Managing Systems Development 101

A Guide to Designing Effective Commercial Products & Systems for Engineers & Their Bosses/CEOs

James T. Karam

The Technical Manager's Survival Guides, Volume 2
Marcus Goncalves, Series Editor

Table of Contents

Table of Figures

Table of Tables

Introduction

So, are you a young engineer that has been asked to become a lead for a team of specialists to work on a product or project that requires many different skills or even several teams? Have you been a lead engineer but have now been asked to be a manager of your department? Have you shown both the inclination and the capability to broaden out of your specialty to become a project or product development chief engineer or manager? Have you managed projects or departments and now you have been asked to manage those managers? In all these cases, you are confronting topics daily that they never taught you in school as you find yourself involved with managing the engineering of what are called "systems".

Managing the development of large-scale systems can be both fun and satisfying. The U.S. Department of Defense (DOD), notably the Air Force (USAF), codified the methodology of such management in the late fifties and sixties in MIL-STD 499 and its ilk. They took their lessons learned from fielding Intercontinental Ballistic Missiles and the like, both good and bad, and embodied them in processes that continued to mature. Many engineers spent at least some of their career in aerospace and this systems culture. However, since "peace broke out" in the early nineties, this opportunity for systems on-the-job training (OJT) has substantially diminished.

This book addresses many of the key topics you will face in your expanded responsibilities. There are good textbooks on the topic of systems engineering, but most still focus primarily on the very large systems of systems typical of aerospace and defense. Further, as textbooks, they tend to focus understandably on the generic processes involved, primarily regarding the earlier phases of development. Regardless, several are cited in a closing section as candidates for additional reading. Instead, this book focuses on specific practical advice to use when executing those processes in commercial environments. In effect, our focus is on the practical mechanics of management. As such, it can also provide an incisive refresher of useful tricks of the trade even for professionals in aerospace.

While large commercial systems also existed, they were mostly the domain of mainframe computer developers until the eighties with its advent of the ubiquitous personal computer (PC). Then the nineties saw the introduction of the World Wide Web (WWW) and a plethora of personal and business software applications of all sizes. Further, PCs became so powerful that many, if not most, applications that used to

require large computers or, more commonly, highly specialized and customized electronic hardware could now run on these relatively cheap machines.

Almost every capital goods industry saw their hardware commoditized to some degree with their products' functionality provided mostly by software. This commoditization of hardware was a watershed event as it meant that software development would become a critical asset (or heartache) for most every industry and product. Moreover, large systems were routinely created using a collection of PCs, some evident and some embedded, but PCs nonetheless. So, where did developers learn to put such commercial systems together? Folklore said that aerospace processes were gross overkill with an excessive focus on paperwork.

In addition to the regulated industries like nuclear and medical equipment that had done so previously, most companies in all industries formalized their system development processes in response to the pragmatically mandatory need to get themselves certified to the ISO-9000 quality standard in the nineties. Many made the mistake of over-promising, particularly with respect to the paperwork, since they proposed to behave like they thought someone might have expected, rather than what they had always done. Either they drowned in their own paperwork, or, more commonly, quickly lapsed into old habits and prayed an auditor would not show soon. (The proper solution was to edit the procedures and processes to reflect what was reasonable. Generally, auditors do *not* tell you what you should do, but only if you are complying with what *you* said you should do.)

As one who stumbled through some of those choices, my conclusion quickly became that, while one should tailor the formalism in a commercial environment, systems are systems, and the aerospace system engineering process basics remain the key to success anywhere. While somewhat facetious, the section titles typically end in "101" because the basics are where your problems, and their solution, lie.

Chapter 1 starts with a review of the key elements of the project systems engineering process. While still the way of life in aerospace and defense, many engineers in commercial enterprises lack exposure to even the terminology of systems development. This initial chapter provides that context along with practical advice regarding execution. Project/program planning is addressed in Chapter 2, as these plans, in effect, become the internal contracts between the various development groups and their management and customers. In fact, it is hard to even claim that one *is* a manager without a plan, much less actually manage,

rather than just react. This section ends with Chapter 3 discussing several topics to consider pragmatically during the various phases of a program or product's lifecycle or evolution, notably at the beginning and at the end of a project.

The next chapters address some of the key mechanics of managing systems development. Since software is such a dominant part of any system nowadays, we start Chapter 4 with a set of very basic design practices that seem to be ignored or forgotten by developers. These topics *were* taught in school, probably in their introductory courses, and staff usually resent being reminded. However, they recur so often that they should remain your focus. Chapter 5 recommends using clickable mockups to facilitate timely development of graphical user interfaces (GUIs) in products. While admitting that they represent just one particular religious bias, we also include an example of GUI design practice rules. We said "religious" because, like many other issues, there is no technical right or wrong involved, just a preference. Nevertheless, the benefit arises to your team because you state your belief, almost independent of its specifics.

Chapter 6 moves away from managing software to using software to make presentations. Every manager is also, some would say mostly, a salesperson. While presentation style would seem to be the ultimate religious preference, we recommend that you become a zealot. Very simple rules are recommended, and they work. Chapter 7 implores and explains how to find and empty all the full in-boxes in your span of control. Nothing you can do will improve responsiveness more. Then, the process of Continuous Improvement is advocated and explained in Chapter 8, with practical examples from all operational departments.

The next set of chapters address people-related topics since people are your means to success. Chapters 9, 10, and 11 address performance ranking, incentive criteria, and matrix organizational structures, respectively. These provide a succinct practical guide to these topics whose mechanics are rarely dealt with, except by osmosis.

Finally, Chapter 12 offers success in improving your productivity with tools, provided you adapt your behavior to them, not vice versa.

Closing remarks refresh our key advice. Candidates for additional reading conclude the text.

Closing Thoughts

Recapping some of my favorite advice…

That's a solution, not a requirement.

Ambiguity in a specification is always to the buyer's advantage

If there is only one feature of aerospace system practice that you can adopt, it should be *design reviews*.

Test to break it, not demonstrate it.

One manages "starts", not "finishes". You react to finishes.

Beware of the student syndrome.

Noah's Principle: Predicting rain doesn't count, building arks does.

There is no such thing as a constant (except maybe π and e).

GUI design should be viewed as a religious preference, not technical. However, it is important that you express your beliefs.

Why are you showing me that slide?

Groups with full in-boxes are invariably keeping up. Flush them (the boxes, not the groups).

Declaring victory (or the contract's changes clause) is a manager's best friend.

Violently reject any attempts to "save money" by not fixing defects

Be brutal in your classification of a "bug". "Better" is not a bug.

We now have at least two generations of staff that have *all* been told their entire lives that they were above average.

Tailor your behavior to the software, not vice versa.

I've never found the software that I'd rather write than buy.

When you are down to a short list, what matters is that you choose and get on with it.

Index

F

G

H

I

J

K

L

M

N

O

The Technical Manager's Survival Guides

MANAGING SYSTEMS DEVELOPMENT 101

A Guide to Designing Effective Commercial Products for Engineers and Their Bosses/CEOs

By James T. Karam

This book provides specific, practical advice for engineers who are advancing beyond their technical specialty and find themselves working with other specialties necessary to the development of a complex system or product. They continually face a variety of issues that were invariably never addressed in their schooling: dealing with specifications, project plans, and budgets; improving quality by working with "downstream" functions such as production and service; resolving incompatibilities and bugs under a variety of test conditions; providing technical direction and reviews; and more.

Based on "lessons learned" by the author over a forty-year career developing complex system products, the book presents basic principles that are applicable whether you are in a bureaucratic, multi-national corporation or one with the founder still in control. Regardless of the organization's size or the particular products, the engineering management issues are eerily the same. Systems are systems, and the engineering process basics, derived largely from aerospace, remain the key to success anywhere.

Chapter titles typically end in "101" because the basics are usually where the problems lie, as well as their solutions. This book is concise, but the content is dense. As such, it will also provide a succinct refresher for more experienced professionals and their management when facing challenges.