# Code Metrics

<table>
<thead>
<tr>
<th>class</th>
<th>Cyclic</th>
<th>Dcy</th>
<th>Dcy*</th>
<th>Dpt</th>
<th>Dpt*</th>
</tr>
</thead>
<tbody>
<tr>
<td>rx.operators.OperationOnExceptionResumeNextViaObservableTest.TestObserve...</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>rx.operators.OperationTimeout.Timeout</td>
<td>216</td>
<td>14</td>
<td>1610</td>
<td>1</td>
<td>398</td>
</tr>
<tr>
<td>rx.operators.OperationSingle</td>
<td>216</td>
<td>10</td>
<td>1610</td>
<td>1</td>
<td>398</td>
</tr>
<tr>
<td>rx.operators.OperationSkip.Skip.ItemObserver</td>
<td>216</td>
<td>3</td>
<td>1610</td>
<td>1</td>
<td>398</td>
</tr>
<tr>
<td>rx.operators.OperationTakeUntil.OtherObservable</td>
<td>216</td>
<td>8</td>
<td>1610</td>
<td>1</td>
<td>398</td>
</tr>
<tr>
<td>rx.operators.OperationTimeout.TimeoutSelector</td>
<td>216</td>
<td>13</td>
<td>1610</td>
<td>1</td>
<td>398</td>
</tr>
<tr>
<td>rx.schedulers.TestScheduler.CompareActionsByTime</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>33</td>
</tr>
<tr>
<td>rx.schedulers.TestScheduler.InnerTestScheduler</td>
<td>1</td>
<td>5</td>
<td>13</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>rx.android.observables.AndroidObservable</td>
<td>0</td>
<td>4</td>
<td>1631</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>rx.observers.SynchronizedObserver.Test.CompletionThread</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>rx.operators.OperationConditionals</td>
<td>0</td>
<td>12</td>
<td>1625</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>rx.operators.OperationSkipWhile.SkipWhile.SkipWhileObserver</td>
<td>216</td>
<td>4</td>
<td>1610</td>
<td>1</td>
<td>398</td>
</tr>
<tr>
<td>rx.operators.OperationSample.SampleWithObservable.ResultManager.Sampler</td>
<td>216</td>
<td>3</td>
<td>1610</td>
<td>1</td>
<td>398</td>
</tr>
<tr>
<td>rx.apache.http.consumers.ResponseConsumerBasic</td>
<td>0</td>
<td>8</td>
<td>1615</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

---

**SWEN-261**  
Introduction to Software Engineering  
Department of Software Engineering  
Rochester Institute of Technology
A metric is not just a number.

- A metric is a quantitative function that calculates some characteristic and produces a numeric measurement which will be used to make a decision.

- For software product development, metrics fall into three broad categories
  - *Process* – measurements of the software process that apply across projects
  - *Project* – measurements of one project team's activities
  - *Product* – measurements of the resulting software product
Software product metrics fall into multiple categories that look at different characteristics.

- **Complexity**
  - *Lines of Code is the most familiar*
  - *Cyclomatic Complexity*

- **Coupling and Dependency**
  - *Robert Martin Package Metrics*

- Counting/averaging lots of things that can be counted/measured
  - *Average lines per method*
  - *Average parameters per method*
  - *Average number of methods per class*

- Some metrics will apply at multiple levels, such as project, package, class, or method
Even though you can count something, it does not necessarily count for anything.

- A metric is only as good as the decisions that it will be used to make.
- Measuring something without it having a solid connection to possibly improving what you are doing is a waste of time and resources.
- Target values for measurements should be set based on a record of past measurements and resulting performance.
  - *Measurement not in some range ➔ some project quality was poorer*
- Initially, measurements need to be made to find the correlations.
A metric target is not absolute.

- A measurement falling outside of a target range is not an absolute indictment.
- Measurements that do not fall in the target range indicate a place for additional scrutiny.
  - *For product metrics, they indicate possible "code smells".*
  - *Places to consider for refactoring, redesign, or reimplementaton*
These are some of the more popular metrics for object-oriented software systems.

- **Cyclomatic complexity**
  - *Count of execution paths through a method*

- **Chidamber and Kemerer**
  - *Coupling between object classes*
  - *Lack of cohesion in methods*

- **Martin Package Metrics**
  - *Fan-out coupling – classes need something outside package*
  - *Fan-in coupling – classes outside package use something inside package*
  - *Instability – ratio of fan-out to fan-out + fan-in*
  - *Abstractness – ratio of abstract classes and interfaces to total number in package*