Preface

Gerald M. Weinberg, author of the book "The Psychology of Computer Programming" is attributed with the quote —"*If builders built houses the way programmers built programs, the first woodpecker to come along would destroy civilization*." I could not agree more. The rate of project failure is much higher in software development compared to either manufacturing or construction. It is not that there are no failures in manufacturing or construction. Those failures are in "first-of-its-kind" projects, especially in manufacturing. In construction, these are even fewer. For example, the Empire State Building of New York city was the first of its kind when it was built. It is the first building in the world to go up 80 floors high above ground. It was the tallest building in the world for a number of years. The issues, there would have been many, were solved in the specifications and the design stage. The construction would scrupulously adhere to the design. It was a success.

Why do software projects fail at such a high rate even when there were similar projects executed earlier?

Two major causes are attributed for this phenomenon. The first is the poor understanding and definition of product requirements. This leads to technical failure. The second is the poor project management of developing the software product to the specified requirements resulting in unsustainable overruns of cost and schedule. Both the reasons lead to project failure.

In this book, I am focusing on the precise understanding and definition of product requirements. As in other areas, there is more misunderstanding about this critical activity than right understanding.

"The hardest single part of building a software system is deciding what to build… No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later." said Frederick Brooks, Jr., Brooks Computer Science Building, University of North Carolina, USA. (From his paper "No Silver Bullet: Essence and Accident in Software Engineering," 1986 as also *The Mythical Man-Month: Essays on Software Engineering,* Anniversary Edition, Chap. 16.) I concur wholeheartedly.

Unfortunately for me or perhaps for the software development industry itself, there is no commonly accepted taxonomy for software engineering activities. It is not that there are no definitions at all. The definitions that are there, are not universally accepted. Some, like "non-functional requirements" are downright ridiculous. That is the reason why I am explaining every term I use here, as you will notice, from the first fundamentals. Wherever, possible, I am using the terminology from a credible source. I am giving all the available meanings with my own interpretation with the idea that the reader is better informed when the same matter is presented by someone else with a different set of nomenclature. Please bear with me for this over-specification.

My own perception was gathered from my experience, observation, study and participation in discussion forums of how well the requirements are either engineered or managed. It is not very flattering to the community of requirements engineers or managers. There are many doubts, which begin right at the fundamental stage among the practitioners. This is, perhaps, due to the fact that universities are not conducting courses in requirements engineering and management. Their focus is more on engineering and producing code rather than on the forward or backward linkages to code production. Managements rather hasten the project teams into coding ASAP. Of course, there are exceptions without which, I would not have been able to gather best practices.

In this book, I tried to give you a complete view of the activities of requirements engineering as well as requirements management. Both the activities, engineering and management, are equally important. Engineering activities, performed well, produce the right deliverable. When we manage the engineering activities diligently, we produce the deliverable within the estimated cost and on schedule. The variances that are bound to be there would be predictable and within acceptable levels. Management activities when performed diligently would also allow us to plow the experience back into the process of performing engineering activities and facilitate improvement.

The information presented here is from my experience, observation, academic study and participation in the Internet discussion forums.

That was my intent and I would like to learn how you perceived my effort to be. Please feel free to email me at murali@chemuturi.com and I promise to respond to every email that I receive normally in one business day.

June 2012 Murali Chemuturi

http://www.springer.com/978-1-4614-5376-5

Requirements Engineering and Management for Software Development Projects

Chemuturi, M.

2013, XXII, 266 p., Hardcover
ISBN: 978-1-4614-5376-5