Process metrics for requirements analysis

Vincenzo Ambriola and Vincenzo Gervasi

Dipartimento di Informatica, Università di Pisa
{ambriola,gervasi}@di.unipi.it

Abstract. In this paper we propose a class of process metrics based on the continuous monitoring of product attributes. Two such metrics are defined for the requirements analysis process, namely *stability* (i.e., how smoothly the process of introducing information in a requirements document flows) and *efficiency* (i.e., which part of the effort of the analysts is spent in reworks). These measures can be used for the timely identification of risky trends in a requirements analysis process.
The paper also gives some results from an experiment on the collection and use of the measures we introduced.

Keywords: Requirements, process metrics, quantitative process evaluation.

1 Introduction

In recent years significant advancements have been made in the field of process modeling and on process-supporting technologies. However, the issue of measuring attributes of the software process has often received less attention. Classic work in this area has been mainly focused on the testing activity [4]. Test-like processes have a *reductive* nature: the goal of the process is to minimize some well-understood attribute of the product. For example, the testing process tends to the identification and removal of defects, while reducing the injection of new defects. A well-understood theory supports reasoning on such process (e.g., by using prey-predator models).

Design-like processes are different in that they exhibit an *accumulative* nature: their goal is to "grow" a product, increasing some of its attributes in the process. However, there is no optimal value for those attributes, again in contrast with the case of reductive processes. A typical example of accumulative process is the requirements elicitation and analysis activity.

It is well known [3] that requirements analysis is one of the most critical parts in the software development process: the late correction of errors in the requirements can cost orders of magnitude more than the correction of coding errors, due to the rework involved. It is thus not surprising that, from a managerial point of view, there is a strong need for some *control* on the requirements process.

Controlling a process means observing it, to obtain an understanding of the current situation; monitoring it, to follow its evolution; and providing guidance, to drive the process to the effective and efficient achievement of its proposed goals. The requirements process can be characterized as the progressive refinement of a document describing a (software-intensive) system. Consequently, it can be observed — and possibly measured — along three dimensions:

- the *document* dimension, pertaining to intrinsic attributes of the requirements document (e.g., size, structure, lexicon);
- the *system* dimension, related to the attributes of the system described in the requirements (e.g., number of components, functional score);
- the *process* dimension, that focuses on the activities involved in the production of the document (e.g., months/man spent, frequency and distribution of changes, mean time between releases).

In this paper we concentrate on those process attributes that can be measured by tracing the evolution of document and system attributes during the process. A simple example of this kind of attributes is the sampling of the document size at regular intervals during the writing process. Such a measure can be used, for example, to estimate the size of the document at some future date, or to verify if new information is being added to the document (increasing size), or rather if the information already present is being polished and structured (stable size).

2 Stability and efficiency analysis

We consider two attributes of the requirements process that are particularly interesting for the correct management of the process itself, namely *stability* (i.e., how "smooth" the analysis activity is) and *efficiency* (i.e., which part of the effort is spent on reworks). We propose a quantitative characterization of these two attributes, and then report on our experiences on their use in a case study.

2.1 Stability

While the requirements process can be ideally modelled as the progressive accumulation of information about the system to design, in practice it often has an *impulsive* behaviour. Experience shows that analysts tend to proceed in alternating phases of writing (adding new requirements) and polishing (correcting errors, restructuring descriptions). Stability is a measure of how smoothly these two phases are integrated. It is maximal for the ideal process in which information grows linearly with time, and gets worse for highly impulsive processes.

To measure stability, we first consider some measure $F(t)$ of the amount of information contained in the requirements[1] at time t. The differential measure $\delta F(t) = F(t) - F(t-1)$, defined for $t > 0$, indicates, for each instant t, the difference in information content with respect to the previous version of the

[1] See Section 3 for examples of such measures.

requirements. A frequency analysis (Fourier transform) of $\delta F(t)$ provides the information about the distribution and magnitude of the impulsive phenomena in the analysis activity. We can identify four classes of processes with respect to stability, according to the principal frequency components of $\delta F(t)$:

I. *low peaks on high frequencies.* Small, continuous changes; the activity is strongly incremental in nature.
II. *low peaks on low frequencies.* Small, infrequent changes; the activity proceeds by relatively minor, well spaced impulses.
III. *high peaks on low frequencies.* The activity proceeds by substantial jumps apart each from the other; the development activity is thus strongly impulsive.
IV. *high peaks on high frequencies.* Frequent modifications of relevant magnitude; the process is characterized by continuous and important upheavals.

Under the hypothesis that stable processes lead to products of predictable quality, we can expect that processes of class I (incremental) produce better requirement documents, while processes of class IV (catastrophic) end up producing documents of dubious quality[2]. Processes of class II (sleepy) and III (impulsive) can be expected to produce results of intermediate quality.

This hypothesis can be validated by independent evaluation of the results of processes of the various classes, as described in Section 3 (see also [1]).

2.2 Efficiency

The measure $F(t)$ introduced above can also be used to measure the efficiency of the analysis process. In an ideal accumulative process, every effort made by the analyst corresponds to a certain amount of information that is added to the requirements. In reality, requirements are seldom complete and correct at the first try, and a certain amount of rework is unavoidable. We can approximate the effort spent by the analyst in modifying a document by measuring the difference in information content between two successive revisions. It is worthwhile to notice that our process is *not* conservative: a certain effort is required both to increment *and* to decrement $F(t)$ (in both cases, a certain conceptual and text editing work has to be performed). In other words, there is no such thing as a "potential of information content", analogous to the potential found in conservative fields in physics.

[2] We use the term *dubious* here to mean that in some particular case the quality of the requirements could indeed be reasonable, but there is no way to predict the outcome beforehand: it depends almost entirely on the very last "revolution" in the document.

Formally, we can estimate the total effort spent by the analysts between revisions a and b[3] by

$$W(a,b,F) = \sum_{t=a+1}^{b} |\delta F(t)|$$

whereas the effective change in information content between a and b is simply

$$E(a,b,F) = F(b) - F(a)$$

The efficiency of the process is the ratio between the global increase in information content and the effort spent, i.e.

$$\varepsilon(a,b,F) = \frac{E(a,b,F)}{W(a,b,F)}$$

By definition, $0 \leq \varepsilon(a,b,F) \leq 1$. When the efficiency is 0, $F(a) = F(b)$ so no information was added between revisions a and b, regardless of any modification made in intermediate revisions. When the efficiency is 1, all the effort was spent in adding information content, i.e., $F(t)$ is monotonically non-decreasing in the interval $[a,b]$.

Notice that $\varepsilon(a,b,F)$ is only a measure of the efficiency of the *accumulative* component of the requirement process, and not a measure of the quality of the work made by the analysts. Indeed, a significant rework can well decrease the information content of a document while increasing its overall quality (according to some criteria). In this case, a low $\varepsilon(a,b,F)$ would indicate that the accumulative component that preceded the rework was not efficient, since the document needed a rework later, and thus part of the initial effort was lost.

It is also worthwhile to notice that this efficiency measure can only be computed *post hoc*, since there is no way to know if part of the information accumulated in the requirements up to a certain revision t_r will be removed or reworked in later revisions (thus decreasing efficiency) or will stay in place until the end of the requirements analysis activity (thus increasing efficiency). This observation leads to the conclusion that $\varepsilon(a,b,F)$ does not lend itself to a predictive use of the efficiency model; it can be used, however, to estimate the expected efficiency of further iterations of the requirements process in similar conditions — i.e., for the next project by the same team.

3 Experimental validation

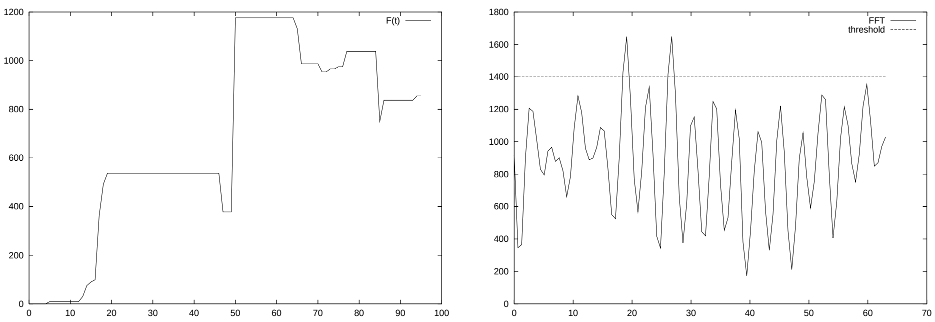
To test the usefulness of the two measures we are proposing here, an experiment was set up by collecting a number of system measures on the requirements produced by 16 teams working in parallel on the same project in an academic environment. Each team was given approximately four months to do a complete

[3] We consider only those intervals $[a,b]$ in which some work has been done, i.e. $W(a,b,F) > 0$.

requirements analysis for a distributed weather monitoring system, followed by a design phase and by the implementation of a prototype of the system. In particular, requirements analysis was conducted with the support of our requirements engineering environment CIRCE [2, 5].

We selected four "information content" measures to use as $F(t)$, namely functional score F_f (measured in feature points [6]), behavioural complexity F_b (measured by counting decision points), static complexity F_s (measured by counting the number of entities, relationships and attributes) and document size F_d (measured by counting the number of atomic requirements). The measures were automatically computed by CIRCE's metrication modules from the documents written by the analysts. Tool support was fundamental at this stage, to guarantee repeatability of the measures and to reduce the manual effort involved in collecting the measures to almost zero.

We discovered significant evidence that the stability measure had a high predictive value: teams with unstable requirements processes invariably had to do a big amount of rework on the requirements in later stages of the development process. The efficiency measure, lacking predictive value, could only be empirically validated by interviewing the teams about their perception of how their work was spent.



(a) (b)

Fig. 1. Stability measure: (a) functional score, (b) impulse analysis.

As an example, Figure 1(a) shows the $F_f(t)$ function (functional score of the system described by the requirements) over a three-months period for one of the teams participating in our experiment. It can be easily observed how impulsive periods of accumulation of information (e.g., around days 17–20 and 50–52) are followed by stable periods of analysis and rethinking (around days 21–47 and 51–65) and then by periods of refinement and possibly of rework (around days 48–50 and 66–68). The frequency analysis of $\delta F_f(t)$ is shown in Figure 1(b). Notice how the first two principal components (with periods around 7 days, due to the weekly rhythm of work, and around 4–5 days, originating from rework activity that can be observed on days 13–17, 47–51, 71–76 — each period being 4 or 5 days long) let us quantitatively classify this particular instance as a type III process, i.e. with substantial but well-spaced reworks.

Given this information, a project manager could well insist that sudden upheavals like those shown in Figure 1 should be avoided, and take appropriate actions to set up a more stable process instead.

4 Conclusion

Requirements writing has often been associated with high risks due to the difficulty of controlling the requirements process. The continuous sampling of product attributes provides an effective means of obtaining process metrics that can be used to control the requirements activity.

The availability of a supporting environment is an enabling factor that can drive the collection and use of metrics of this kind. Differently from traditional cases, which are based on counting process events (e.g., the number of bug reports coming back from beta testers) and on measuring times between events, our technique of projecting product attributes along the time axis requires some form of automatic support to be applicable without placing an excessive burden on the participants in the process.

Once such an automatic support is in place, however, continuous monitoring of product attributes offers marked advantages. In particular, the two measures of the accumulative component that we proposed make it possible for a project manager to identify risky or inefficient behaviour of the analysis team and to take corrective actions to ensure the timely delivery of stable requirements.

References

1. V. Ambriola, R. Di Meglio, V. Gervasi, and B. Mercurio. Applying a metric framework to the software process: an experiment. In B. C. Warboys, editor, *Proceedings of the Third European Workshop on Software Process Technology*, pages 207–226. Lecture Notes in Computer Science Nr. 772, Springer–Verlag, Feb. 1994.
2. V. Ambriola and V. Gervasi. Processing natural language requirements. In *Proceedings of ASE 1997*, pages 36–45. IEEE Press, 1997.
3. B. W. Boehm. Verifying and validating software requirements and design specifications. *IEEE Software*, 1(1):75–88, Jan. 1984.
4. N. Fenton and S. L. Pfleeger. *Software Metrics - A Rigorous and Practical Approach*. International Thomson Computer Press, London, 2 edition, 1996.
5. V. Gervasi and B. Nuseibeh. Lightweigth validation of natural language requirements. In *Proceedings of the Fourth IEEE International Conference on Requirements Engineering*, June 2000. (to be published).
6. C. Jones. *Applied Software Measurement: Assuring Productivity and Quality*. McGraw-Hill, New York, N.Y., 1991.