Project-entropy: A Metric to Understand Resource Allocation Dynamics across Software Projects

Subhajit Datta
Department of Computer Science
Florida State University
Tallahassee, FL 32306-4530, USA
Email: sd05@fsu.edu

Robert van Engelen
Department of Computer Science
Florida State University
Tallahassee, FL 32306-4530, USA
Email: engelen@scs.fsu.edu

***Abstract*— Reliability of a software system, or the lack of it, is often reflected in user satisfaction. Software development organizations frequently need to reallocate resources amongst projects to help satisfy user needs better. In this paper, we propose the *project-entropy* metric to understand the dynamics of such resource allocation across projects. Calculation of the metric is illustrated through an example scenario; and we hypothesize on the existence of an *entropic limit* for an organization. Open issues and plans of future work are also outlined.**

I. Introduction and Motivation

In a typical software development organization, many projects run concurrently. Resources from a common resource-pool are deployed to the projects, and redeployment of resources from one project to another happens frequently. Often, resources are diverted to a project with low user satisfaction from a project that is at a relatively higher satisfaction level. For several reasons, we can not ignore such situations as mere symptoms of the ignorance of Brooks' Law [2], which mandates adding people to an already late project will only make it later. The troubled project may be fetching customer dissatisfaction for issues unrelated to schedule. The diverted resources may not just be people; for example, more servers running larger suits of automated regression tests can help fix issues that were earlier being discovered only during user acceptance tests. Besides, underlying assumptions as well as the veracity of Brooks' Law have been questioned for many common scenarios [5], [4]. In terms of its ubiquity and utility, reallocation of resources from one project to another within an organization towards ensuring higher user satisfaction is an interesting phenomenon. In this paper we present the project-entropy metric to better understand the dynamics of such resource flow and consider whether there is a limit beyond which reallocation does not lead to enhanced user satisfaction.

Though it is difficult to find an universally accepted definition of "software entropy", the idea of entropy has been invoked to understand the degradation of software with use [1], its inherent complexity [3] etc. While we recognize the value of these studies, this paper takes a more *organizational* view of entropy in the software development context.

The notion of project-entropy is inspired by the thermodynamic idea of entropy. Entropy is taken to represent disorder and chaos; an antithesis to efforts that can lead to any organized and favorable outcome. When projects start, plans look perfect on paper. But with the progression of their life cycles, disarray manifests, fuelled by unexpected risks, oscillating requirements and a slew of other unforeseen realities. Project-entropy helps us analyze the actions taken at an organizational level to address the effects of this inevitable decay of order across a set of projects. In the following sections, we explain project-entropy further, illustrate its application through an example and conjecture about the effects of its increase in an organization.

II. Project-entropy

In the context of a project, we define *satisfaction* (F) as the *percentage of user acceptance tests succeeding per release*, and *endeavor* (E) as the *resource-hours deployed per release*. (Resources are most frequently personnel, but they can also be anything else needed for fulfilling project tasks, such computing equipment etc.) We assume the project follows the iterative and incremental development methodology. A *release* is thus an incremental launch of a subset of the project's functionality after an iteration of development; for users to test, use and give their feedback. A user acceptance test *succeeds* when it confirms that the aspect of the software system being tested by users is functioning as per their expectations. Evidently, the goal of the development organization is to distribute endeavor such that satisfaction in each project is maximal.

We take our universe as the software development organization. Each individual project running within the organization is a system of interest. When endeavor flows from one project to another, and ΔE is the amount of endeavor transferred into or out of a project which is at satisfaction level F, ΔP is the change in *project-entropy* (P), which is given by,

$$\Delta P = \frac{\Delta E}{F} \tag{1}$$

III. An Example Scenario

Let us consider an example scenario with reference to Figure 1. A software development organization has three projects running, A, B, and C. Table I shows the units of satisfaction of the three projects at times T_1, and T_2. At T_1, 21 units of endeavor are moved from C (at $F = 73$) to B (at $F = 27$). Thus for the whole organization, the

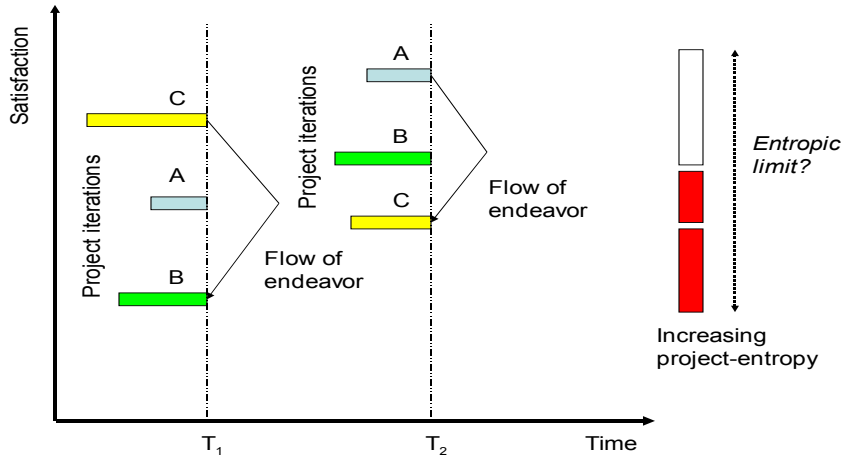

Fig. 1. Flow of Endeavor across Projects and the Entropic Limit

TABLE I
SATISFACTION LEVELS FOR PROJECTS A, B, C AT TIMES $T_2 > T1$

	Project A	Project B	Project C
t = T_1	56	27	73
t = T_2	85	66	54

project-entropy increases by $21/27 - 21/73 = 0.49$ units. Similarly, at T_2, if 35 units of endeavor are moved from A (at $F = 85$) to C (at $F = 54$), the project-entropy increases by $35/54 - 35/85 = 0.24$ units. So the net increase in project-entropy for the organization is $0.49 + 0.24 = 0.73$ units. As endeavor is diverted from a project at higher satisfaction to one at lower satisfaction, project-entropy invariably increases for the organization. What does this increase in project-entropy mean at the organizational level?

Endeavor is moved from a project at a higher satisfaction level to one at a lower level with the expectation that satisfaction will increase in the latter. This is likely to work well during the earlier iterations; but as projects go deeper into their life cycles, reallocation of endeavor slowly loses its capacity to increase satisfaction. This can depend on many factors: circumstances of a long running project may present a steeper adjustment curve to redeployed resources, low satisfactions for two long may already have prejudiced users so that no amount of positive results appeal to them any more, frequent realignment of resources may have adversely affected team synergy etc. But these factors may just as well be mitigated up to a *limit* by organizational capability and maturity, adherence to processes and best practices, experienced and talented personnel etc. Based on the discussions so far, and general observation of the ways of software organizations that have several projects running simultaneously, we put forward the following hypothesis: *For given set of projects in an organization, there exists a level of project-entropy – an entropic limit – beyond which reallocation of endeavor amongst the projects will not result in significant increase in satisfaction.* Recognizing the entropic limit will help organizations plan their resource allocations with more purpose and effect.

IV. Open Issues and Future Work

The hypothesis proposed above needs to be validated in the light of empirical data across a range of projects and organizations. We are in the process of extending the simple example outlined earlier to cover more involved scenarios and conducting further case studies. We also expect empirical data to indicate the relationship between endeavor and satisfaction. The underlying assumption of diverting endeavor to a troubled project is that it will enhance satisfaction. From our experience, this correlation seems to hold (till the entropic limit, as we hypothesize). But is satisfaction linked linearly to endeavor, or is there a more complex relationship? Also, we have worked with the formula for the *change* in project-entropy. It would be helpful to be able to measure the entropy of a project, irrespective of endeavor being added or taken away from it. Will a definition of project entropy along the lines of $P = klog(W)$ – again, inspired by thermodynamics – where k is an *project* constant and W relates to the combinations of situations in a project that influences project-entropy, withstand empirical validation? Another question of interest is whether project-entropy is correlated in any way to a reliability measure such as Mean-Time-To-Failure (MTTF) of the software system developed by the project. We seek to address these questions through our ongoing and future work.

V. Conclusion

This paper introduces and illustrates the use of the project-entropy metric to understand the dynamics of allocating resources across software projects. We also forwarded a hypothesis regarding the limit to which resource reallocation enhances user satisfaction and outlined plans for further empirical validation of our ideas.

References

[1] BIANCHI, A., CAIVANO, D., LANUBILE, F., AND VISAGGIO, G. Evaluating software degradation through entropy. In *METRICS '01: Proceedings of the 7th International Symposium on Software Metrics* (Washington, DC, USA, 2001), IEEE Computer Society, p. 210.

[2] BROOKS, F. P. *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition*. Addison-Wesley, 1995.

[3] HARRISON, W. An entropy-based measure of software complexity. *IEEE Trans. Softw. Eng. 18*, 11 (1992), 1025–1029.

[4] MCCONNELL, S. Brooks' law repealed. *IEEE Softw. 16*, 6 (1999), 6–8.

[5] RAYMOND, E. S. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly, 2001.