Working on One Part at a Time is the Best Strategy for Software Production: A Proof

Francisco Zapata[1], Maliheh Zargaran[2], and Vladik Kreinovich[2]

[1]Department of Industrial, Manufacturing, and Systems Engineering
[2]Department of Computer Science
University of Texas at El Paso, El Paso, TX 79968, USA
fazg74@gmail.com, mzargaran@miners.utep.edu, vladik@utep.edu

It is possible to start earning money before the whole software package is released. When a company designs a software package, usually, it does not have to wait until the whole package is fully functional to profit from sales: the company can often start earning money once some useful features are implemented; see, e.g., [1].

In view of this possibility, what is the optimal release schedule for different parts? The software projects consist of several parts. Some of these parts depend on others, in the sense that we cannot design one part until the other part is ready. This dependency relation makes the set of all parts into a partially ordered set, i.e., in other words, into a directed acyclic graph.

For each part i, we know the overall effort e_i (e.g., in man-hours) that is needed to design this part (by utilizing, if needed, all the parts on which it depends). We also know the profit p_i that we can start earning once this part is released – by adding this part's functionality to whatever we were selling before.

So, if we release part i at time t_i, then by some future moment of time T, selling this part will bring us the profit of $p_i \cdot (T - t_i)$.

The question is: how can we organize our work on different parts so as to maximize the resulting overall profit.

What is known. In [1], several semi-heuristic strategies are described that lead to optimal (or at least close-to-optimal) release schedules.

Interestingly, while it is, in principle, possible for the company to work on several parts at a time, in all known optimal schedules, the design is performed one part at a time.

What we do in this paper. In this paper, we show that the above empirical fact is not a coincidence: we will actually prove that in the optimal schedule, we *should* always work on one part at a time.

References

1. M. Denne and J. Cleland-Huang, *Software by Numbers: Low-Risk, High-Return Development*, Prentice Hall, Upper Saddle River, New Jersey, 2004.