Alex Groce (agroce@gmail.com), Northern Arizona University

Charles Babbage's *On the Economy of Machinery and Manufactures* is good enough that it makes us regret that the great Babbage is not around to turn his eye to the modern software industry at large, or perhaps to open source development (Eghbal's *Working in Public* is a good book, but another such would not do anyone any harm).

On the Economy of Machinery and Manufactures (henceforth *OEMM*, not a bad acronym here) is not the kind of exciting book that Babbage's autobiography, *Passages from the Life of a Philosopher*, which gives this column its name, is. *Passages from the Life of a Philosopher* is almost a picaresque steampunk adventure novel, where Babbage sets out to descend into volcanos and meet eminent Victorians in order to show the modern reader how it's done. There are long parts that are not narrative, of course, Babbage's digressions/rants on everything under the sun, especially street musicians and his critiques of the shortsightedness of government funding, but much of the book consists of extraordinary incidents from Babbage's life.

OEMM, in contrast, is not a narrative; there are scattered brief stories of, e.g., successful improvements to manufacturing methods, or the woes brought about by some labor dispute, but by and large this is a numbered sequence of facts, sometimes arithmetically analyzed, about the state and nature of British manufacturing at the time. How is this relevant to SEN? Well, the first numbered item in the book begins: "1. There exists, perhaps, no single circumstance which distinguishes our country more remarkably from all others, than the vast extent and perfection to which we have carried the contrivance of tools and machines…" Software engineering is inherently interested in the contrivance of "tools and machines" (machines broadly construed) for the construction of software. Babbage's book gives insight into two permanent features of the world that are essential to the art and science of software engineering. First, there is the general question of how firms dedicated to the creation of certain artifacts (here, everything from pins to clothing, in our case, software systems) operate and can succeed. Some elements of this problem are inherently tied to the physical machinery Babbage examine, and his lengthy examination of modes of using and enhancing power via steam are interesting, but not particularly germane to our field. The other element, however, is the "power" in another sense of improvements in tooling, automation, and methods for making the work of making things proceed more smoothly.

One element of this is Babbage's perceptive emphasis on the fact that improvements "can be applied with equal success to mental as to mechanical operations, and that it ensures in both the same economy of time" – here discussing division of labor, but more generally in effect proposing algorithmic improvements. The full context of this final proposal of this point comes at the beginning of Babbage's 35th, and final, chapter, beginning with the 453rd numbered section(!) starts thus:

> "In reviewing the various processes offered as illustrations of those general principles which it has been the main object of the present volume to support and establish, it is

> impossible not to perceive that the arts and manufactures of the country are intimately connected with the progress of the severer sciences; and that, as we advance in the career of improvement, every step requires, for its success, that this connection should be rendered more intimate.
>
> The applied sciences derive their facts from experiment; but the reasonings, on which their chief utility depends, are the province of what is called abstract science. It has been shown, that the division of labour is no less applicable to mental productions than to those in which material bodies are concerned; and it follows, that the efforts for the improvement of its manufactures which any country can make with the greatest probability of success, must arise from the combined exertions of all those most skilled in the theory, as well as in the practice of the arts; each labouring in that department for which his natural capacity and acquired habits have rendered him most fit."

The entire final chapter is a panegyric to the combined effect of abstract science and practical knowledge, which seems a proper placement for the field of software engineering. This grand vision is the most exciting aspect of the book, for software engineers, I suspect, though there are many intriguing points scattered through this (short) volume. For example, earlier in the book, the idea of improvements to mental/intellectual "machinery" is embodied in a description of the method of differences, and some of the ideas informing Babbage's simpler computing machines for producing mathematical tables. Similarly, Babbage's extended examination of methods of *copying* items, and the advantages of methods that produce little or no degradation in quality, must bring to mind the effects and value of digital copying, where a book can be endlessly reproduced, with essentially no degradation and at extraordinarily low cost. Such connections between the industrial and information revolutions are why we turn to Babbage, and the pre-foundations of the field of computing, again and again, to enrich our thinking about the present.