Resolving the Historical Confusions about the Meaning of Software Size and Its Use for Project Effort Estimation

Charles Symons

[1] *(Retired, unaffiliated) Reigate, England*

Keywords:
Software measurement, project effort estimation, functional requirements, non-functional requirements

Extended Abstract

The software industry does not have a good track record of delivering systems on time and budget. In part this is due to weaknesses in software sizing and project effort estimating methods and practices. This paper gives the author's perspective on how some of these weaknesses have arisen historically, resulting in differing views on some basic underlying concepts that are still causing confusion today. Some remedies are proposed to help eliminate the confusions.

It is not easy to define what we mean by the 'size' of an item of software, and this size is not always clearly distinguished from the size of the project or the activity to develop and implement the software item. Even in recent years the Agile community has defined a measure (Story Points) of the size of a User Story (a simple statement of software requirements) but has used the same units of measurement to estimate or measure the effort to develop and implement the Story.

A software item manifests itself in different forms as it progresses through the states of its life-cycle from requirements, design, and source code to executing code (using a 'waterfall' development process for ease of illustration). In each state, the software item's artefacts differ thus necessitating different size measures, for example counts of SLOC (Source Lines of Code) for the source code, and bytes for the executing code.

Software size is usually the biggest driver of the effort to develop the size. Consequently, designers of software size measurement methods all aim that their sizes should correlate with effort. Allan Albrecht and other designers of early methods of sizing software requirements proposed to measure size as the product of two factors which became known as: i) the 'Unadjusted Function Point' (UFP) size, which is a measure of the required software functionality, and ii) a 'Value Adjustment Factor' (VAF) that aims to account for the relative difficulty of developing the software, e.g. by considering its 'complexity' and other types of requirements. These various VAF constituents are called 'size-drivers' in this paper. The product of a UFP and a VAF gives a software size in units of 'Function Points' (FP).

The paper argues why those software size measures that are one-dimensional (e.g. UFP, SLOC, bytes) may all be thought of as different types of software 'length'. It further concludes that there is no reason to expect that different methods for measuring the 'length' of software items at different states in their life-cycle will produce results that correlate well with each other, or with measurements of the same items that attempt to also account for the difficulty in developing the sizes, or that account for this difficulty in different ways.

Effort estimation methods must take account of other factors besides size, referred to as 'effort-drivers' in this paper. These can be either 'pure' effort drivers, i.e. attributes only of the project, such as the numbers and capability of the staff assigned to the project, or other types of requirements for the software system that are not taken into account by the software size

IWSM Mensura 2023, September 14–15, 2023, Rome, Italy
cr.symons@btinternet.com (C. Symons)

measures such as for its quality, e.g. portability requirements, or technical requirements such as the required response time.

The paper notes that the designers of different size measurement methods and different effort estimation methods have, at different times, made different decisions on whether to classify some of these factors as size-drivers or as effort-drivers. A further conclusion, therefore, is that accurate effort estimation requires a coherent allocation of these factors as either size-drivers or as effort-drivers. The result will be a 'Coherent Size/Effort Ecosystem'. Not recognising this need for coherence is another cause of confusion and a source of error in effort estimation practices that rely on converting an estimated size from one state, e.g. FPs, to another size, e.g. SLOC, for a different state before converting size to effort.

The paper next discusses the weights applied to the various components of the UFP and VAF sizes. These were all derived by expert judgements of the relative effort to develop the components. It is concluded that these early FP sizes are all, strictly-speaking, standard measures (or indices) of Relative Effort for a software project. In spite of this conclusion, the paper also shows that UFP-like sizes conform to the principles of Functional Size Measurement (FSM) for measuring Functional User Requirements (FUR) as defined by ISO/IEC. A UFP size may therefore also be legitimately regarded as a measure of software Functional Size. But VAF-like factors do not conform to these same principles, and this has led to their demise. As a matter arising from this discussion, the paper proposes improvements to the ISO/IEC definition of the term 'FUR' to clarify its real meaning.

The final major topic discussed is what we mean by the 'Non-Functional Requirements' (NFR) of a software-intensive system or software product. There are various definitions of NFR and varying ways of accounting for them in effort estimation. The different definitions list different examples as types of non-functional requirements; these lists overlap with the list of requirement types that were accounted for by VAF-like measures, and with the requirement types that are given as examples of what are not FUR in the ISO/IEC definition. In response to this variety of definitions, the COSMIC and IFPUG organizations collaborated to agree a definition of NFR which was compatible with the ISO/IEC definition of FUR. A consequence of these two definitions was that any software requirement could be clearly classified as either functional or non-functional,

With hindsight, this definition is flawed, as is the ISO/IEC definition of FUR because both define quality requirements as non-functional. This has caused confusion in practice, because many quality requirements appear, when first expressed, to be NFR but on closer examination evolve into FUR for software, or into a mixture of both FUR for software and requirements for 'non-software' e.g. hardware. As an example, a requirement for the security of a software system (typically regarded as a quality requirement) may be met by software or hardware, or a mixture.

The paper therefore proposes further revisions to the ISO/IEC definition of FUR and offers two options to revise the COSMIC/IFPUG definition of NFR to help resolve this confusion. These revisions recognise that all requirements for a software system must ultimately be allocated to software or to 'non-software'. The revised definitions leave customers and project teams to decide how quality requirements will be allocated in their own individual developments, regardless of whether they think of the requirements as a FUR or as NFR.

The paper then examines a method ('SNAP') which claims to measure a collective size of the NFR for a software item. The author gives several reasons why he considers this approach as an unwise way of dealing with NFR in a size/effort ecosystem.

In overall conclusion, the paper identifies two priorities for action by the software metrics community: first, to accept the proposed revisions to the ISO/IEC definition of FUR and to the COSMIC/IFPUG definition of NFR, and second to convince the Agile community to incorporate a modern functional size measurement method into their sizing/estimating eco-systems.