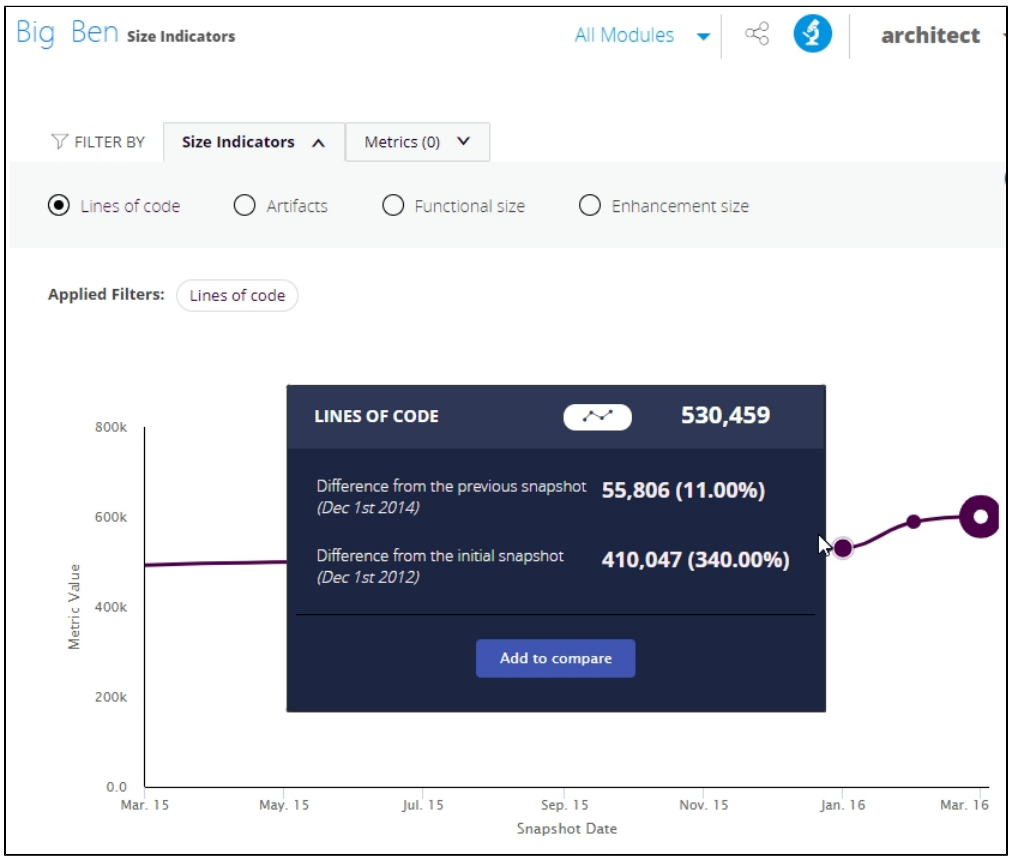
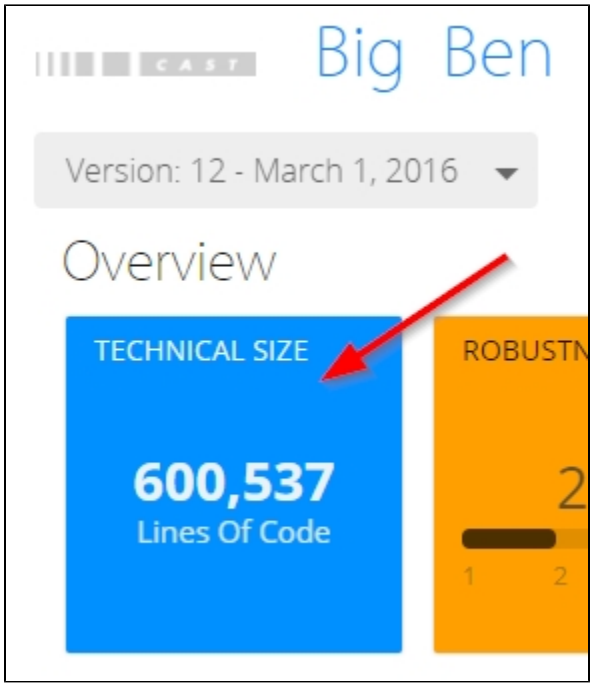
Technical Size metrics

AIP Core provides a number of **Technical Size metrics** which are designed to measure the size of specific items such as **code lines**, **comment lines** or **specific artifact types**. These metrics are also used by other structural rules provided by CAST. Some Technical Size result data is provided in the Health Dashboard:





A list of available Technical Size metrics is shown below - where no information is provided in the Comment column this is simply because the metric is self-explanatory.

ID	Metric name	Comment

19192	Number of ABAP Transactions	-
19191	Number of ABAP User-Exits	-
10152	Number of Artifacts	-
10155	Number of Classes	-

10151	Number of Code Lines	Number of Code Lines (LOC) is measured as the number of source code lines excluding comments and blank/empty lines. The algorithm used by CAST is as follows for all object types (For container type objects, the algorithm includes lines of code of children objects) identified by CAST:
		
		• BEFORE pre-processing source code
		• IGNORE binary files and 3rd party files included in application
		• FOR EACH OBJECT
		  • REMOVING blank lines (i.e., only LF + CR or LF or CR)
		  • REMOVING comments (can be single line comments or multi line comments)
		  • COUNT remaining lines of code
		
		More generally, there are two contexts for LOC measurement.
		
		• First context is Technical Sizing. In this context, source files are taken into account. More specifically, objects that are analyzed by CAST whose type inherit from the "APM Sources" category.
		• Second context is Quality Measurement. In this context, Source Files, Artifacts, and specifically targeted object types are taken into account.
		
		It is used as a basis for several major features:
		
		• LOC is used to validate analysis boundaries
		• LOC is also used to compare sizes between two different versions of source code
		• Some quality rules relly directly on this value:
		  • E.g. Avoid XXX too many Lines of Code, etc.
		  • Commented-out Code Lines/Code Lines ratio (% of LOC) ensures the absence of too much commented-out code lines within the comment lines
		  • Complexity Volume (% of LoC) ensures that the volume of complex code is kept low enough
		  • Copy Pasted Code (% of LOC) ensures that the volume of duplicated code is kept low enough
		
		ⓘ See also **Core Metrics** which is an extension that provides an alternative Code Lines counting mechanism.

10107	Number of Comment Lines	Number of Comment Lines is measured as the number of all comment lines including mono-line and multi-line comments.
10109	Number of Commented-out Code Lines	Number of Commented-out Code Lines is measured as the number of all comment lines that contain source code only. Please note that unless the following criteria are met by the source code, then the metric will be disabled: • the source code must contain more than 20 "clean" comments • the source code must contain at least 80 artifacts that have more than 10 lines of code What is a "clean" comment? • if the comment contains more than 4 words, then there must be less than 40% OOVs • if the comment contains less than 5 words, then there must be a maximum of 1 OOV • a single word on its own must not represent 50% of the comment What is an "OOV"? An "OOV" is an "Out of Vocabulary Word". With regard to CAST, an OOV is a word that has never appeared in the code.
19173	Number of Copybooks	-
19174	Number of Datablocks	-
19169	Number of Datawindows	-
10167	Number of Events	-

10154	Number of Files	-
10157	Number of Forms	-
19181	Number of Function Pools	-
10162	Number of Functions	-
19175	Number of Functions and Procedures	-
19180	Number of Includes	-
10160	Number of Interfaces	-
19168	Number of Macros	-
10161	Number of Methods	-
19177	Number of Modules	-
19176	Number of Namespaces	-
10166	Number of Packages	-
19172	Number of Paragraphs	-
19182	Number of Processing Screens	-
10156	Number of Programs	-
19171	Number of Sections	-
10158	Number of SQL Artifacts	-
10163	Number of Tables	-
19184	Number of Template Class Instances	-
19183	Number of Template Classes	-
19190	Number of Template Function Instances	-
19189	Number of Template Functions	-
19186	Number of Template Interface Instances	-

19185	Number of Template Interfaces	-
19188	Number of Template Method Instances	-
19187	Number of Template Methods	-
10165	Number of Triggers	-
19178	Number of Units	-
19170	Number of Userobjects	-
10164	Number of Views	-
10159	Number of WEB Pages	-