
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.

Melikon Pty Ltd holds the publishing rights and the copyright of this work. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval systems, without prior written permission from Melikon.

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.

© Copyright – Melikon 2004

TABLE OF CONTENTS

1. INTRODUCTION.....	9
1.1 SOFTWARE'S GROWING IMPORTANCE	9
1.2 MANAGING SOFTWARE DEVELOPMENT	9
1.3 PURPOSE AND SCOPE OF THIS BOOK	10
2. MANAGING SOFTWARE PROJECTS	12
2.1 FUNDAMENTALS OF MANAGEMENT	12
2.2 THE ELEMENTS OF MANAGEMENT	12
2.3 THE UNIVERSALITY OF MANAGEMENT.....	14
2.4 PREREQUISITES FOR EFFECTIVE MANAGEMENT	15
2.5 MANAGEMENT AS A FEEDBACK SYSTEM.....	18
2.6 DIFFERENT MANAGEMENT APPROACHES	20
2.7 COMMUNICATION AND DOCUMENTATION.....	20
2.8 PROJECT MANAGEMENT FUNDAMENTALS	22
2.9 RISK MANAGEMENT FOR PROJECTS.....	22
2.9.1 Introduction.....	22
2.9.2 Risk Identification.....	23
2.9.3 Risk Analysis	24
2.9.4 Options for Risk Alleviation	25
2.9.5 Cost of Risk Abatement	26
2.10 A CHECKLIST FOR PROJECT MANAGEMENT	26
3. INTRODUCTION TO SOFTWARE ENGINEERING.....	29
3.1 IS SOFTWARE AN ENGINEERING DISCIPLINE?	29
3.2 THE "SOFTWARE CRISIS"	30
3.2.1 Typical Software Development Problems.....	30
3.2.2 Case Studies	31
3.2.3 Software Development Status.....	32
3.2.4 The Quest for First-Time-Right Software.....	33
3.3 FUTURE SOFTWARE DEVELOPMENT TRENDS	35
3.3.1 Software Development Categories.....	35
3.3.2 Software Factories.....	36
3.4 SOFTWARE PROCESS MODELS.....	37
3.4.1 Introduction.....	37
3.4.2 Linear Sequential Models.....	38
3.4.3 Waterfall Model.....	41
3.4.4 The V-Model	42
3.4.5 Prototyping Model.....	42
3.4.6 Incremental / Evolutionary Model.....	43
3.4.7 Concurrent Model	44
3.5 DEVELOPMENT POLICIES / METHODOLOGIES	45
3.5.1 Objectives of Software Development Policies and Methodologies	45
3.5.2 Typical Benefits of Software Development Policies.....	46
3.5.3 Typical Aspects Defined by Software Development Policies.....	46

3.6	USING NEW TECHNOLOGY FOR SOFTWARE PROJECTS.....	47
3.6.1	Introduction.....	47
3.6.2	Expected Future Practices.....	48
3.6.3	General Advice on Using New Technology.....	49
3.6.4	Inserting New Technologies into a Software Project.....	49
3.7	THE CAPABILITY MATURITY MODEL.....	50
3.7.1	Background.....	50
3.7.2	Five Levels of Software Process Maturity.....	51
3.7.3	Examples of Practical Application of CMM.....	52
4.	PLANNING A SOFTWARE PROJECT	55
4.1	INTRODUCTION.....	55
4.2	PROJECT PLANNING.....	55
4.2.1	Major Problems in Project Planning.....	55
4.2.2	The Two Pillars of Project Planning.....	57
4.2.3	Summary of Project Planning Activities.....	58
4.3	REQUIREMENTS DEFINITION.....	59
4.3.1	Goal of the Requirements Definition Phase.....	59
4.3.2	Perspectives on Software Requirements.....	59
4.3.3	Effects of Improper Requirements Definition.....	60
4.3.4	Proper Requirements Definition.....	61
4.3.5	Methods to Aid Requirements Definition.....	65
4.3.6	Essentials of Requirements Definition.....	67
4.4	WORK BREAKDOWN.....	68
4.4.1	Introduction.....	68
4.4.2	Types of Work Breakdown Structures.....	68
4.4.3	Developing a WBS.....	71
4.5	SOFTWARE EFFORT ESTIMATION.....	72
4.5.1	Introduction.....	72
4.5.2	Common Estimation Techniques.....	72
4.5.3	Major Problems in Estimating Software Effort.....	73
4.5.4	Experience-Based Estimation Methods.....	74
4.5.5	Empirical Estimation Methods.....	75
4.5.6	Improving Estimation Accuracy.....	75
4.6	THE CONSTRUCTIVE COST MODEL (COCOMO).....	76
4.6.1	Introduction.....	76
4.6.2	COCOMO Definitions and Assumptions.....	77
4.6.3	COCOMO Software Development Types.....	78
4.6.4	Basic COCOMO.....	78
4.6.5	COCOMO for Software Maintenance.....	79
4.6.6	Adapting COCOMO to Own Needs.....	80
4.6.7	Later versions of COCOMO.....	80
4.7	ESTIMATING SOFTWARE SYSTEM SIZE.....	81
4.7.1	Introduction.....	81
4.7.2	Function Block Counting.....	82
4.7.3	Function Points.....	82
4.7.4	Object Points.....	85
4.7.5	Software Size Growth.....	86

4.7.6	<i>Size Estimation for Reuse and Breakage</i>	87
4.8	STEPS FOR IMPROVING ESTIMATION ACCURACY	87
4.9	PROJECT SCHEDULING	88
4.9.1	<i>Introduction</i>	88
4.9.2	<i>Scheduling Techniques</i>	89
4.9.3	<i>Comparison of Scheduling Techniques</i>	91
4.10	ESTIMATING SOFTWARE PROJECT SCHEDULES	92
4.10.1	<i>Introduction</i>	92
4.10.2	<i>Estimating the Development Schedule</i>	93
4.10.3	<i>Impact of Reused Code on Development Schedule</i>	94
4.10.4	<i>Exchanging People and Months</i>	95
4.11	SCHEDULE SLIP	96
4.11.1	<i>An Unstable System</i>	96
4.11.2	<i>Project On Schedule until Testing Begins</i>	97
4.11.3	<i>Does Project Slip Matter?</i>	98
4.12	USING RULES OF THUMB FOR PROJECT ESTIMATION	100
4.12.1	<i>Introduction</i>	100
4.12.2	<i>Why Bother With Rules Of Thumb?</i>	101
4.12.3	<i>General Rules of Thumb</i>	101
4.12.4	<i>Rules of Thumb for Schedule and Effort</i>	104
4.12.5	<i>Cautionary Remarks on Rules of Thumb</i>	105
5.	ORGANIZING AND STAFFING A SOFTWARE PROJECT	106
5.1	INTRODUCTION	106
5.2	ORGANIZATIONAL STRUCTURES	107
5.2.1	<i>Introduction</i>	107
5.2.2	<i>Functional Structures</i>	107
5.2.3	<i>Project Structures</i>	109
5.2.4	<i>Matrix Structures</i>	110
5.3	SOFTWARE TEAM STRUCTURES	115
5.3.1	<i>Introduction</i>	115
5.3.2	<i>The Democratic Decentralised (Weinberg) Team</i>	115
5.3.3	<i>The Controlled Centralised (Baker) Team</i>	115
5.3.4	<i>The Controlled Decentralised Team Structure</i>	116
5.3.5	<i>Choosing a Team Structure</i>	116
5.4	NUMBER OF SOFTWARE STAFF REQUIRED	118
5.4.1	<i>Software Development Productivity</i>	118
5.4.2	<i>Back to the COCOMO Formulae</i>	119
5.4.3	<i>The Rayleigh Function</i>	120
5.4.4	<i>Rule of Thumb for Software Project Staffing</i>	123
5.5	JOB ANALYSIS	124
5.5.1	<i>Reasons for Job Analysis</i>	124
5.5.2	<i>Defining the Job Type and Job Title</i>	125
5.5.3	<i>Determining Job Content</i>	125
5.5.4	<i>Objective Assessment of the Vacancy</i>	127
5.5.5	<i>The Bigger Picture</i>	128
5.6	JOB SPECIFICATION	128
5.7	STAFFING A SOFTWARE PROJECT	129

5.8	RECRUITMENT	130
5.8.1	<i>Recruiting Suitable Candidates</i>	130
5.8.2	<i>Important Aspects of Recruiting</i>	131
5.9	SELECTING THE RIGHT PEOPLE	131
5.9.1	<i>Introduction</i>	131
5.9.2	<i>Profile of the Software Developer</i>	132
5.9.3	<i>Selection Tools</i>	132
5.9.4	<i>Different Interview Methods</i>	133
5.9.5	<i>Guidelines for Conducting a Structured Interview</i>	134
5.10	APPOINTMENT AND KEEPING THE POSITIONS FILLED	136
5.10.1	<i>The Initial Days</i>	136
5.10.2	<i>Staff Turnover</i>	136
5.10.3	<i>Suitability-Replaceability Matrix</i>	137
5.10.4	<i>Career Planning</i>	139
5.11	PERFORMANCE APPRAISAL	140
5.11.1	<i>Purpose of Performance Appraisals</i>	140
5.11.2	<i>Ability and Willingness</i>	140
5.11.3	<i>Factors Influencing Job Performance</i>	141
5.11.4	<i>Important Issues in Performance Appraisal</i>	142
5.11.5	<i>Guidelines for Effective Performance Appraisal</i>	142
5.12	TRAINING	144
5.12.1	<i>Importance of Training</i>	144
5.12.2	<i>Important Issues in Training</i>	145
5.12.3	<i>Training To Help New Employees Settle In</i>	145
6.	DIRECTING A SOFTWARE PROJECT	147
6.1	INTRODUCTION	147
6.2	LEADERSHIP AND EFFECTIVE USE OF POWER	147
6.2.1	<i>Introduction</i>	147
6.2.2	<i>Types of Power</i>	147
6.2.3	<i>Making People Feel in Control</i>	148
6.2.4	<i>Goal-Oriented Behaviour</i>	148
6.2.5	<i>Assess Your Own Situation as a Leader</i>	149
6.3	DELEGATION	149
6.3.1	<i>Introduction</i>	149
6.3.2	<i>Helping Employees Develop</i>	150
6.3.3	<i>Making Delegation Easier</i>	151
6.3.4	<i>Determining What and How Much to Delegate</i>	151
6.4	MOTIVATING AND KEEPING SOFTWARE DEVELOPERS	153
6.4.1	<i>Introduction</i>	153
6.4.2	<i>Creating Enthusiasm and a Shared Vision</i>	153
6.4.3	<i>Using People's Intelligence</i>	155
6.4.4	<i>Rewards versus Incentives</i>	156
6.4.5	<i>Hygienes versus Motivators</i>	157
6.5	CONFLICT MANAGEMENT	158
6.5.1	<i>Introduction</i>	158
6.5.2	<i>Sources of Conflict</i>	159
6.5.3	<i>Different Perceptions on Conflict</i>	160

6.5.4	<i>Dealing with Conflict</i>	161
7.	CONTROLLING A SOFTWARE PROJECT	163
7.1	INTRODUCTION	163
7.2	TWO BRANCHES OF PROJECT CONTROL	163
7.3	FEEDBACK CONTROL FOR PROJECT MANAGEMENT	164
7.4	STATUS- VERSUS ACTION INFORMATION	165
7.5	MEASUREMENT FOR COST AND PROGRESS CONTROL	165
7.5.1	<i>Binary Reporting</i>	165
7.5.2	<i>Budgeted- versus Actual Effort</i>	166
7.6	PROJECT DOCUMENTATION	168
7.6.1	<i>Importance of Project Documentation</i>	168
7.6.2	<i>Document Reviews</i>	169
7.6.3	<i>Types of Project Documents</i>	170
7.7	SOFTWARE PROJECT MANAGEMENT PLAN	170
7.7.1	<i>Scope and Purpose of a Project Management Plan</i>	170
7.7.2	<i>Framework for a Project Management Plan</i>	171
7.8	THE UNIT DEVELOPMENT FOLDER (UDF)	173
7.8.1	<i>Scope and Purpose of the Unit Development Folder</i>	173
7.8.2	<i>Framework for a Unit Development Folder</i>	174
7.9	SOFTWARE QUALITY ASSURANCE	177
7.10	IMPROVING SOFTWARE QUALITY	178
7.11	TECHNIQUES FOR SOFTWARE QUALITY CONTROL	178
7.11.1	<i>Failure Analysis</i>	178
7.11.2	<i>Early Defect Removal</i>	179
7.11.3	<i>Adherence to Task Entry / Exit Criteria</i>	180
7.11.4	<i>Software Testing</i>	180
7.11.5	<i>Other Techniques</i>	181
7.12	SOFTWARE CONFIGURATION MANAGEMENT	181
7.13	BASELINE IDENTIFICATION	182
7.14	SOFTWARE CONFIGURATION CONTROL	183
7.15	SOFTWARE CONFIGURATION AUDITING	184
7.16	SOFTWARE PROJECT AUDITS	185
7.16.1	<i>Important Issues in Software Project Audits</i>	186
7.16.2	<i>Things to Look for During an Audit</i>	187
7.16.3	<i>Where and How To Do An Audit</i>	188
7.16.4	<i>Who Must Do An Audit?</i>	188
7.16.5	<i>Tools For The Audit Team</i>	188
7.17	SOFTWARE PEER REVIEWS	189
7.18	SOFTWARE INSPECTION	189
7.18.1	<i>Principal Steps in the Software Inspection Process</i>	190
7.18.2	<i>Data Collection</i>	191
7.18.3	<i>Implementing Inspections</i>	191
7.19	SOFTWARE WALK-THROUGHS	191
7.20	OTHER REVIEW PROCESSES	192
7.21	BENEFITS AND COSTS OF PEER REVIEWS	194
7.22	SOFTWARE METRICS	196

7.22.1	Definitions.....	196
7.22.2	Successful Use of Metrics	197
7.22.3	Main Categories of Metrics.....	197
7.22.4	Sub-Categories of Metrics	198
7.22.5	Implementation of Metrics.....	198
7.22.6	Typical Metrics Implementation Problems.....	200
7.22.7	Examples of Metrics	202
7.23	REASONS FOR PROJECT CONTROL PROBLEMS	208
7.24	STEPS FOR MORE EFFECTIVE PROJECT CONTROL.....	210
8.	CONCLUSION	212
8.1	IMPORTANCE OF PROJECT MANAGEMENT	212
8.2	SETTING THE SCENE FOR A SOFTWARE PROJECT.....	213
8.3	PLANNING A SOFTWARE PROJECT	215
8.4	ORGANISING A SOFTWARE PROJECT.....	217
8.5	STAFFING A SOFTWARE PROJECT.....	218
8.6	DIRECTING A SOFTWARE PROJECT.....	219
8.7	CONTROLLING A SOFTWARE PROJECT.....	220
9.	LITERATURE REFERENCES	223

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 not a programming-, or software-, or engineering book - but instead it is a book aimed at **introducing 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.

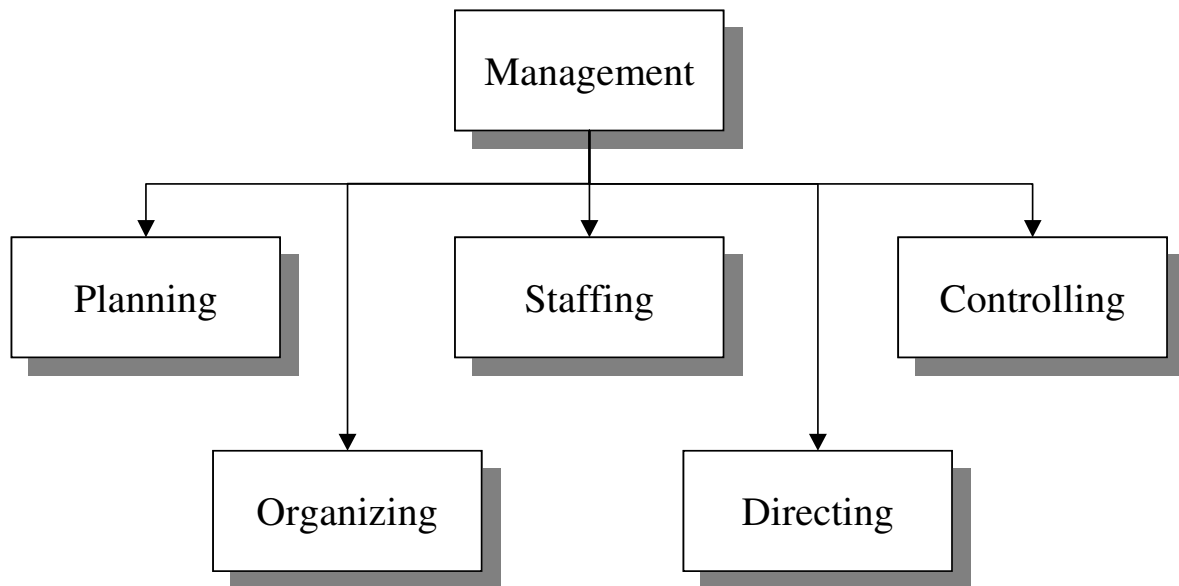


Figure 1: Elements of Management

- **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**.

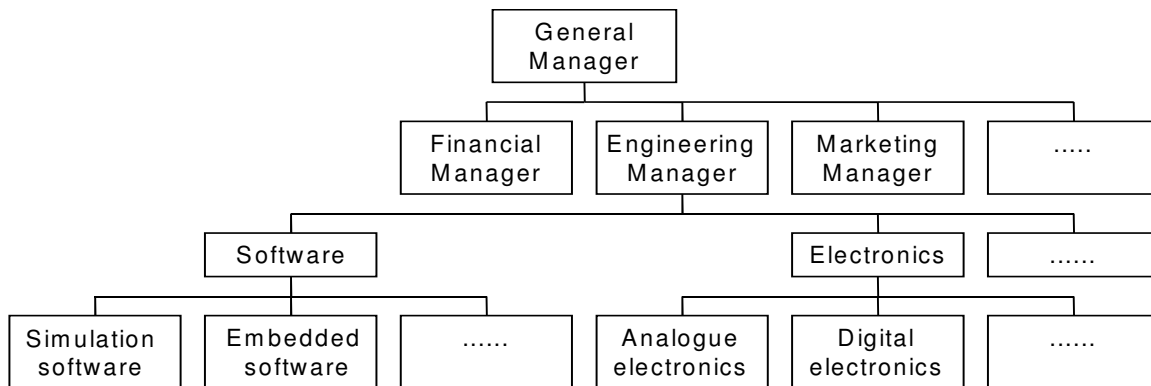


Figure 13: Example of a Functional Structure

- 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