Yevhenii Koshovets, Master student,

Igor Sikorsky Kyiv Polytechnic Institute, Kyiv

**SOFTWARE METRICS E-CATALOGUE**

A software metric is a way to measure different aspects of software development, such as how good the software is, how quickly it is being developed, and how complex it is. These measurements are used to help evaluate and improve the software development process, and to track progress over time. Metrics can be used at different stages of the software development life cycle to monitor progress, identify risks, and make decisions based on data rather than just intuition or guesswork. Choosing the right metrics for a specific project is important, and they should be used carefully, along with other factors, to make informed decisions about the software development process.

The biggest problem is that choosing metrics is difficult. There are many interpretations of them, and all of them are all over the Internet. We can solve this problem by creating a software metrics e-catalogue.

In current work, we will focus on creating a template itself and not on the filling in the catalogue.

There has been already done some work in this direction. Authors of publication [1] have proposed a template. They try to create a framework for collecting web metrics. The problem is that it is outdated. Field, like “automation type” with possible values “manual”, “automated” only proves my point.

Publication [2] is another example of metrics catalogue format. Authors try with their template to answer questions in [3], which should be sufficient to define metric. Despite the fact, that format is quite good it lacks practical value. Some of the fields are too complicated to understand. There are no classifications or tools mentioning that can help find needed metric faster.

In the end of the research the next format was defined:

* name. State the name of the metric which is described.
* Alias. Alternative names and short version of a metric which can be used.
* Goal. Short, definite answer of what is the goal of usage of such a metric.
* Consequences. Describes both good and bad consequences of not reaching and reaching of a metric standard.
* Entity. The entity to which metric is applied.
* Entity type. Can be : process, product or resource.
* Attribute. Is what we want to measure on entity the metric is applied.
* Attribute type. Can be internal or external.
* Software lifecycle phases. Should be one or several phases that are related to the metric.
* Metric type. Can be based (doesn't use other metrics during calculation) or derived (use other metrics to determine this one).
* Metric category and subcategory. Relates to the characteristics, which are mentioned in ISO/IEC 25010 [4].
* Measurement scale. Refer to the possible type scale measurement. Can be ratio, absolute, nominal, ordinal or interval.
* Unit of measurement. Mention a unit of measurement of the metric.
* Range. Describes a range of possible values.
* Critical points. Defines specific values or ranges that correspond to different levels of performance ("optimal", "good", "bad") and provide examples of how these levels are determined.
* Definition. Describes a formula or algorithm with the help of which metric is calculated. If the metric type is derived then apart from formula itself, should also contain a short mentioning of metrics used in the formula and links to them.
* Metrics where used. Contains links to the metrics for which current metric is used in the formula. For example, if current metric is LOC, then value of this field can be CC, as LOC is used in the formula to calculate CC.
* Tool. Consists of three subfields: name, type of automation, tags.
  + Name: contain the name of the existing tool that can measure current metric.
  + Type of automation: describes the way the tool is atomized: collecting information, scheduling of data collection, visualization of collected data.
  + Tags: contain list of tags(I.e. technology, programming language, ) that can define if this tool is applicable for your case and ease your search.
* Use cases. Specific use cases or scenarios where the metric is particularly useful.
* Limitations. Any known limitations or weaknesses of the metric, including potential pitfalls or areas where the metric may not be appropriate.
* Normalization. Standardize the metric by taking into account differences in scale or size of the entities being measured.
* Trend analysis. Analyzes how the metric changes over time and could be useful for identifying patterns and predicting future trends

References

1. Towards Automated Web Metrics Towards a Catalog Format for Software Metrics/ Luis Olsina [and oth.] // Conference: VIII Workshop Brasilero de Calidad. – 2001.
2. Towards a catalog format for software metrics / Eric Bouwers [and oth.], 2014.
3. Software engineering metrics: What do they measure and how do we know? / C. Kaner and W. Bond., 2004 – C. 4-6.
4. International Organization for Standardization. ISO/IEC 25010: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, 2011.