Robert L. Glass: Software Runaways - Lessons Learned from Massive Software Project Failure.
Prentice Hall PTR, Upper Saddle River, NJ 07458, 1998, 259 pp., ISBN 0-13-673443-X, US$???

Software Runaways - Projects to Learn from

„There seems to be a much more indelible lesson to be learned from failures than from successes.“ is one of the first sentences in the preface of the book. Robert L. Glass collected more than a dozen stories about software projects which failed with an echo in the public. He did the best thing one can do to advance the field of software engineering according to the failure guru Henry Petroski [1]: „And since failures contain more unambiguous information than successes, the most fruitful data that any designer can be provided are case studies of failure or the explicit avoidance of failure.“ Add to designer ‘manager and project leader’, and you have the intended audience of the book which is a collection of war stories (term Glass uses) published in the decade 1986 to 1996. Glass ordered them according to the main cause of the failures and provides his insights.

A software runaway is „a project that goes out of control primarily because of the difficulty of building the software needed by the system“. Glass’ definition of ‘out of control’ is really out of control - project consumed twice its allotted estimated time or more and consumed twice its estimated cost or more and the product could not meet the (functional, reliability) requirements. It is this high standard which prevented him from including more war stories.

„Perhaps because software is a product built from no physical resources, a product constructed purely out of intellect, it is especially devastating to the psyche when it fails.“ It is even more devastating that it is an accepted behavior, it is considered to be business as usual that projects or activities within a project don’t meet the targets. ‘O.k. you did not keep the deadline - when do you think you’ll accomplish the task?’ That’s it. No search for reasons, no consequences, no learning on either side. This type of culture inevitably leads to runaways.

Chapter 1 of the book defines software runaways (see above) and distinguishes them from projects in crunch mode (there is a threat to its reaching its original targets) and death march (the targets exceed the norm by at least 50%) projects. The death march projects and the ones in crunch mode risk to become runaways but don’t need to. Research findings, expected and surprising ones, and trends conclude the chapter.

Chapter 2 contains the war stories grouped according to the frequency of their cause:

1. project objectives not fully specified, e.g. the Denver airport story
2. bad planning and estimating, e.g. the Mitch Kapor story
3. technology new to the organization, e.g. the New Jersey DMV story
4. inadequate / no project management methodology, e.g. the IRS story
5. insufficient senior staff on the team, e.g. the CONFIRM project
6. poor performance by suppliers of hardware / software (actually no story under this heading)
7. other - performance (efficiency) problems, e.g. the NCR inventory system

The stories are reprints. Where available Glass collected more than one publication about the same project (e.g. Denver, IRS, CONFIRM); we can ‘see’ the same project from different angles and obtain a better perspective on what did happen.

One of the trends Glass identifies is the increasing relevance of the technology as cause of failure. This contrasts with the folk wisdom telling us that management accounts for the vast majority of failed projects. One could argue that it is a management problem if the technology choice is not verified or the development process is not adapted (because new technology requires different types of activities conventional development processes do not include) or the staff is not adequately trained in the use of new technology. We don’t argue, we accept the increasing relevance of the technology; it sounds logical if we take into account its rapid change.

In Chapter 3 remedies like risk and issue management are presented. These are considered by the author to be text book remedies with little practical value. Practical remedies used in most of the companies are extending the schedule, improve management procedures, take on more people, increase funds, pressure on suppliers by withholding payments, reduction in scope of the project, new outside help, etc. Seldom does a single one of these bring the project running away back on track.

The book is an insightful reading for everybody involved in software projects. Managers and project leaders could learn most for their daily work. Software developers reading the book can help improve their sensors for the danger of runaways and encourage them to trigger management action before the project indeed runs away.

Henry Petroski writes in [2]: „Thus the tragedy no doubt made a lot of inexperienced detailers suddenly much more experienced. And it is precisely to keep these lessons in the mind of young engineers that failures should be a permanent part of the engineering literature.“ because „Yet no disaster need be repeated, for by talking and writing about the mistakes that escape us we learn from them, and by learning from them we can obviate their recurrence“. Thanks to the authors of the stories we are able to learn and try to avoid the mistakes our colleagues already tried out. Thanks to Robert L. Glass this learning is an easy exercise. It would be to the benefit of the community if many software professionals would take these lessons. Before a project starts every manager and every project leader should read the book (again).

[1] Petroski, Henry: Design Paradigms, Cambridge University Press, 1994
[2] Petroski, Henry: To Engineer is Human - The Role of Failure in Successful Design, St. Martin's Press, 1985.

March 17, 1999

Karol Frühauf, *INFOGEM AG*, 5401 Baden