Agile software requirements : lean requirements practices for teams, programs, and the enterprise

Leffingwell D., Addison-Wesley Professional, Upper Saddle River, NJ, 2011. 560 pp. Type: Book (978-0-321635-84-6)

Date Reviewed: Jun 2 2011

Full Text

This book is a major contribution to modern software development. Leffingwell addresses the most important part of every software development process: requirements practices. Without a doubt, it is definitely a good source of knowledge for "lighter-weight, more flexible, more adaptable and more user-centric software development methods." It is impossible to predict just by reading it, however, if this book will stand the test of time.

We know very well that the biggest bottleneck in most projects is that of software requirements. By providing principles and techniques, the author's contribution seems to be of great importance to software development educators and practitioners. Basically, the approach combines agile development, lean software, and a software Kanban system to improve the software requirements process.

Some may criticize the author for the lack of advice on handling the large number of requirements and prioritization, although the use of the hierarchy of different requirements seems to be the solution. For many epics (groups of related user stories) with different priorities and features, however, this is a nontrivial, multi-criteria problem.

The book addresses "User Story Acceptance Criteria" in 20 lines of text (pages 104 and 105). Needless to say, this is insufficient. Multi-criteria decision analysis (MCDA), or multi-criteria decision making (MCDM), is a well-established discipline. In the software requirements process, different stakeholders must consider numerous, and often conflicting, evaluations.

The author addresses this issue on his blog:

> The purpose of this model is to allow an enterprise a basic mechanism to reason about differing kinds of things at differing, and appropriate, levels of abstraction. As an artifact model, it doesn't tell you how to prioritize one epic from another, but it's an improvement over attempting to prioritize unlike things, for example an important epic versus an important story. [1]

We could consider this a reasonable explanation based on practicality.

Another potential deficiency of the book is its coverage of nonfunctional requirements (NFRs): this topic is covered in one 16-page chapter. We know, however, that this will be an open problem for years to come.

The book concludes: "The real work is left up to you, the reader. [...] The rest is actual software, and that's up to you." This seems very appropriate, since there is no magic powder in software development other than hard work. The author's contribution to the very difficult problem of software requirements is commendable, but not a turnkey solution.

Reviewer: Waldemar Koczkodaj

Review #: CR139103 (1201-0015)