
XS – A Measure of Structural Over-Complexity

White Paper

January, 2006

www.headwaysoftware.com

Contents

Executive Summary	3
Measuring Fat	3
Function-level Fat	3
Higher-level Fat	4
Measuring Tangles	5
Normalizing XS	7

Executive Summary

“Structure” is the way in which the thousands of lines of code that comprise a code-base are organized into higher-level abstractions (functions, classes, files, packages, jars, etc.), and the dependencies that emerge through this organization.

A large code-base is necessarily complex. However it is possible to keep this complexity within defined thresholds at any point of compositional breakout.

[Structure101](#) uses a single metric called XS (“excess”) to measure the degree to which any item of composition (e.g. a package, class or method) exceeds the defined complexity threshold. XS reflects the size of a code item so that, for example, an overly complex high level package will have higher XS than a single method. This reflects the likely negative impact on development processes.

A single measure means that it is possible to drill down on a code-base to discover areas of over-complexity. It also helps to set priorities for simplification.

XS is measured for every code item in a compositional hierarchy based on the graph of the immediate contents of the item.

There are 2 components of XS – Fat and Tangles.

Fat measures the amount of basic complexity with an item. Tangles are applied to higher (design, package) levels and measures cyclic dependency.

Measuring Fat

The objective of the Fat measure is to limit the amount of “stuff” at any level of compositional break-out to a reasonable “mind-sized chunk”.

Function-level Fat

Structure101 uses [cyclomatic complexity](#) (CC), also known as “McCabe’s Metric”, as the measure of Fat for subprograms (functions, methods, etc.)

CC is computed using a graph that describes the control flow of the subprogram. It is the number of possible execution paths.

CC is considered a more useful measure of complexity than, for example, the number of statements in a program. It more closely reflects the intellectual effort required to understand and modify the code, since programs with complex, nested control structures will have

a higher CC than methods with the same number of simple, sequential statements.

Higher-level Fat

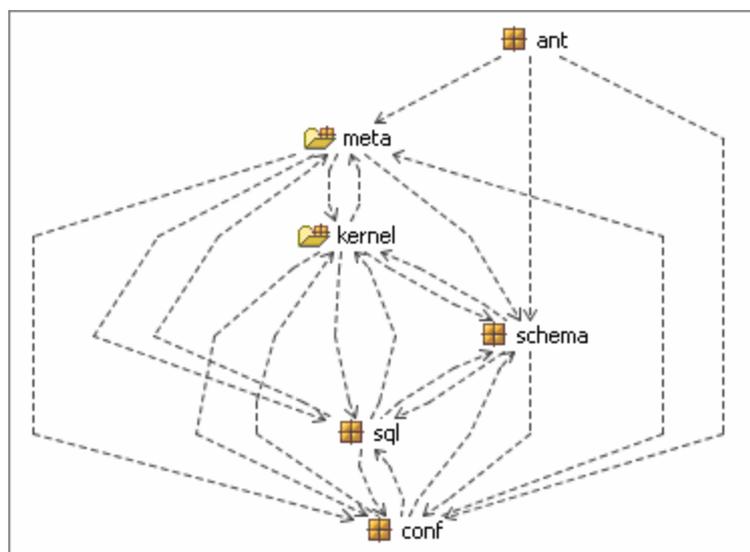
Structure101 measures complexity at compositional levels above the function level by extending the principle used by cyclomatic complexity.

Just as a function contains statements, a class contains methods, a package contains classes and a high-level package contains child packages, and it is the complexity of this breakout we wish to measure and control.

Again, the relationships between sub-items better reflects the intellectual effort to understand and modify an item than does the count of sub-items.

Structure101 therefore uses the number of dependencies in the dependency graph of sub-items as a measure of the complexity of an item.

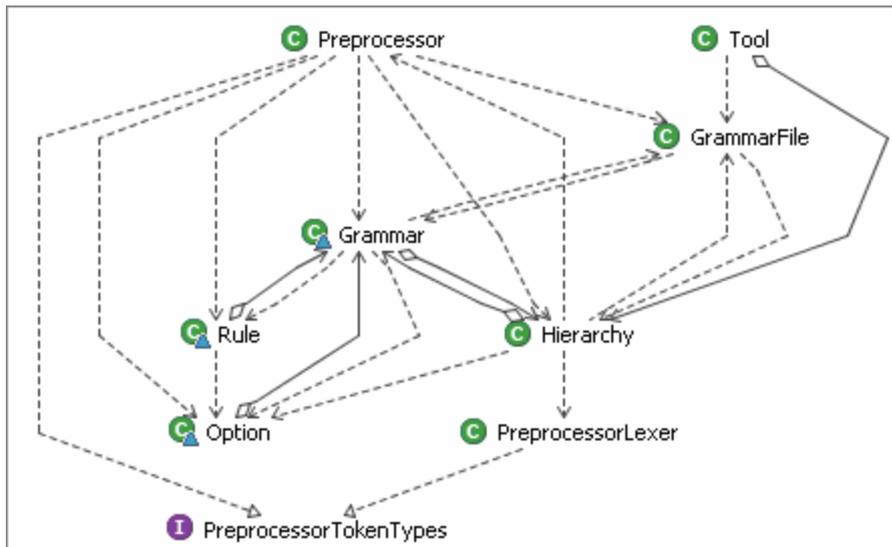
For example, the following is the sub-item dependency diagram for a high-level package with a complexity of 22:



Dependency graph for a high-level package

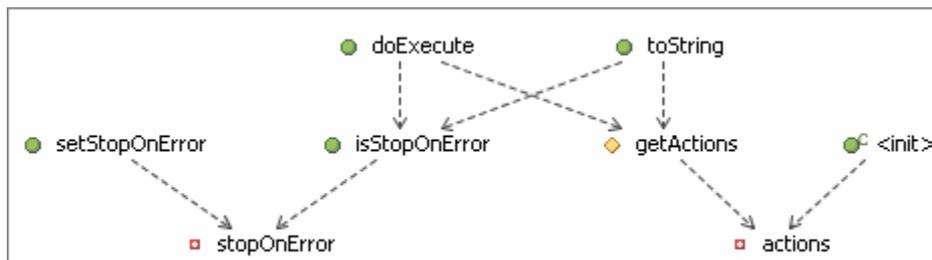
The complexity for a low-level package (one that contains only classes) is similarly the number of dependency edges between the classes it contains. For example, this is the graph of a package with a complexity of 23:

XS – A Measure of Over-complexity



Dependency graph for a leaf package

Finally, the complexity of a class is calculated from the dependency graph of its methods and fields. The following is the dependency graph for a class with a complexity of 8:



Dependency graph for a class

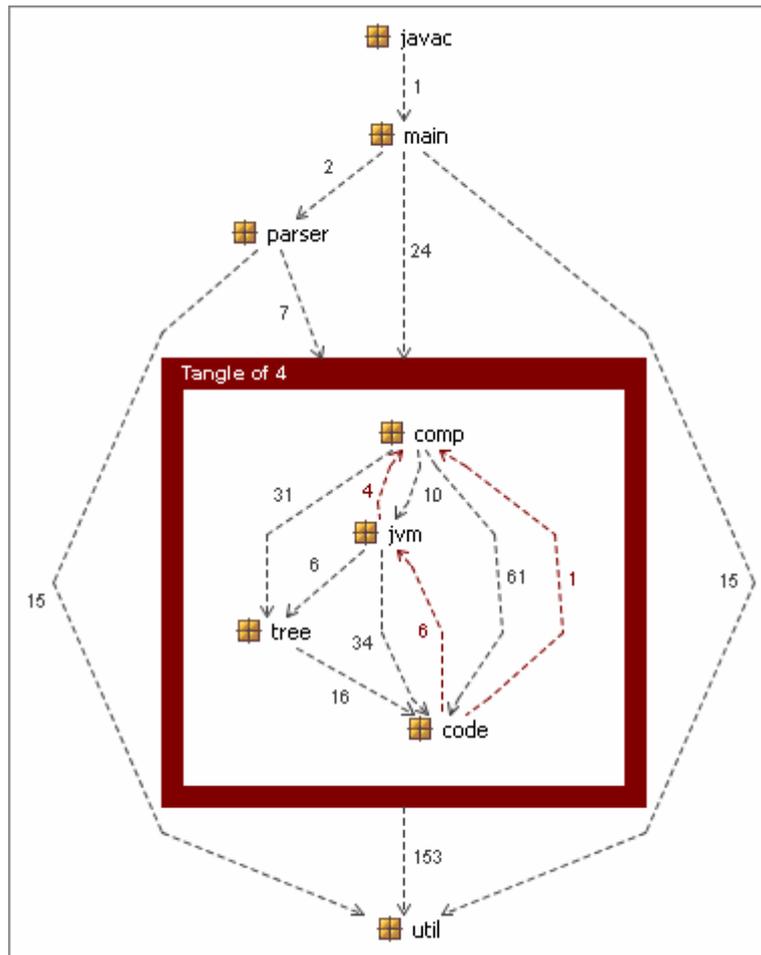
In each case, Fat is the degree to which an item exceeds the specified complexity threshold (see “normalization” below).

Measuring Tangles

“Tangles” are a different kind of structural complexity caused by cyclic dependencies at the higher levels of decomposition (design, packages).

These warrant separate measurement as they can make a code-base very hard to understand, modify, extend, test and deploy (see Bob Martin).

The following diagram shows a tangled package-level dependency graph:



Tangled Package dependency graph

This is tangled because, for example, packages “comp” and “jvm” are mutually dependant.

A tangle is a set of packages for which there is a path from every package to every other package in the dependency graph.

In the diagram above, structure101 has “auto-partitioned” the dependency graph to isolate the tangled packages.

Structure101 analyses the dependency graph and highlights the **Minimum Feedback Set (MFS)**. This is the minimum set of dependencies that would need to be removed in order to make the graph **acyclic** (no longer tangled). The edges in the MFS are colored red – there are 3 such dependencies in the example above.

With the MFS identified, structure101 calculates the **degree** of the tangle as the number of dependencies in the MFS divided by the total number of dependencies in the graph (usually expressed as a percentage).

The number of **code-level references** is used for both the identification of the MFS and the calculation of the degree. This number is shown on each edge of the graph. For example, there are 2 code-level

references from the code contained in package "main" to code contained in package "parser".

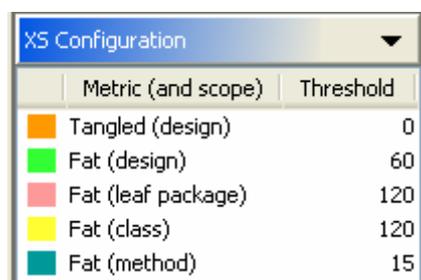
In the example then, the degree of the tangle is $6+4+1 / 386 = 2.8\%$.

Normalizing XS

The key concept in controlling structural complexity is that of *excessive* complexity.

It can be reasonably argued that all software systems are complex – complexity is an inherent attribute of software. However, individual items need never exceed certain complexity thresholds. The degree to which an item *exceeds* these thresholds is its XS.

The first step, therefore, is to define thresholds for Fat and Tangles. Structure101 lets us define thresholds for different levels or *scopes*:



Metric (and scope)	Threshold
Tangled (design)	0
Fat (design)	60
Fat (leaf package)	120
Fat (class)	120
Fat (method)	15

Example thresholds

"Design" refers to packages that contain other packages. "Leaf package" refers to packages that contain only classes. Structure101 measures XS using the "Package Hierarchy¹", in which packages are always either "design" or "leaf package".

As already mentioned, Tangles are considered a problem between packages only. Design-scope dependency graphs should be acyclic, so the threshold for Tangled (Design) is usually set to 0.

The thresholds for Fat are less clear cut. The default values shown above are taken from various studies and experience, and can be adjusted to company or project norms.

The Tangle metric is already normalized to a value between 0 and 1.

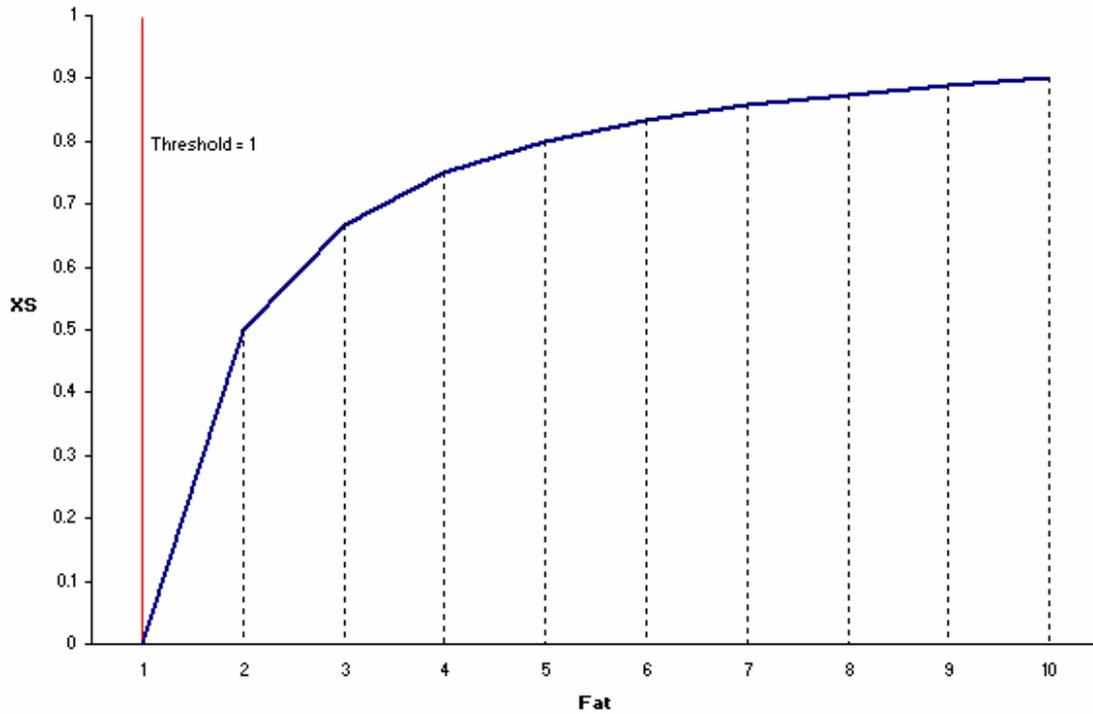
The following formula is used to normalize the Fat metric:

$$\text{Max} \left(\frac{\text{Value} - \text{Threshold}}{\text{Value}}, 0 \right)$$

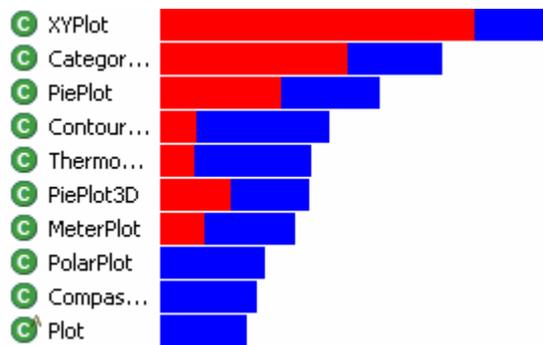
This converts the Fat value to XS according to the following curve (note that the threshold in this example is 1, so $2 = 2 \times \text{threshold}$, $3 = 3 \times \text{threshold}$, etc.).

¹ For more details, see the Structure101 help.

XS – A Measure of Over-complexity



With the values for Fat and Tangles normalized, it is possible to combine them for specific items, and to average them across projects or subsystems. Structure101 averages the XS values and associates them with the size so that the distribution of both can be understood at a glance, and areas of high complexity discovered by drill-down.



Size and XS Distribution Chart