FUNDAMENTALS OF SOFTWARE ENGINEERING PROJECT MANAGEMENT

Johan Gouws

B.Eng. & M.Eng. (Elec.) (Rand Afrikaans University, South Africa)
MBA (Heriot-Watt University, Scotland)
Ph.D. (Wageningen, the Netherlands)

Leonie E. Gouws

B.Eng. (Mech.) (Rand Afrikaans University, South Africa)
M.Eng. (Engineering Management) (Rand Afrikaans University, South Africa)

Disclaimers

Melikon Pty Ltd published this work as a contribution to the education of software engineering project managers and other people involved in software development. The material herein is for general information only and Melikon cannot be held liable for any actions taken or not taken on the basis of material contained herein.

Published by: Melikon Pty Ltd
Date: June 2004
Re-Published by: Feed Forward Publications (www.feedforward.com.au)
Date: October 2006
Issue: 3.4

First edition published in 1999.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 SOFTWARE'S GROWING IMPORTANCE

- The days when computer software development could be handled as part of *documentation* or *general* on the agenda of a project meeting (if handled at all), are long gone.
- For many engineering and other projects, software has become the pivotal part: it controls generation and distribution of electricity; water purification and distribution; robotic systems in production plants; vehicles, their engines, and traffic flows; household equipment; aircraft, air traffic, and passenger bookings; telecommunications; logistics; spacecraft and space missions, etc., etc.
- Software also plays an ever-increasing role in business management: it controls equipment maintenance management, logistics, resource allocations, business processes, financial transactions, accounting, communication, human resources, etc.
- Because of software's growing importance, its development must be managed even more carefully than other areas of large projects.
- Often, in the past, software was *randomly assembled* – almost in an artistic way.
- This approach is no longer appropriate; and the departure point for proper software development should be the realization that software development has grown from an art, to a craft, to a proper engineering discipline.
- From this departure point follows:
  - Software is a product (although a rather "fluid" one), like any other result of engineering methodologies.
  - Software development needs the structured application of scientific and engineering principles in order to analyse, design, construct, document and maintain it.
  - Like any engineering development, large-scale software development also requires the disciplined application of project management principles.

1.2 MANAGING SOFTWARE DEVELOPMENT

- Any project "stands" on three legs: cost, schedule and functionality (performance).

- These three *project legs* must be balanced, planned in advance, and managed throughout the project's lifetime in order to have a successful project.
- Rigorous application of proper project management techniques on a software development project greatly improves balancing of the three project legs, and the chances of project success.
- "Proper project management" involves *planning, organizing, staffing, directing* and *control* of the project.
- To do all these things successfully, a project manager must have good technical-, management-, and people skills.
- Over- or under emphasis of any of the project-, management- or skills elements will certainly result in a failed project.

1.3 PURPOSE AND SCOPE OF THIS BOOK

- Merging the application of *structured engineering* with that of *disciplined project management* for software development, results in the concept *software engineering project management* – and that is the focus of this book.
- This is <u>not</u> a programming-, or software-, or engineering book - but instead it is a book aimed at **<u>introducing</u> project management principles for software development**.
- The intention with this book is not to instantly convert readers into the world's leading managers for software development projects, nor to provide *recipes* or *quick-fix solutions* – but it is specifically aimed at:
  - Making **Software Engineering Project Managers** more aware of a variety of available project management techniques; and helping them to plan, organize, staff, direct and control software development projects.
  - Providing **Line Managers** (especially Engineering- and Marketing Managers) with a better understanding of the major issues involved in managing a software development project.
  - Giving **Software Developers** (e.g. Designers, Programmers, and Testers) an opportunity to gain a better understanding of:
    - Inputs required from them by their project managers, in order to enable the project managers to better manage complex software development.

  - Their own roles in a software development project.
  - Management techniques for improved software development.
  - *Behind the scenes issues* dealt with by software engineering project managers (see why they sometimes seem to act so "*foolishly*").

- Since the book aims to maintain a balance between *width* and *depth* of the presentation, there will certainly be some aspects which might require further reading.
- Literature references are provided throughout the text, to enable readers to research and study specific topics further.
- Some "review activities" are included:
  - These are typically in the form of a table, where choices can be ticked off, based on the topics covered in the preceding sections.
  - The intention is that readers can use these to link the concepts presented, with the actual situations in their own organizations.
- The remainder of the book is also in the point-wise format used in this introductory chapter. This format is used in order to make the book an easy-to-read reference source.

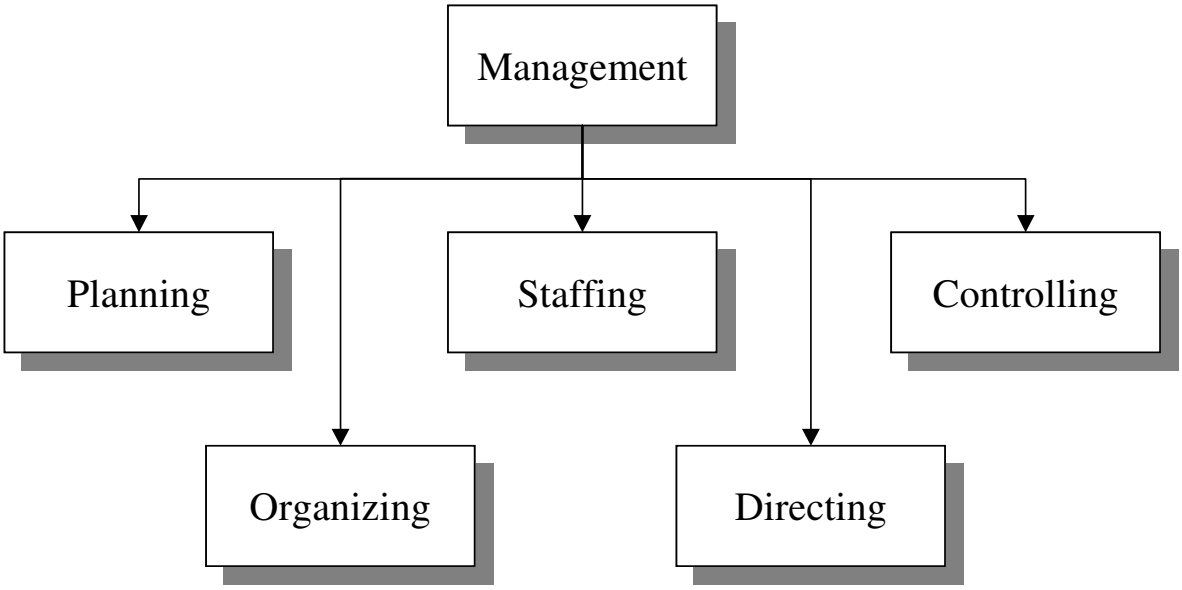

Figure 1: Elements of Management

- **Organizing** involves:
  - Establishing a structure to be filled by people, aimed at reaching the defined goals and objectives.
  - Defining job content, interfaces, responsibilities, authority, and resource allocation.
- **Staffing** involves:
  - Filling the positions in the organizational structure with suitable people.
  - Keeping the positions filled, in order to execute the plan.
- **Directing** (or **Leading**) involves:
  - Creating an environment in which individuals, working together in groups, can accomplish well-selected aims.
  - Influencing people to contribute to reaching the goals and objectives.
  - Using leadership styles, communication, conflict resolution, delegation, etc. in order to overcome the problems arising from *people issues* (attitudes, desires, motivations, behaviour in groups, etc.) on a project.
- **Controlling** (and co-ordination) involves:
  - Measuring actual performance.
  - Comparing actual- with desired results and implementing corrective actions – e.g. by controlling the actions of the people doing the work.

in **Figure 13**.

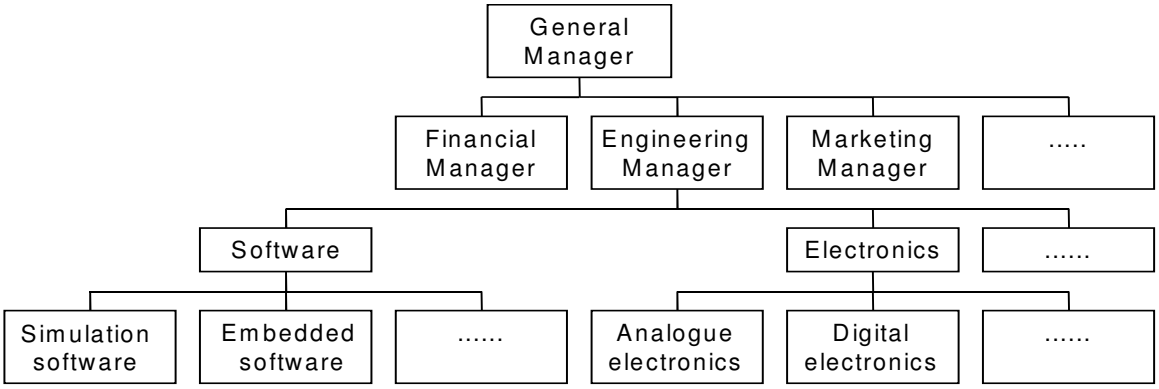

Figure 13: Example of a Functional Structure

- This structure keeps specialists together; and it fragments complex problems and assigns it to different departments.
- Typical functions: Engineering, Finance, Marketing, Production, etc. - each of which can be further divided into sub-functions, as shown in **Figure 13**.
- Typical advantages:
  - Centralisation of similar resources and control thereof.
  - Simplified training.
  - No duplication of similar responsibilities on different projects.
- Typical disadvantages:
  - Prioritising different assignments when more than one project is done.
  - The functional group's goals can become more important than the overall organization's.
  - Group members get limited experience.
  - Progress reporting can be difficult, especially when staff members are reporting on different projects simultaneously.
- Organizations experiencing difficulties often discard their functional structures in favour of project- or matrix structures.
- However, it is essential to first analyse the reasons for difficulties before going for a new structure.
- Lack of proper communication is often the biggest problem with the functional

novices can consistently handle routine designs. No such handbook yet exists for software, so mistakes are often repeated on project after project, year after year.

Activity 24: Estimate the following for your team / organization:

Software development productivity (number of fully documented and commented lines of code per person-year)	
Productivity ratio (top : poorest performers in the team)	

5.4.2 Back to the *COCOMO* Formulae

- The COCOMO equations were used in the previous chapter to estimate effort and schedule.
- These formulae can now be extended – as shown in Table 17 – in order to also estimate the average number of software staff necessary.
- The following symbols are used:
  - PM = person-months
  - KDSI = delivered source instructions, in thousands ("kilo")
  - TD = number of months for software development
  - ANS = Average number of staff

Table 17: COCOMO Basic Equations for Estimating Staff Requirements

Type	Effort	Schedule	Staffing
Organic	$PM = 3.6 (KDSI)^{1.05}$	$TD = 2.5 (PM)^{0.38}$	ANS = PM / TD
Semi-detached	$PM = 3.0 (KDSI)^{1.12}$	$TD = 2.5 (PM)^{0.35}$	ANS = PM / TD
Embedded	$PM = 2.4 (KDSI)^{1.20}$	$TD = 2.5 (PM)^{0.32}$	ANS = PM / TD

Reason for Problems with Project Control	Ranking by	
	Project Managers	Line Managers
Technical complexities	2	10
Unrealistic project plan	3	2
Staffing problems	4	9
Inability to detect problems early	5	7
Priority shifts	6	11
Sinking team spirit	7	14
Project Scope underestimated	8	3
Insufficient number of checkpoints	9	8
Insufficient project planning	10	1
No commitments by staff to plan	11	12
Uncooperative support groups	12	13
Inability to track progress	13	6
Insufficient contingency planning	14	5
Unqualified project personnel	15	15

7.24 STEPS FOR MORE EFFECTIVE PROJECT CONTROL

- Break the overall program into phases and subsystems (WBS).
- Clearly define objectives, results and deliverables.
- Define measurable milestones and quantitative checkpoints.
- Obtain commitment from all team members and management.
- Ensure that different teams can work together, and that outputs are compatible.
- Project tracking - ensure proper project control.
- Ensure measurability of progress parameters.
- Hold regular reviews of project goals, plans, progress, etc.
- Ensure interesting work to maintain interest (take personal preferences into account).
- Communication, communication, communication!!
- Leadership - making people feel strong and in control.
- Minimise threats - manage conflict and power struggles, avoid surprises (up or down) and unrealistic demands, foster mutual trust.
- Design an appropriate personnel appraisal and reward system.
- Assure continuous senior management involvement, endorsement and support.
- Personal drive - project manager must be enthusiastic about the project.
- Note all problems experienced in the project database, for future reference.
- Successful project management requires a proper project plan, commitment from