  - Works with all modeling tools with XMI export
  - Meta models for UML 1.x and UML 2.x
# Metric Analysis: Table View

![Metric Analysis Table View](image)

## Metric Data Tables

<table>
<thead>
<tr>
<th>Name</th>
<th>NumAttr</th>
<th>NumOps</th>
<th>NumPubOps</th>
<th>Setters</th>
<th>Getters</th>
<th>Nesting</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang.AbstractMethodError</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>java.lang.IncompatibleClassChangeError</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>java.lang.String.CaseInsensitiveComparator</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>9</td>
<td>64</td>
<td>59</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>java.lang.ArithmeticException</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>java.lang.RuntimeException</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>java.lang.ArrayIndexOutOfBoundsException</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>java.lang.IndexOutOfBoundsException</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>java.lang.ArrayStoreException</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>java.lang.Boolean</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>java.lang.Object</td>
<td>0</td>
<td>13</td>
<td>10</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>java.lang.Class.1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>java.lang.Class</td>
<td>4</td>
<td>53</td>
<td>35</td>
<td>2</td>
<td>41</td>
<td>0</td>
</tr>
<tr>
<td>java.lang.Byte</td>
<td>5</td>
<td>19</td>
<td>19</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>java.lang.Number</td>
<td>1</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>java.lang.Character.UnicodeBlock</td>
<td>68</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>java.lang.Character.Subset</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Metric Analysis: Element View
# Design Rule Checking

## Rule Checker

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Name</th>
<th>Rule</th>
<th>Value</th>
<th>Category</th>
<th>Severity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang.String</td>
<td>GodClass</td>
<td>#ops/attr: 73</td>
<td>Style</td>
<td>2-med</td>
<td>The class has more than 60 attributes</td>
<td></td>
</tr>
<tr>
<td>java.lang.String</td>
<td>DupOps</td>
<td>String&lt;31,31,31,31,...&gt;</td>
<td>Correctness</td>
<td>1-high</td>
<td>Class has duplicate operations.</td>
<td></td>
</tr>
<tr>
<td>java.lang.ArithmeticException</td>
<td>DepCycle</td>
<td>cyc#1 (11 nodes)</td>
<td>Style</td>
<td>2-med</td>
<td>The class has circular references.</td>
<td></td>
</tr>
<tr>
<td>java.lang.ArrayIndexOutOfBoundsException</td>
<td>Unused</td>
<td></td>
<td>Completeness</td>
<td>1-high</td>
<td>The class is not used anywhere.</td>
<td></td>
</tr>
<tr>
<td>java.lang.ArrayStoreException</td>
<td>Unused</td>
<td></td>
<td>Completeness</td>
<td>1-high</td>
<td>The class is not used anywhere.</td>
<td></td>
</tr>
<tr>
<td>java.lang.Boolean</td>
<td>Unused</td>
<td></td>
<td>Completeness</td>
<td>1-high</td>
<td>The class is not used anywhere.</td>
<td></td>
</tr>
<tr>
<td>java.lang.Byte</td>
<td>Unused</td>
<td></td>
<td>Completeness</td>
<td>1-high</td>
<td>The class is not used anywhere.</td>
<td></td>
</tr>
<tr>
<td>java.lang.Character</td>
<td>Unused</td>
<td></td>
<td>Completeness</td>
<td>1-high</td>
<td>The class is not used anywhere.</td>
<td></td>
</tr>
<tr>
<td>java.lang.Character, UnicodeBlock</td>
<td>GodClass</td>
<td>#ops/attr: 70</td>
<td>Style</td>
<td>2-med</td>
<td>The class is not used anywhere.</td>
<td></td>
</tr>
<tr>
<td>java.lang.Character, UnicodeBlock</td>
<td>Unused</td>
<td></td>
<td>Completeness</td>
<td>1-high</td>
<td>The class is not used anywhere.</td>
<td></td>
</tr>
<tr>
<td>java.lang.Byte</td>
<td>Unused</td>
<td></td>
<td>Completeness</td>
<td>1-high</td>
<td>The class is not used anywhere.</td>
<td></td>
</tr>
<tr>
<td>java.lang.character, UnicodeBlock</td>
<td>GodClass</td>
<td>#ops/attr: 73</td>
<td>Style</td>
<td>2-med</td>
<td>The class is not used anywhere.</td>
<td></td>
</tr>
<tr>
<td>java.lang.reflect.Field</td>
<td>DepCycle</td>
<td>cyc#1 (11 nodes)</td>
<td>Style</td>
<td>2-med</td>
<td>The class has circular references.</td>
<td></td>
</tr>
<tr>
<td>java.lang.reflect.Method</td>
<td>DepCycle</td>
<td>cyc#1 (11 nodes)</td>
<td>Style</td>
<td>2-med</td>
<td>The class has circular references.</td>
<td></td>
</tr>
<tr>
<td>java.lang.reflect.Constructor</td>
<td>DepCycle</td>
<td>cyc#1 (11 nodes)</td>
<td>Style</td>
<td>2-med</td>
<td>The class has circular references.</td>
<td></td>
</tr>
<tr>
<td>java.lang.reflect.Array</td>
<td>Unused</td>
<td></td>
<td>Completeness</td>
<td>1-high</td>
<td>The class is not used anywhere.</td>
<td></td>
</tr>
<tr>
<td>java.lang.reflect.InvocationTargetException</td>
<td>AttrNameOvr</td>
<td>serialVersionUID</td>
<td>Naming</td>
<td>2-med</td>
<td>The class defines an attribute of the same name as another in a super or related class.</td>
<td></td>
</tr>
<tr>
<td>java.lang.reflect.InvokeTargetException</td>
<td>Unused</td>
<td></td>
<td>Completeness</td>
<td>1-high</td>
<td>The class is not used anywhere.</td>
<td></td>
</tr>
</tbody>
</table>
## Design Comparisons

### Metric Deltas

<table>
<thead>
<tr>
<th>Name</th>
<th>NumAttr</th>
<th>NumOps</th>
<th>NumPubOps</th>
<th>Setters</th>
<th>Getters</th>
<th>Nesting</th>
<th>IFImpl</th>
<th>NOC</th>
<th>Nur</th>
</tr>
</thead>
<tbody>
<tr>
<td>model.xmi.sdmetrics.app.DesignComparator</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.DiagHistogram</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.DiagKiviat.KiviatDiagAdapter</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.DiagKiviat</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.DiagPanel</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.ElementPoolAdapter</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.ElementTreeNode</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.ExportFormatter</td>
<td>0</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.ExportFormatterFactory</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.GraphExporter.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.GraphExporter.GraphList</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.GraphExporter.FileNotOverwrittenE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.GraphExporter</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.MeasurementCatalog measurme</td>
<td>5</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.MeasurementCatalog.MetricTable</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model.xmi.sdmetrics.app.MeasurementCatalog.RuleTable</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Relation Matrices

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.AbstractMethodError</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.IncompatibleClassChangeError</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.String</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.String</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.ArithmeticException</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.RuntimeException</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.ArrayIndexOutOfBoundsException</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.IndexOutOfBoundsException</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.ArrayStoreException</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.Boolean</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.Class</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.Character</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.CharSequence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.reflect.Field</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.java.lang.reflect.Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data Export

• Detailed metric data analysis with statistical software packages or spreadsheets
• Export of data tables
  – Tab or comma-separated text files
  – HTML
  – XML for Microsoft Excel
  – OpenOffice.org Calc
• Export of graphs
  – SVG, PNG, JPG
Summary of SDMetrics' Features

- Over 120 design metrics, 130 design rules
  - Cover all diagram types of the UML
  - Users can define new metrics and rules
- Works with all UML tools with XMI export
  - Customizable XMI import
- Batch processing via command line interface
- Fast execution
  - Analyses large designs with hundreds of thousands of model elements within seconds
Customizing SDMetrics

User-defined UML meta models

User-defined XMI import

User-defined metrics and rules
SDMetrics Project Files

UML Modeling Tool

XMI Source File

SDMetrics

Metric Data Output

XMI Transformation File

Metamodel Definition File

Metrics Definition File
Meta Model Definition File

- Defines the model elements that SDMetrics knows:
  - Meta classes
  - Attributes
  - Relationships

- Defined as far as is needed for metric calculation and design rule checking
  - Not a complete 1:1 representation of the standard UML meta models
  - Default meta models for UML1.3/1.4 and UML2.x
<modelelement name="class">
  <attribute name="visibility"
    type="data" multiplicity="one"/>
  <attribute name="abstract" type="data"/>
  <attribute name="ownedattributes"
    type="ref" multiplicity="many"/>
  <attribute name="ownedoperations"
    type="ref" multiplicity="many"/>
  ...
</modelelement>
XMI Transformation File (1)

- How to retrieve information of the model elements and attributes from XMI files
- Defines a mapping:

  UML meta model as manifested in the XMI file

  SDMetrics' simplified meta model
Excerpt from an XMI File

<ownedMember xmi:type='UML:Class' xmi:id='id345' visibility='public' isAbstract='false' name='MyClass'>
    <ownedAttribute xmi:id='id1138' name="myAttr" visibility='private' type='id42'/>
    <ownedOperation xmi:id='id1139' name='myOperation' visibility='public'>
        <ownedParameter xmi:id='1140' name='par1' type='id123' />
    </ownedOperation>
    ...
</ownedMember>
XMI Transformation File Sample

```xml
<xmitransformation modelelemnet="class"
    xmipattern="UML:Class" recurse="true">
  <trigger name="visibility"
    type="attrval" attr="visibility"/>
  <trigger name="abstract" type="attrval"
    attr="isAbstract"/>
  <trigger name="ownedattributes"
    type="xmi2assoc" src="UML:Property"
    attr="ownedAttribute" />
  <trigger name="ownedoperations"
    type="xmi2assoc" src="UML:Operation"
    attr="ownedOperation" />

  ...
```
XMI Transformation File (2)

• UML model exchange via XMI is difficult in practice
  – Different versions of the XMI
  – Implementations not fully compliant with standards
  ➔ Customizable XMI import to account for this!

• Default XMI transformation files:
  – For UML1.3/1.4 meta model and XMI 1.x files
  – For UML2.x meta model and XMI 2.0 / 2.1 files

• Plus specialized files for non-conforming tools
Metric Definition File

- Contains the definitions of:
  - Design metrics
  - Design rules
  - Relation matrices
  - Complete documentation with glossary and literature references

- SDMetrics provides default metric definitions for UML1.x and UML2.x meta models
Example Metric Definition

<metric name="NumPubOps" domain="class" category="Size">
  <description>The number of public operations in a class.((p))
Same as metric metric://class/NumOps/, but only counts operations with public visibility.
((ul))((li))Also known as: NPM (Number of Public Methods) ref://LK94/.((/ul))</description>
  <projection relset="ownedoperations" condition="visibility='public'"/>
</metric>
Example Metric Definition (2)

- Definitions of intermediate “helper” sets and metrics
  - E.g. sets “InAssoc” and “OutAssoc” of incoming and outgoing associations for an interface

```xml
<metric name="Assoc" domain="interface" category="Coupling">
  <description>The number of associations the interface participates in.</description>
  <compoundmetric term="size(InAssoc+OutAssoc)" />
</metric>
```
Example Rule Definition

<rule name="OutgoingAssoc" severity="low" domain="interface" category="Style" >
  <description>The interface has outgoing associations. (p)
  ref://Oes04/ suggests to avoid this.
  </description>
  <violation condition="size(OutAssoc)!\=0"
    value="'#Assoc: '+size(OutAssoc)" />
</rule>
Applications of Design Metrics

Building Quality Benchmarks

Building Predictive Models
Quality Benchmarks (1)

- Scenario: acquisition of new components
  - Developed in-house or bought from external source
- Is the design of the components acceptable?
  - Demand rework if there are expected maintainability/reliability problems
- Comprehensive inspection/testing often too expensive
  - Need a pointer to potentially critical design areas to inspect or test
Quality Benchmarks (2)

- Build a database of measurement values
  - Selected measures of size, coupling, complexity
  - Obtained from existing proven, “tried and tested” components
- Apply measurements to new components
- Compare obtained measurement distributions
Quality Benchmarks (3)

![Bar chart showing the number of classes (%) for different numbers of invoked operations. The x-axis represents the number of invoked operations (0 to 8), and the y-axis represents the number of classes (%). There are two bars for each x-value, one for Benchmark and one for Candidate. The chart shows that the number of classes decreases as the number of invoked operations increases.]
Prediction Models

• Design metrics data difficult to interpret
  – component XYZ has “coupling=5”, “complexity=7”

• What does this mean in terms of …
  – future maintenance work for this component
  – likelihood that the component still contains defects

• Quantify the relationship between metrics data and system quality for better interpretation
Building A Prediction Model

- Component Design
- Design Metrics
- Fault Data
- SDMetrics
- Inspection
- Statistic Analysis
- Prediction Model
Using the Prediction Model

New Component Design

SDMetrics

Design Metrics

Prediction Model

Component OK

P(fault)>25%

P(fault)<25%

Inspection
Prediction Model Pros and Cons

- **Pros**
  - Captures the combined effect of multiple influencing factors in one cohesive model
  - Translates non-interpretable internal quality data to easily interpreted external quality data

- **Cons**
  - Very data intensive
  - Requires statistical expertise