21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015) IOP Publishing
Journal of Physics: Conference Series **664** (2015) 062053 doi:10.1088/1742-6596/664/6/062053

First statistical analysis of Geant4 quality software metrics

Elisabetta Ronchieri[1], **Maria Grazia Pia**[2], **Francesco Giacomini**[1]

[1] INFN CNAF, Bologna 40126, Italy
[2] INFN, Sezione di Genova, Genova 16146, Italy

E-mail: `elisabetta.ronchieri@cnaf.infn.it`

Abstract.
Geant4 is a simulation system of particle transport through matter, widely used in several experimental areas from high energy physics and nuclear experiments to medical studies. Some of its applications may involve critical use cases; therefore they would benefit from an objective assessment of the software quality of Geant4. In this paper, we provide a first statistical evaluation of software metrics data related to a set of Geant4 physics packages. The analysis aims at identifying risks for Geant4 maintainability, which would benefit from being addressed at an early stage. The findings of this pilot study set the grounds for further extensions of the analysis to the whole of Geant4 and to other high energy physics software systems.

1. Introduction

Geant4 [1, 2] is a simulation system that is used in a wide variety of scientific contexts, including critical applications. As a mature (20 years old) software system, it is an ideal playground to study metrics and metrics tools addressing the maintainability of large scale high energy physics software systems over the range of decades.

The issue of software maintainability is especially relevant for such a widely used, mature software system. To evaluate the maintainability of Geant4 software, we used existing standards, such as ISO/IEC 25010:2011 [3], which identifies the relevant software characteristics, and we exploited a set of product metrics - aggregated in the program size, code distribution, control flow complexity and object-orientation metrics categories - which allow appraising the code state. By using various software metrics tools (such as Imagix4D [4] and Understand [5]), we were able to collect a large amount of measurements that characterize the software.

This paper reviews the adopted methodologies to perform this research and documents a series of release measurements. The analysis uses the RStudio development environment [6] for R [7]. In doing this, there is the intent to identify potential risks and provide the Geant4 maintenance team with useful information.

The remainder of this paper is structured as follows. Section 2 details the research methodology. Section 3 describes the data analysis methodology, whose results are shown in Section 4. Finally Section 5 presents conclusions detailing future work.

2. Research Methodology

The following steps describe the methodology we used to perform this research:

Step 1 established software quality standard, ISO/IEC 25010:2011 (former ISO/IEC 9126) [3], were used to identify software characteristics related to the maintainability factor;

Step 2 software metrics tools were identified and evaluated to collect a large amount of measurements of software characteristics;

Step 3 a set of product metrics were exploited to assess the code state.

2.1. Step 1: Software quality standard

There are many views of software quality. The IEEE defines quality as "the degree to which a system, component, or process meets specified requirements or customer or user needs or expectations" [8]. The International Organization for Standardization (ISO) [9] defines quality as "the degree to which a set of inherent characteristics fulfils requirements". Other experts define quality based on conformance to requirements and fitness for use. However, a good definition must lead us to measure quality meaningfully.

According to Fenton and Pfleeger [10], "measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules". A possible way to measure software is to use software metrics. The IEEE defines software metrics as "the quantitative measure of the degree to which a system, component or process possesses a given software attribute" [8] related to quality factors (also told characteristics). On the one hand, measurement allows us to know if the quality of the software improves over time, to know how process quality affects the product, to determine the quality of the current product or process, to predict qualities of a product or process. On the other hand, software metrics allow us to estimate the cost and schedule of future projects, to evaluate the productivity impacts of new tools and techniques, to establish productivity trends over time, to improve software quality, to anticipate and reduce future maintenance needs.

Software quality standards describe software quality models categorizing software quality into a set of characteristics. About software attributes, the ISO/IEC 25010:2011 standard defines six software factors, each subdivided in sub-characteristics (or criteria) [3]. The quality factors include: functionality, reliability, usability, efficiency, maintainability, portability. In the study reported in this paper we consider maintainability. The concern of maintainability is anything that helps with identifying the cause of a fault and then fixing the fault. Maintainability is affected by code readability or complexity, as well as by modularization.

2.2. Step 2: Software metrics tools

Several commercial and free tools are available, which calculate a variety of software metrics. Some software metrics are not univocally defined: therefore, they favour different interpretations and include various implementations in metrics tools that produce non-comparable values. Consequently, it is essential to consider more than one of these tools to gather as many measurements as possible and evaluate their role in scientific software.

Since Geant4 is written in C++, we chose tools that mainly support that language. A survey of tools used in this analysis is reported in [11]; with respect to this initial list, for the analysis reported in this paper we replaced non-supported tools, such as CCCC (C and C++ Code Counter) [12], Pmccabe [13] and Unified CodeCount [14], with new ones, such as Imagix 4D and SourceMonitor [15]. CLOC v. 1.60 counts blank lines, comment lines and physical lines of source code. SLOCCount v. 2.26 computes Source Lines of Code. Understand v. 3.1.278 is a static analysis tool. Imagix 4D v. 8.0.4 analyzes, documents and improves complex, third party or legacy C, C++ and Java software. SourceMonitor v. 3.5.0.306 is a source code metrics measurement tool.

2.3. Step 3: Software product metrics

We grouped the metrics according to file, class and function with respect to the type of information provided by the used software metrics tools. For each group, metrics are then categorized by size [16], complexity (such as McCabe [17] and Halstead [18]), object-orientation (from Chidamber and Kemerer (CK) [19]). Tables 1, 2 and 3 show a set of metrics that belong to the various groups. Furthermore, we also took into account metrics that can be useful to determine ratio information, such as Number of Files NF_{File} and Number of Classes NC_{File}.

Table 1. Some metrics of the size group

Group	Size Metric	Source
File	Comment Ratio (CR_{File})	Lorenz
	Declarations in File (NOD_{File})	Lorenz
	File Size (bytes)	Lorentz
	Functions in File (NOF_{File})	Lorenz
	Lines in File ($TLOC_{File}$)	Lorenz
	Lines of Source Code ($SLOC_{File}$)	Lorenz
	Lines of Comments ($CLOC_{File}$)	Lorenz
	Number of Statements (NOS_{File})	Lorenz
	Variables in File (NOV_{File})	Lorenz
Function (F)	Lines in Function ($TLOC_F$)	Lorenz
	Lines of Source Code ($SLOC_F$)	Lorenz
	Variables in Function (NOV_F)	Lorenz

Comment Ratio represents the ratio of the lines of comments to the lines of source code in the file.

NOD is the number of top-level declarations in the file, including types, variables, functions and macro defines.

The definitions of the other metrics are evident.

Table 2. Some metrics of the object-oriented group

Group	Object-Oriented Metric	Source
Class	Class Cohesion (LCOM)	CK
	Class Coupling (CBO)	CK
	Depth of Inheritance (DIT)	CK
	Number of Children (NOC)	CK
	Response for Class (RFC)	CK
	Weighted Methods (WMC)	CK

Lack of Cohesion of Methods (LCOM) is a measure of the cohesion of the member functions of the class.

Coupling Between Object (CBO) measures the coupling, or dependency, of the class.

DIT measures the depth of the hierarchy of base classes of the class.

NOC provides the number of classes directly derived from class.

RFC measures the number of methods called by the class methods.

WMC provides the total cyclomatic complexity for the class methods.

2.4. Preliminary scope of the analysis

Initial appraisals [11] concern a subset of Geant4 packages with a key role in scientific applications:

The Geometry package makes it possible to describe a geometrical structure and to navigate through it. In turn, it includes a set of sub-packages such as biasing, divisions, magnetic field, management, navigation, solids and volumes. Any simulation application involves some geometrical modelling of the experimental configuration.

Table 3. Some metrics of the complexity group

Group	Complexity Metric	Source	
File,	Intelligent Content (HI)	Halstead	HI measures the amount of content (complexity) of the file/function/class.
Function,	Mental Effort (HE)	Halstead	HE measures the number of elemental mental discriminations necessary to create, or understand, the file/function/class.
Class	Program Volume (HV)	Halstead	HV measures the information content of the file/function/class.
	Program Difficulty (HD)	Halstead	HD measures how compactly the file/function/class implements its algorithms.
File,	Average Cyclomatic Complexity (MACC)	McCabe	MACC, MMCC and MTCC measures the average, maximum and total cyclomatic complexity for all methods in file/class.
Class	Maximum Cyclomatic Complexity (MMCC)	McCabe	
	Total Cyclomatic Complexity (MTCC)	McCabe	
File	Maintainability Index (MI)	Welker	MI measures the maintainability of the file [20], incorporating source code metrics into a single number.
Function	McCabe Cyclomatic Complexity (v(G))	McCabe	McCabe v(G) represents the number of decision points in the function.
	McCabe Decision Density	McCabe	McCabe Decision Density measures decision density (cyclomatic density) of a function.
	McCabe Essential Complexity (ev(G))	McCabe	McCabe ev(G) represents the number of decision points in the function which contain unstructured constructs.
	McCabe Essential Density	McCabe	McCabe Essential Density measures essential density, or degree of unstructuredness, of a function.

The Processes package handles particle interactions with matter. Like the geometry package, it comprehends a set of sub-packages such as biasing, cuts, decay, electromagnetic, hadronic, management, optical, parameterisation, scoring and transportation. Electromagnetic physics represents the core of particle transport, as almost any simulation scenario involves electromagnetic interactions either as primary or secondary particles.

The PhysicsLists package contains selections of physics processes and modelling options.

3. Data Analysis Methodology

The following steps describe the methodology we used to perform data analysis:

Step 1 consolidate data;

Step 2 use descriptive statistics for each release to get the distribution (mean and median), variance (standard deviation) and quantiles of each measure;

Step 3 adopt correlations between metrics to eliminate metrics that do not provide additional insights;

Step 4 identify thresholds from metrics analysis.

Step 1 is the most important one. Data it not always presented in one table, but it is in several files that: need to be merged; may include information that is not necessary or unclear; exist on different format. Steps 2 and 3 allow us to identify a minimal, non-redundant set of metrics that is meaningful for our analysis. As a result, it is possible to identify areas with potential design problems and establish quality benchmarks. Step 4 is the most challenging one. The literature details thresholds of software goodness that are derived from specific domains, such as aerospace, telecommunication and student exercises. However, these studies are quite old and may not reflect the evolution of the programming languages; moreover, they may reflect domain-specific characteristics; therefore, they cannot be blindly applied to the high energy physics field.

Due to the large amount of data involved, only the first two steps are detailed in this paper to fit within the page limit constraints of these proceedings. The full information and related discussions will be documented in a dedicated journal publication.

4. A Sample of Analysis Results

We applied our methodology to a fairly representative set of Geant4 software releases. This set includes 31 releases for Geometry and Processes, and 16 releases for PhysicsLists, which was first introduced in the Geant4 release 7. Furthermore, for the release that comprises at least one patch we only considered the last one, as shown in Table 4. In the following we will refer to each metric by using its acronym followed by a specific index with respect to the corresponding package, such as g for Geometry, p for Processes and pl for PhysicsLists.

Table 4. The Geant4 software releases

Number	Name	Year	Number	Name	Year
1	0.0.p04	1999	17	7.0.p01	2005
2	0.1	1999	18	7.1.p01	2005
3	1.0	1999	19	8.0.p01	2006
4	1.1	2000	20	8.1.p02	2006
5	2.0.p01	2000	21	8.2.p01	2007
6	3.0	2000	22	8.3.p02	2008
7	3.1	2001	23	9.0.p02	2008
8	3.2	2001	24	9.1.p03	2008
9	4.0.p02	2002	25	9.2.p04	2010
10	4.1.p01	2002	26	9.3.p02	2010
11	5.0.p01	2003	27	9.4.p04	2012
12	5.1.p01	2003	28	9.5.p02	2012
13	5.2.p02	2003	29	9.6.p04	2015
14	6.0.p01	2004	30	10.00.p04	2015
15	6.1	2004	31	10.01.p01	2015
16	6.2.p02	2004			

For each package we derived metrics considering the groups: file, class and function. At each group, we adopted the metrics detailed in Tables 1, 2 and 3 due to their use in determining the maintainability software characteristic [21]. In this paper, we report measurements mainly calculated with the Imagix 4D tool.

At this stage of the analysis, we did not apply any special treatment to outliers. This subject will be studied in depth in the next stage of the analysis.

A prevalent technique to visualize a distribution is to plot a histogram. However, it has several shortcomings: the choice of the bins affects the shape of the histogram, which may cause misinterpretations of data; the distributions of two metrics are difficult to compare when they have different sizes. To overcome these problems, an alternative way to clearly examine the evolution of data over release (i.e. time) is to use the box-plot [22]. In this case, the x-axis depicts groups of numerical data through their quantiles and the y-axis represents the metric values and some other statistical information.

The Geometry package has a number of source files in the range [501, 697] over release. The overall total amount of lines (TLOC) examined is near 5 million. CLOC $\in$ [32K, 59K] and SLOC $\in$ [50K, 116K] are the measurements obtained in the various releases. The comment ratio (CR_g) (see Figure 1) highlights that there not always exists a one-to-one correspondence between increasing lines of code and comment lines. Figure 1 shows vertical peaks in two directions - upwards for release numbers 8 and 14, while downwards for release numbers 5 and 30 - whose reasons are justified in the respective release notes. The maintainability index (MI_g) (see Figure 2) highlights the presence of anomalous files - those below the lower whiskers - which will require more careful examination.

Processes is a large package with a number of source files in the range [840, 3492] over release. The overall total amount of lines (TLOC) examined is 13 million. CLOC $\in$ [25K, 222K] and SLOC $\in$ [138K, 469K] are the measurements obtained in the various releases. This package exhibits the same behaviour of Geometry regarding the comment lines of code. Figure 3 shows vertical upwards peak for release number 8, whose reasons are justified in the respective release notes. Concerning MI_p (see Figure 4), this has a similar trend to that shown in Figure 2, but the large amount of code increases the number of anomalous files.

PhysicsLists is a small package with a number of source files in the range [203, 410] over release. The overall total amount of lines examined (TLOC) is 430 thousand. CLOC $\in$ [6K, 17K] and SLOC $\in$ [8K, 23K] are the measurements obtained in the various releases. Referring to CR_{pl}, Figure 5 shows vertical upwards peaks for release numbers 19 and 25, whose reasons are justified in the respective release notes. The maintainability index (MI_{pl}) (see Figure 6) highlights the presence of anomalous files, which will require further in-depth study.

Table 5 summarizes the prevalent object-oriented metrics at class group for the three packages. The reported ranges are for the maximum value determined from their distributions.

Table 5. The Geant4 object-oriented measurements with $NC_g \in [244,362]$, $NC_p \in [438,2046]$ and $NC_{pl} \in [127,264]$

Package	DIT	NOC	CBO	LCOM	RFC	WMC	MMCC
Geometry	3	[10,18]	[10,17]	[266,677]	[46,98]	[38,137]	[11,31]
Procesess	[2,3]	[30,66]	[23,52]	[435,2168]	[47,133]	[36,101]	[18,33]
PhysicsLists	1	[10,18]	[6,14]	[99,114]	[61,80]	[57,65]	[19,47]

5. Conclusions

With this work we aim to build a data set of measurements about Geant4 software in order to evaluate its quality.

In this study we considered the Geometry, PhysicsLists and Processes packages, which play a major role in a wide range of experimental applications. After selecting suitable software metrics tools, such as Imagix 4D and Understand, we used them to collect data. The use of descriptive statistics allowed us to perform an initial analysis on the various Geant4 software releases from 0.0 up to 10.01.p01 to identify the evolution of the software characteristics over time.

An analysis of the correlations among the various metrics is in progress with the goal to retain a sample of meaningful metrics. The analysis of the metrics will contribute to identifying parts of Geant4 that would benefit from close attention regarding future maintainability. Furthermore, we will work on the identification of appropriate thresholds and ranges associated with the risk of maintainability of the software. This study will take into account the peculiarities of high energy physics software, which may be different from the characteristics of other software domains for which such estimates are documented in the literature.

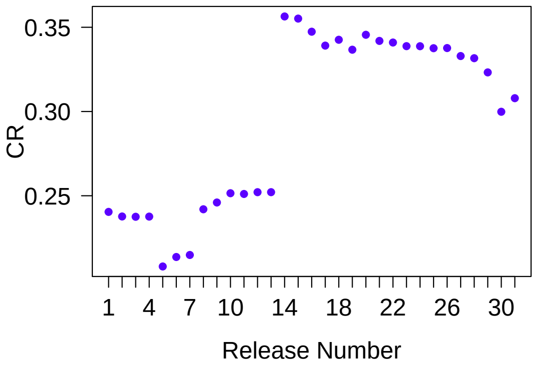

Figure 1. CR_g over release.

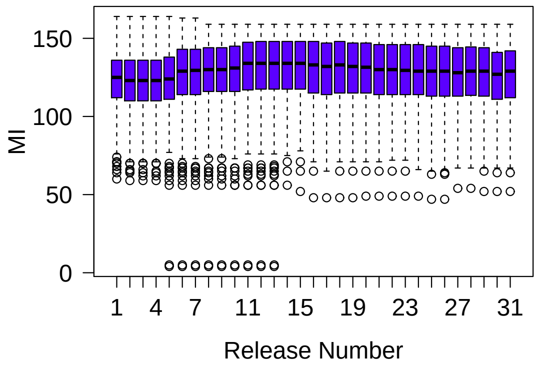

Figure 2. MI_g over release.

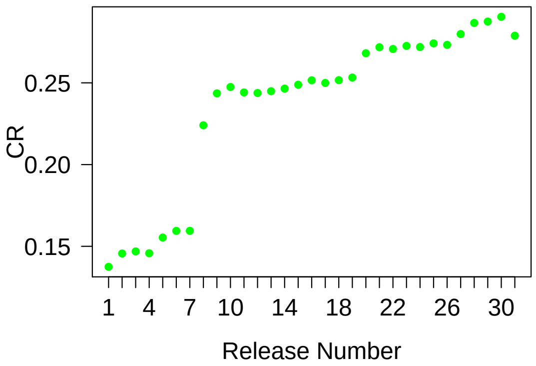

Figure 3. CR_p over release.

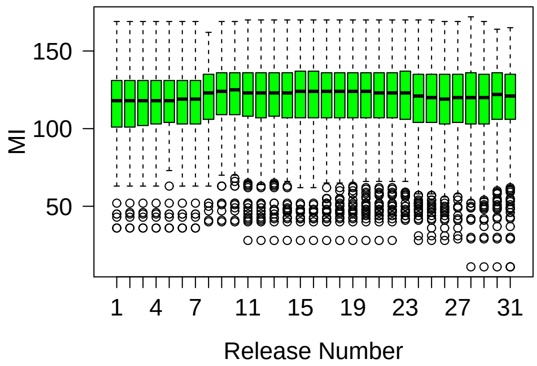

Figure 4. MI_p over release.

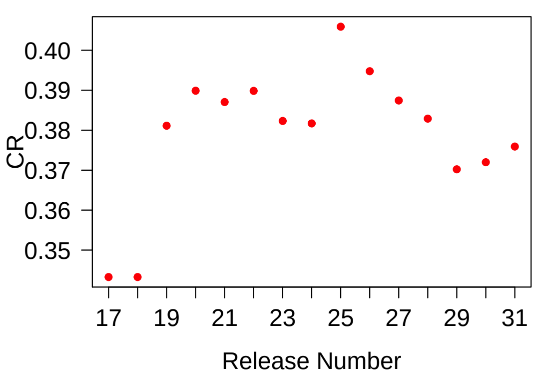

Figure 5. CR_{pl} over release.

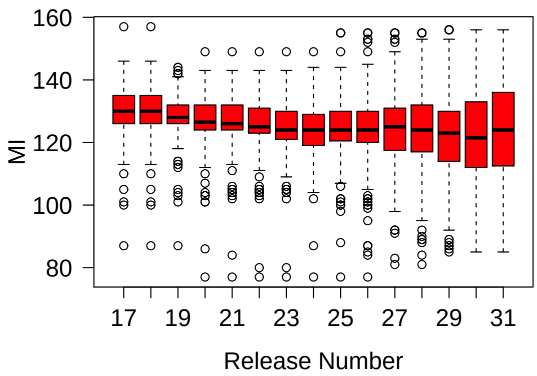

Figure 6. MI_{pl} over release.

Acknowledgements

The authors acknowledge Imagix Corporation that provided extended free full license for performing this work, and Scitool and their European partner Emenda (www.emenda.eu) that provided several evaluation licenses for relevant metrics. Furthermore, they are particularly grateful for the prompt support given by Gabriele Cosmo (CERN) during the research.

References

[1] Agostinelli S and et al 2003 *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* vol 506 n 3 p 250–303
[2] Allison J and et al 2006 *IEEE Trans. Nucl. Sci* vol 53 n 1 p 270–278
[3] ISO/IEC 25010:2011, `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733`
[4] Imagix 4D, `www.imagix.com`
[5] scitools Understand, `https://scitools.com`
[6] RStudio, `http://www.rstudio.com/products/rstudio/`
[7] R, "The R Project for Statistical Computing", `http://www.r-project.org/`
[8] IEEE, "610.12 - 1990 - IEEE Standard Glossary of Software Engineering Terminology", December 1990.
[9] ISO 9000 - Quality Management, `http://www.iso.org/iso/iso\_9000`
[10] Fenton N and Bieman J 2014 *Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series* (CRC Press) p 1–67
[11] Ronchieri E and Pia M G and Giacomini F 2014 *Proceedings of the 18th Topical Meeting of the Radiation Protection & Shielding Division of ANS, RPSD 2014* Knoxville Tennessee USA September 14-18.
[12] CCCC, `http://cccc.sourceforge.net/`
[13] Pmccabe, `http://people.debian.org/`
[14] Unified CodeCount, `http://sunset.usc.edu/research/CODECOUNT/`
[15] Campwood Software, SourceMonitor, `www.campwoodsw.com`
[16] Lorenz M and Kidd J 1994 *Object-Oriented Software Metrics: A Practical Guide* (Prentice-Hall)
[17] McCabe T J 1976 *IEEE Transaction on Software Engineering* vol SE-2 n 4
[18] Halstead M H 1977 *Elements of Software Science* (Elsevier, North-Holland Inc.)
[19] Chidamber S R and Kemerer C F 1994 *IEEE Transactions on Software Engineering* vol 20 n 6
[20] Welker K D 2001 *The Journal of Defense Software Engineering*
[21] Lincke R 2007 *Licentiate thesis MSI* (Vaxjo University)
[22] Weissgerber T L and Millic N M and Winham S J and Garovic V D 2015 *PLOS Biology*