ESA PSS-05-08 Issue 1 Revision 1
March 1995

Guide to software project management

Prepared by:
ESA Board for Software
Standardisation and Control
(BSSC)

Approved by:

The Inspector General, ESA

european space agency / agence spatiale européenne
8-10, rue Mario-Nikis, 75738 PARIS CEDEX, France

DOCUMENT STATUS SHEET

DOCUMENT STATUS SHEET			
1. DOCUMENT TITLE: ESA PSS-05-08 Guide to Software Project Management			
2. ISSUE	3. REVISION	4. DATE	5. REASON FOR CHANGE
1	0	1994	First issue
1	1	1995	Minor updates for publication

Issue 1 Revision 1 approved, May 1995
Board for Software Standardisation and Control
M. Jones and U. Mortensen, co-chairmen

Issue 1 approved, 15th June 1995
Telematics Supervisory Board

Issue 1 approved by:
The Inspector General, ESA

Published by ESA Publications Division,
ESTEC, Noordwijk, The Netherlands.
Printed in the Netherlands.
ESA Price code: E1
ISSN 0379-4059

TABLE OF CONTENTS

This page is intentionally left blank

PREFACE

This document is one of a series of guides to software engineering produced by the Board for Software Standardisation and Control (BSSC), of the European Space Agency. The guides contain advisory material for software developers conforming to ESA's Software Engineering Standards, ESA PSS-05-0. They have been compiled from discussions with software engineers, research of the software engineering literature, and experience gained from the application of the Software Engineering Standards in projects.

Levels one and two of the document tree at the time of writing are shown in Figure 1. This guide, identified by the shaded box, provides guidance about implementing the mandatory requirements for software project management described in the top level document ESA PSS-05-0.

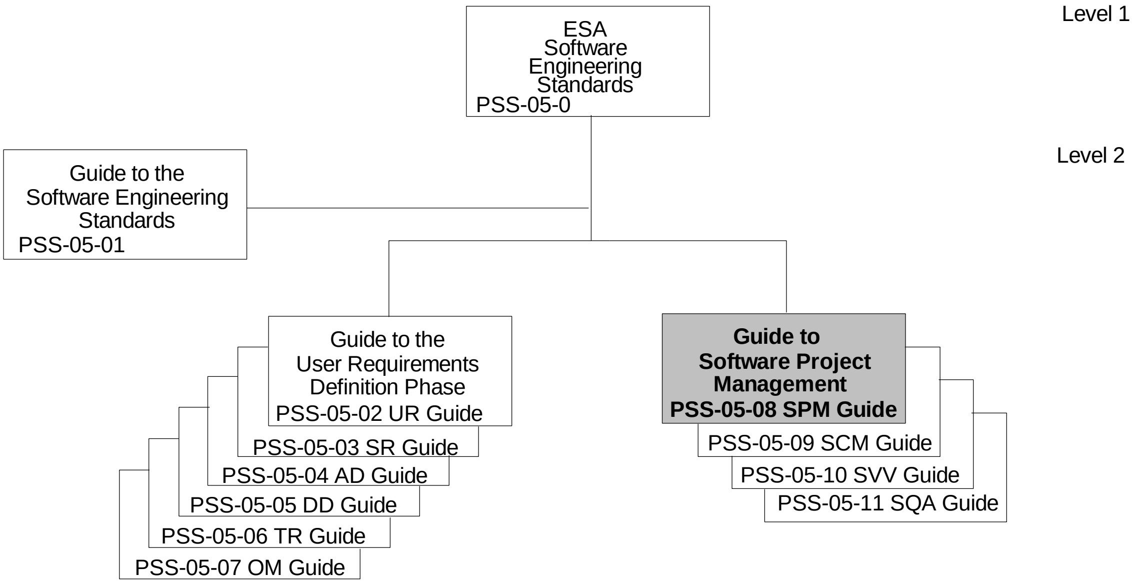

Level 1
Level 2
ESA Software Engineering Standards PSS-05-0
Guide to the Software Engineering Standards PSS-05-01
Guide to the User Requirements Definition Phase PSS-05-02 UR Guide
PSS-05-03 SR Guide
PSS-05-04 AD Guide
PSS-05-05 DD Guide
PSS-05-06 TR Guide
PSS-05-07 OM Guide
Guide to Software Project Management PSS-05-08 SPM Guide
PSS-05-09 SCM Guide
PSS-05-10 SVV Guide
PSS-05-11 SQA Guide

Figure 1: ESA PSS-05-0 document tree

The Guide to the Software Engineering Standards, ESA PSS-05-01, contains further information about the document tree. The interested reader should consult this guide for current information about the ESA PSS-05-0 standards and guides.

The following past and present BSSC members have contributed to the production of this guide: Carlo Mazza (chairman), Gianfranco Alvisi, Michael Jones, Bryan Melton, Daniel de Pablo and Adriaan Scheffer.

The BSSC wishes to thank Jon Fairclough for his assistance in the development of the Standards and Guides, and to all those software engineers in ESA and Industry who have made contributions.

Requests for clarifications, change proposals or any other comment concerning this guide should be addressed to:

BSSC/ESOC Secretariat
Attention of Mr M Jones
ESOC
Robert Bosch Strasse 5
D-64293 Darmstadt
Germany

BSSC/ESTEC Secretariat
Attention of Mr U Mortensen
ESTEC
Postbus 299
NL-2200 AG Noordwijk
The Netherlands

CHAPTER 1
INTRODUCTION

1.1 PURPOSE

ESA PSS-05-0 describes the software engineering standards to be applied for all deliverable software implemented for the European Space Agency (ESA) [Ref 1].

ESA PSS-05-0 requires that every software project be planned, organised, staffed, led, monitored and controlled. These activities are called 'Software Project Management' (SPM). Each project must define its Software Project Management activities in a Software Project Management Plan (SPMP).

This guide defines and explains what software project management is, provides guidelines on how to do it, and defines in detail what a Software Project Management Plan should contain.

This guide should be read by software project managers, team leaders, software quality assurance managers, senior managers and initiators of software projects.

1.2 OVERVIEW

Chapter 2 contains a general discussion of the principles of software project management, expanding upon ESA PSS-05-0. Chapter 3 discusses methods for software project management that can be used to support the activities described in Chapter 2. Chapter 4 discusses tools for software project management. Chapter 5 describes how to write the SPMP. Chapter 6 discusses progress reporting.

All the mandatory practices in ESA PSS-05-0 relevant to software project management are repeated in this document. The identifier of the practice is added in parentheses to mark a repetition. This document contains no new mandatory practices.

This page is intentionally left blank

CHAPTER 2
SOFTWARE PROJECT MANAGEMENT

2.1 INTRODUCTION

Software project management is 'the process of planning, organising, staffing, monitoring, controlling and leading a software project' [Ref 3]. Every software project must have a manager who leads the development team and is the interface with the initiators, suppliers and senior management. The project manager:

- produces the Software Project Management Plan (SPMP);
- defines the organisational roles and allocates staff to them;
- controls the project by informing staff of their part in the plan;
- leads the project by making the major decisions and by motivating staff to perform well;
- monitors the project by measuring progress;
- reports progress to initiators and senior managers.

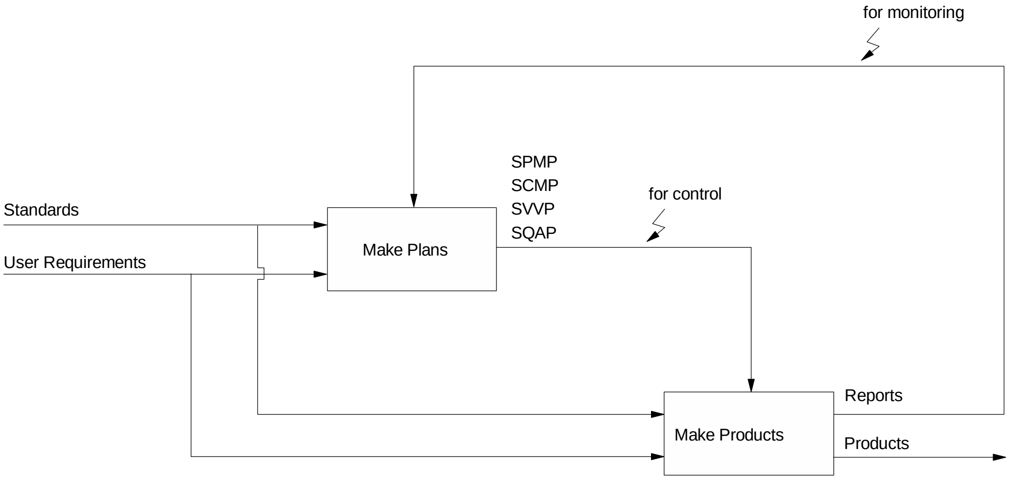


Figure 2.1: Management control loop

Figure 2.1 shows the control and monitoring loop required in every software project. Standards and user requirements are the primary input to both the planning and production processes. Plans are made for project management, configuration management, verification and validation and quality assurance. These plans control production. Reports, such as progress reports, timesheets, work package completion reports, quality assurance reports and test reports provide feedback to the planning process. Plans may be updated to take account of these reports.

The project manager is the driving force in the management control loop. The following sections define the role of the project manager and discuss project management activities.

2.2 THE PROJECT MANAGER'S ROLE AND RESPONSIBILITIES

When they are appointed, project managers should be given terms of reference that define their:

- objectives;
- responsibilities;
- limits of authority.

The objective of every project manager is to deliver the product on time, within budget and with the required quality. Although the precise responsibilities of a project manager will vary from company to company and from project to project, they should always include planning and forecasting. Three additional areas of management responsibility defined by Mintzberg [Ref 17] are:

- interpersonal responsibilities, which include:
    - leading the project team;
    - liaising with initiators, senior management and suppliers;
    - being the 'figurehead', i.e. setting the example to the project team and representing the project on formal occasions.
- informational responsibilities, which include:
    - monitoring the performance of staff and the implementation of the project plan;
    - disseminating information about tasks to the project team;
    - disseminating information about project status to initiators and senior management;
    - acting as the spokesman for the project team.

- decisional responsibilities, which include:
  - allocating resources according to the project plan, and adjusting those allocations when circumstances dictate (i.e. the project manager has responsibility for the budget);
  - negotiating with the initiator about the optimum interpretation of contractual obligations, with the company management for resources, and with project staff about their tasks;
  - handling disturbances to the smooth progress of the project such as equipment failures and personnel problems.

2.3 PROJECT INTERFACES

Project managers must identify the people or groups the project deals with, both within the parent organisation and outside. A project may have interfaces to:

- initiators;
- end users;
- suppliers;
- subcontractors;
- the prime contractor;
- other subsystem developers.

When defining external project interfaces, the project manager should:

- ensure that a single, named point of contact exists both within the project team and each external group;
- channel all communications between the project and external groups through as few people as possible;
- ensure that no person in the project has to liaise with more than seven external groups (the 'rule of seven' principle).

2.4 PROJECT PLANNING

Whatever the size of the project, good planning is essential if it is to succeed. The software project planning process is shown in Figure 2.4. The five major planning activities are:

- define products;
- define activities;
- estimate resources and duration;
- define activity network;
- define schedule and total cost.

This process can be applied to a whole project or to a phase of a project. Each activity may be repeated several times to make a feasible plan. In principle, every activity in Figure 2.4 can be linked to the other activities by 'feedback' loops, in which information gained at a later stage in planning is used to revise earlier planning decisions. These loops have been omitted from the figure for simplicity. Iteration is essential for optimising the plan.

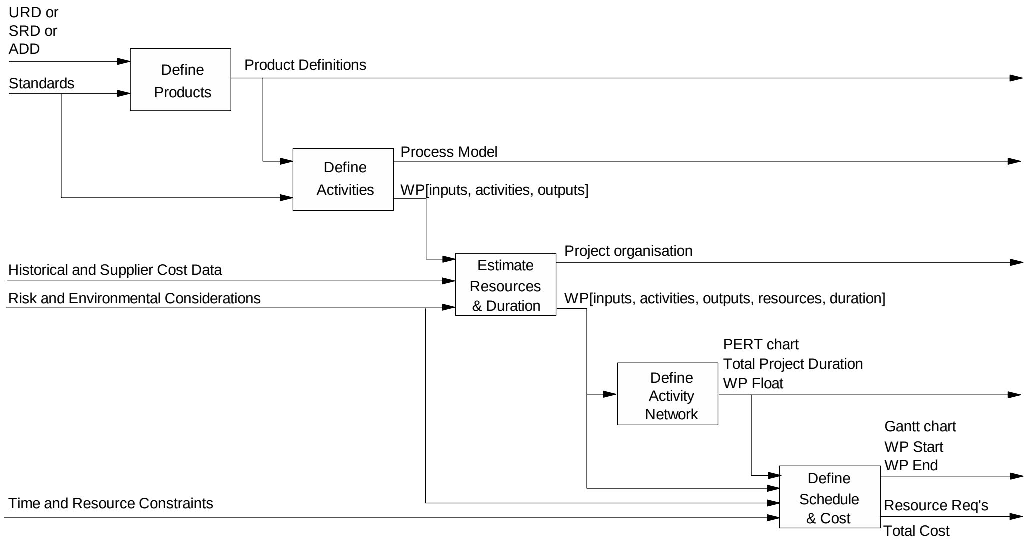


Figure 2.4: Planning process

The inputs to software project planning are:

- User Requirements Document (URD), Software Requirements Document (SRD) or Architectural Design Document (ADD), according to the project phase;
- software standards for products and procedures;
- historical data for estimating resources and duration;
- supplier cost data;
- risks to be considered;
- environmental factors, such as new technology;
- time constraints, such as delivery dates;
- resource constraints, such as the availability of staff.

The primary output of project planning is the Software Project Management Plan. This should contain:

- definitions of the deliverable products;
- a process model defining the life cycle approach and the methods and tools to be used;
- the work breakdown structure, i.e. a hierarchically structured set of work packages defining the work;
- the project organisation, which defines the roles and reporting relationships in the project;
- an activity network defining the dependencies between work packages, the total time for completion of the project, and the float times of each work package;
- a schedule of the project, defining the start and end times of each work package;
- a list of the resources required to implement the project;
- a total cost estimate.

The following subsections describe the five major activities.

2.4.1 Define products

The first activity in project planning is to define the products to be delivered. Table 2.4.1 summarises the software products of each phase as required by ESA PSS-05-0.

Phase	Input Product	Output Product
SR	URD	SRD
AD	SRD	ADD
DD	ADD	DDD, code, SUM
TR	DDD, code, SUM	STD

Table 2.4.1: Phase input and output products

While standards define the contents of documents, some consideration should be given as to how each section of the document will be prepared. Care should be taken not to overconstrain the analysis and design process in defining the products. Some consideration should also be given to the physical characteristics of the products, such as the medium (e.g. paper or computer file), language (e.g. English, French) or programming language (in the case of code).

2.4.2 Define activities

Once the products are specified, the next step is to define a process model and a set of work packages.

2.4.2.1 Process model

A software process model should define the:

- activities in a software development process;
- the inputs and outputs of each activity;
- roles played in each activity.

The software development process must be based upon the ESA PSS-05-0 life cycle model (SLC03). The key decisions in software process modelling are the:

- selection of the life cycle approach, i.e. the pattern of life cycle phases;
- modifications, if any, to the phase process models (see below);
- selection of methods and tools to support the activities in the phase process models.

2.4.2.1.1 Selection of life cycle approach

ESA PSS-05-0 defines three life cycle approaches, waterfall, incremental and evolutionary.

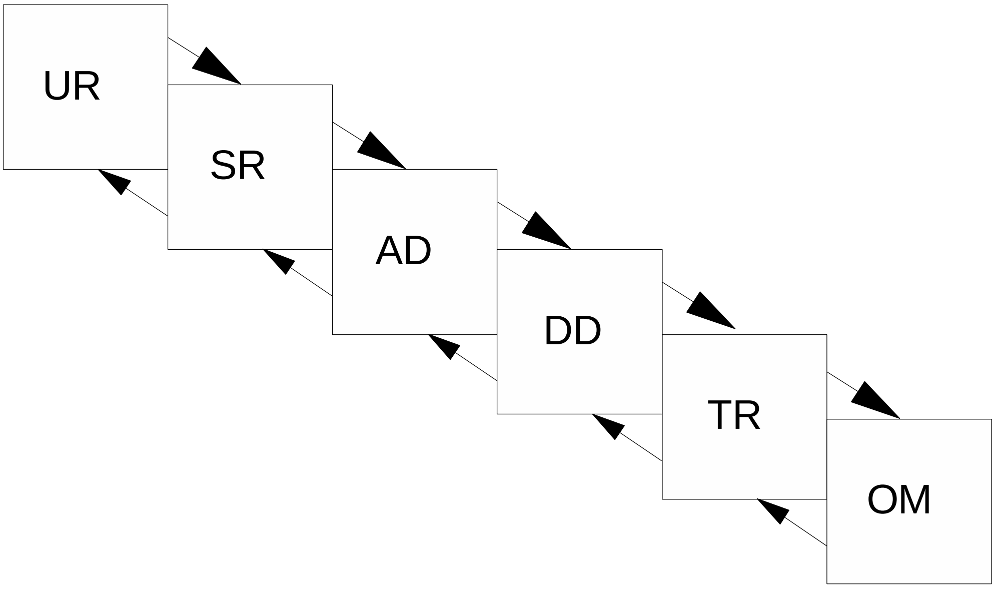


Figure 2.4.2.1.1A: Waterfall approach

The waterfall approach shown in Figure 2.4.2.1.1A executes each phase of the life cycle in sequence. Revisiting earlier phases is permitted to correct errors. The simple waterfall approach is suitable when:

- a set of high quality, stable user requirements exists;
- the length of project is short (i.e. two years or less);
- the users require the complete system all at once.

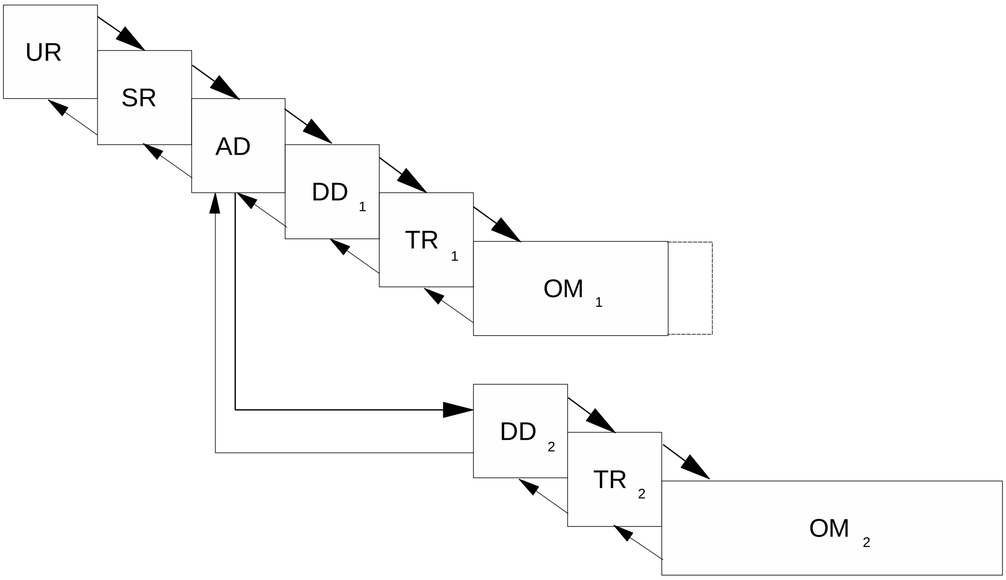


Figure 2.4.2.1.1B: Incremental approach

The incremental approach to development has multiple DD, TR and OM phases as shown in Figure 2.4.2.1.1B. This approach can be useful if:

- delivery of the software has to be according to the priorities set on the user requirements;
- it is necessary to improve the efficiency of integration of the software with other parts of the system (e.g. the telecommanding and telemetry subsystems in a spacecraft control system may be required early for system tests);
- early evidence that products will be acceptable is required.

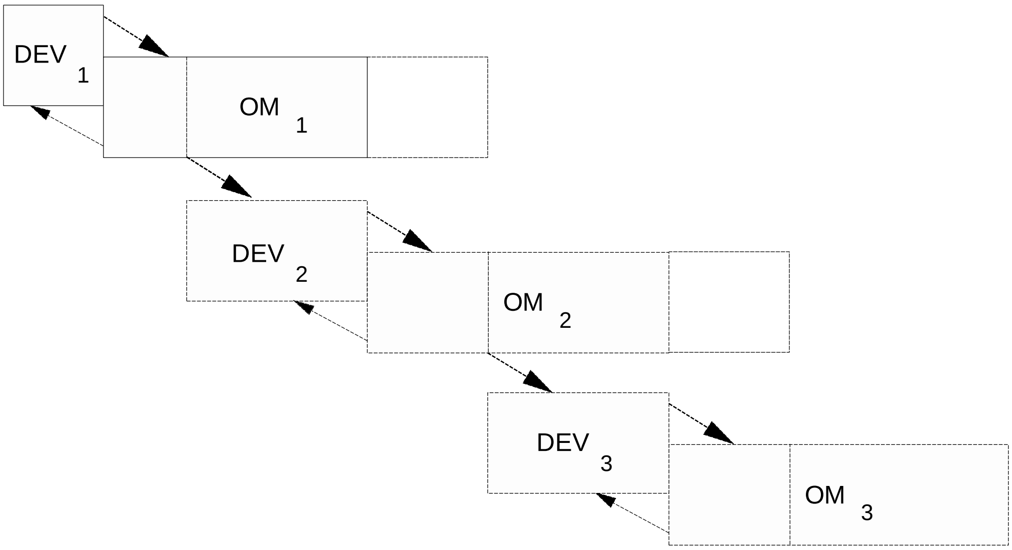


Figure 2.4.2.1.1C: Evolutionary approach

The evolutionary approach to development shown in Figure 2.4.2.1.1C consists of multiple waterfall life cycles with overlap between operations, maintenance and development. This approach may be used if, for example:

- some user experience is required to refine and complete the requirements (shown by the dashed line within the OM boxes);
- some parts of the implementation may depend on the availability of future technology;
- some new user requirements are anticipated but not yet known;
- some requirements may be significantly more difficult to meet than others, and it is not decided to allow them to delay a usable delivery.

2.4.2.1.2 Tailoring of the phase process models

The process models for the SR, AD and DD phases are shown in Figure 2.4.2.1.2A, B and C.

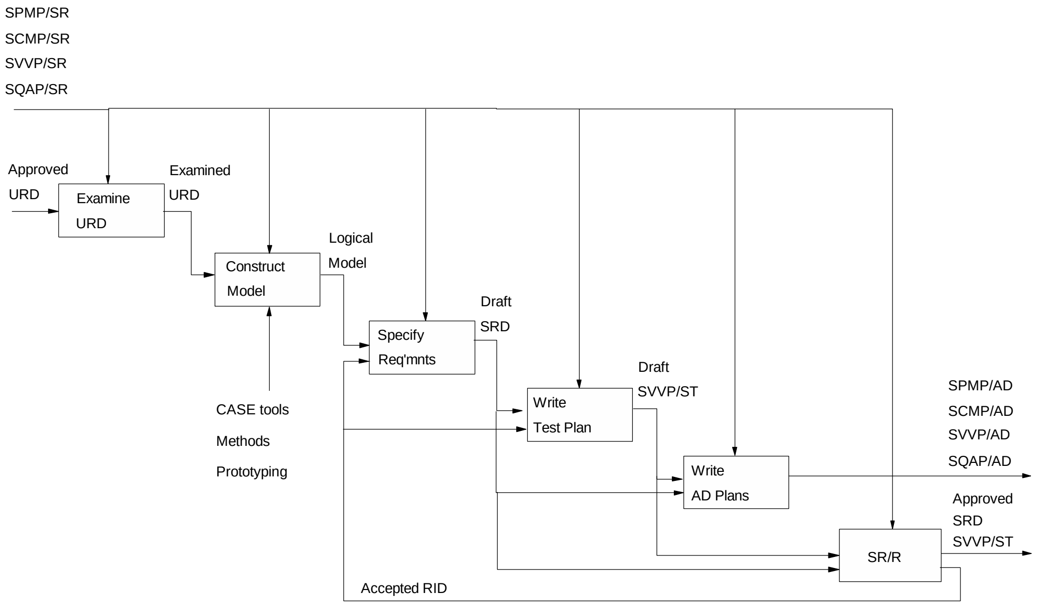


Figure 2.4.2.1.2A: SR phase process model

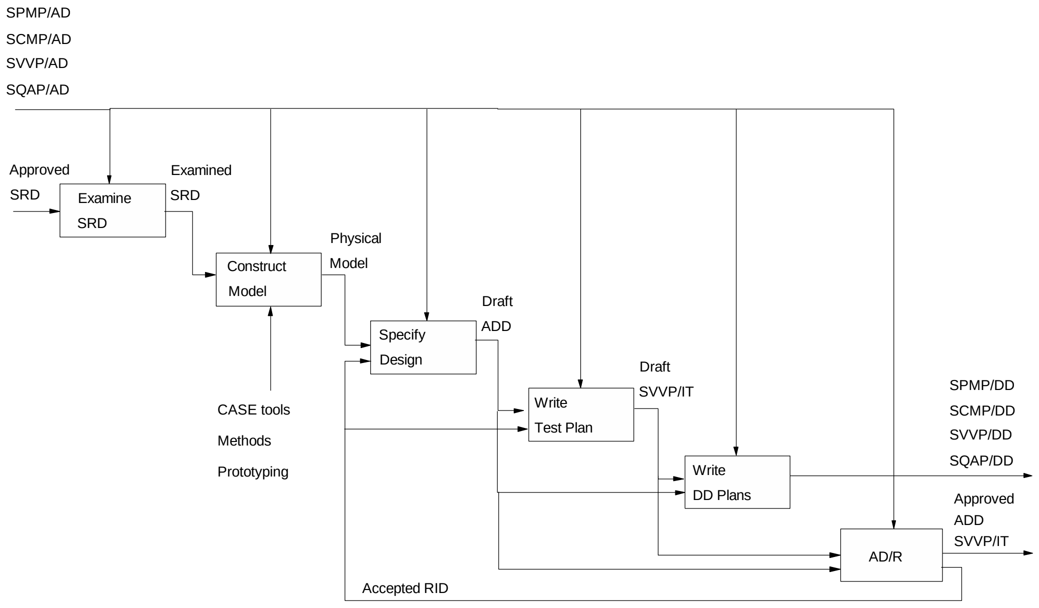


Figure 2.4.2.1.2B: AD phase process model

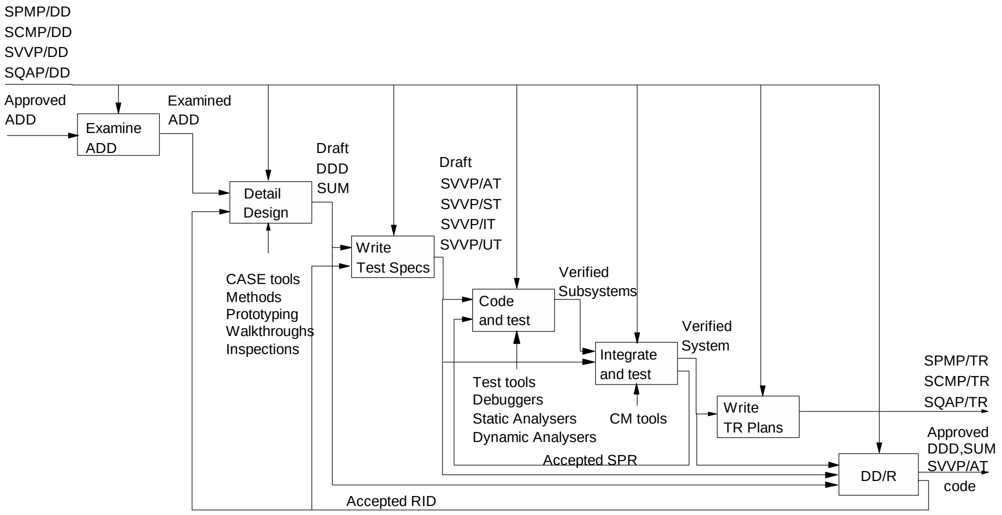


Figure 2.4.2.1.2C: DD phase process model

Project managers should tailor these process models to the needs of their projects. For example the specification of unit tests often has to wait until the code has been produced because there is insufficient information to define the test cases and procedures after the detailed design stage.

Further decomposition of the activities in the phase process models may be necessary. The granularity of the process model should be such that:

- managers have adequate visibility of activities at all times;
- every individual knows what to do at any time.

Detailed procedures, which should be part of the development organisation's quality system, should be referenced in the SPMP. They need not be repeated in the SPMP.

2.4.2.1.3 Selection of methods and tools

The project manager should, in consultation with the development team, select the methods and tools used for each activity in the phase process models. The ESA PSS-05 series of guides contains extensive information about methods and tools that may be used for software development [Ref 19].

2.4.2.2 Work package definition

The next step in planning is to decompose the work into 'work packages'. Activities in the process model are instantiated as work package tasks. The relationship between activities and tasks may be one-to-one (as in the case of constructing the logical model), or one-to-many (e.g. the code and unit test activity will be performed in many work packages).

Dividing the work into simple practical work packages requires a good understanding of the needs of the project and the logical relationships embedded in the process model. Some criteria for work package design are:

- coherence; tasks within a work package should have the same goal;
- coupling; work package dependencies should be minimised, so that team members can work independently;
- continuity; production work packages should be full-time to maximise efficiency;
- cost; bottom level work packages should require between one man-week and one man-month of effort.

The level of detail in the work breakdown should be driven by the level of accuracy needed for estimating resources and duration in the next stage of planning. This may be too detailed for other purposes, such as progress reporting to senior management. In such cases the higher level work packages are used to describe the work.

The work packages should be hierarchically organised so that related activities, such as software project management, are grouped together. The hierarchy of work packages is called the 'Work Breakdown Structure' (WBS). Table 2.4.2.2 shows an example of a WBS for a medium size project. In this example, each work package has a four digit identifier allowing up to four levels in the WBS.

1000	**Software Project Management**			
	1100	UR phase		
		1110	SPMP/SR production	
	1200	SR phase		
		1210	SPM reports	
		1220	SPMP/AD production	
	1300	AD phase		
		1310	SPM reports	
		1320	SPMP/DD production	
	1400	DD phase		
		1410	SPM reports	
		1420	SPMP/TR production	
		1430	SPMP/DD updates	
	1500	TR phase		
		1510	SPM reports	
2000	**Software Production**			
	2100	UR phase		
		2110	Requirements engineering	
	2200	SR phase		
		2210	Logical model	
		2220	Prototyping	
		2230	SRD draft	
		2240	SRD final	
	2300	AD phase		
		2310	Physical model	
		2320	Prototyping	
		2330	ADD draft	
		2340	ADD final	
	2400	DD phase		
		2410	Unit 1 production	
			2411	Unit 1 DDD
			2412	Unit 1 code
			2413	Unit 1 test
		2420	Unit 2 production	
			2421	Unit 2 DDD
			2422	Unit 2 code
			2423	Unit 2 test
		2430	Unit 3 production	
			2431	Unit 3 DDD
			2432	Unit 3 code
			2433	Unit 3 test
		2440	Unit 4 production	
			2441	Unit 4 DDD
			2442	Unit 4 code
			2443	Unit 4 test
		2450	Unit 5 production	
			2451	Unit 5 DDD
			2452	Unit 5 code
			2453	Unit 5 test

Table 2.4.2.2: Example Work Breakdown Structure

		2460	Integration and test	
			2461	Integration stage 1
			2462	Integration stage 2
			2463	Integration stage 3
			2464	Integration stage 4
		2470	SUM production	
	2500	TR phase		
		2510	Software installation	
		2520	STD production	
3000	**Software Configuration Management**			
	3100	UR phase		
		3110	SCMP/SR production	
	3200	SR phase		
		3210	Library maintenance	
		3220	Status accounting	
		3230	SCMP/AD production	
	3300	AD phase		
		3310	Library maintenance	
		3320	Status accounting	
		3330	SCMP/DD production	
	3400	DD phase		
		3410	Library maintenance	
		3420	Status accounting	
		3430	SCMP/TR production	
	3500	TR phase		
		3510	Library maintenance	
		3520	Status accounting	
4000	**Software Verification and Validation**			
	4100	UR phase		
		4110	SVVP/SR production	
		4120	SVVP/AT plan production	
		4130	UR review	
	4200	SR phase		
		4210	Walkthroughs	
		4220	Tracing	
		4230	SR review	
		4240	SVVP/ST plan production	
	4300	AD phase		
		4310	Walkthroughs	
		4320	Tracing	
		4330	SR review	
		4340	SVVP/IT plan production	
	4400	DD phase		
		4410	Unit review	
			4411	Unit 1 Review
			4412	Unit 2 Review
			4413	Unit 3 Review
			4414	Unit 4 Review
			4415	Unit 5 Review

Table 2.4.2.2: Example Work Breakdown Structure (continued)

		4420	SVVP/UT production
		4430	SVVP/IT production
		4440	SVVP/ST production
		4450	SVVP/AT production
		4460	System tests
		4470	DD review
	4500	TR phase	
		4510	Acceptance tests
		4520	Provisional Acceptance review
5000	**Software Quality Assurance**		
	5100	UR phase	
		5110	SQAP/SR production
	5200	SR phase	
		5210	SQA reports
		5220	SQAP/AD production
	5300	AD phase	
		5310	SQA reports
		5320	SQAP/DD production
	5400	DD phase	
		5410	SQA reports
		5420	SQAP/TR production
	5500	TR phase	
		5510	SQA reports

Table 2.4.2.2: Example Work Breakdown Structure (continued)

The number of work packages in a project normally increases with project size. Small projects (less than two man years) may have less than ten bottom-level work packages, and large projects (more than twenty man years) more than a hundred. Accordingly, small projects will use one or two levels, medium size projects two to four, and large projects four or five. Medium and large projects may use alphabetic as well as numerical identifiers.

New work packages should be defined for new tasks that are identified during a project. The tasks should not be attached to existing, unrelated, work packages. Some projects find it useful to create an 'unplanned work' package specifically for miscellaneous activities that arise in every project. However such 'unplanned work' packages should never be allocated more than a few per cent of the project resources. New work packages should be defined for tasks that involve significant expenditure.

Work packages are documented in the Work Packages, Schedule and Budget section of the SPMP (see Chapter 5). Work packages may reference standards, guidelines and manuals that define how to carry out the work. An example of a Work Package Description form is given in

Appendix D. A form should be completed for every work package. Repetition of information should be minimised.

2.4.3 Estimate resources and duration

The next steps in planning are:

- defining human resources;
- estimating effort;
- estimating non-labour costs;
- estimating duration.

2.4.3.1 Defining human resources

Project managers should analyse the work packages and define the 'human resource requirements'. These define the roles required in the project. Examples of software project roles are:

- project manager;
- team leader;
- programmer;
- test engineer;
- software librarian;
- software quality assurance engineer.

The project manager must also define the relationships between the roles to enable the effective coordination and control of the project. The following rules should be applied when defining organisational structures:

- ensure that each member of the team reports to one and only one person (the 'unity of command principle');
- ensure that each person has no more than seven people reporting directly to him or her (the 'rule of seven' principle).

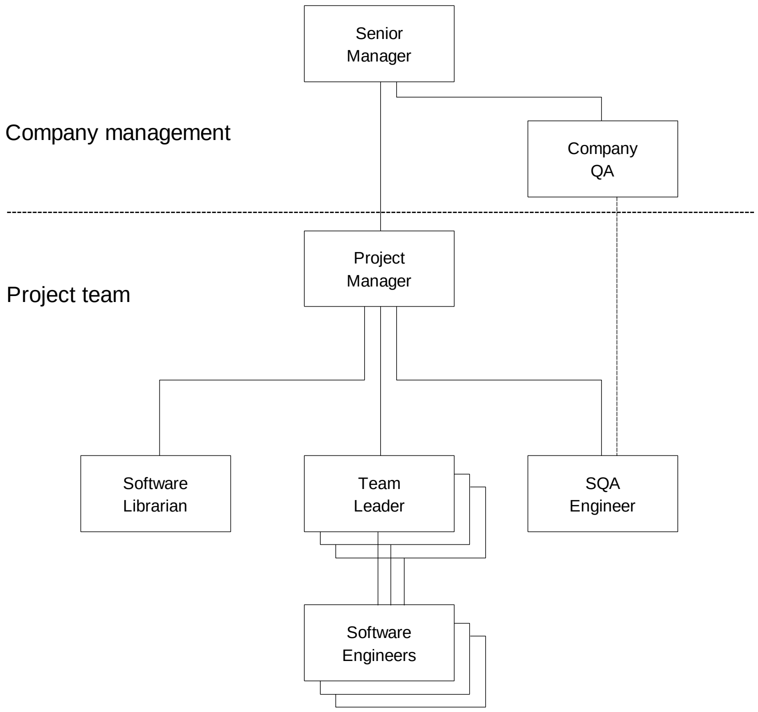


Figure 2.4.3.1.1: Example organigram for a medium-size project

The project team structure should be graphically displayed in an organigram, as shown in Figure 2.4.3.1.1. The 'dotted line' relationship between the project SQA engineer and the Company QA is a violation of the 'unity-of-command' principle that is justified by the need to provide senior management with an alternative, independent view on quality and safety issues.

The senior manager normally defines the activities of the project manager by means of terms of reference (see Section 2.2). The project manager defines the activities of the project team by means of work package descriptions.

2.4.3.2 Estimating effort

The project manager, with the assistance of his team, and, perhaps, external experts, should make a detailed analysis of the work packages and provide an estimate of the effort (e.g. number of man hours, days or months) that will be required.

Where possible these estimates should be compared with historical data. The comparison method can work quite well when good historical data exist. Historical data should be adjusted to take account of factors such as staff experience, project risks, new methods and technology and more efficient work practices.

Formula approaches such as Boehm's Constructive Cost Model (COCOMO) and Function Point Analysis (FPA) should not be used for making estimates at this stage of planning. These methods may be used for verifying the total cost estimate when the plan has been made. These methods are further discussed in Chapter 3.

2.4.3.3 Estimating non-labour costs

Non-labour costs are normally estimated from supplier data. They may include:

- commercial products that form part of the end product;
- commercial products that are used to make the end-product but do not form part of it, e.g. tools;
- materials (i.e. consumables not included in overheads);
- internal facilities (e.g. computer and test facilities);
- external services (e.g. reproduction);
- travel and subsistence;
- packing and shipping;
- insurance.

2.4.3.4 Estimating duration

The duration of the work package may be calculated from effort estimates and historical productivity data or other practical considerations (e.g. the duration of an activity such as a review meeting might be fixed at one day). The duration of each work package is needed for building the activity network and calculating the total project duration in the next stage of planning.

Productivity estimates should describe the average productivity and not the maximum productivity. Studies show that miscellaneous functions can absorb up to about 50% of staff time [Ref 10], reducing the average productivity to much less than the peak. Furthermore, industrial productivity figures (such as the often quoted productivity range of 10 to 20 lines of code per day) are normally averaged over the whole development, and not just the coding stage.

2.4.4 Define activity network

An activity network represents the work packages in the project as a set of nodes with arrows linking them. A sequence of arrows defines a path, or part of a path, through the project. Circular paths are not allowed in an activity network. The primary objective of building an activity network is to design a feasible pattern of activities that takes account of all dependencies. Two important by-products of constructing the activity network are the:

- critical path;
- work package float.

The critical path is the longest path through the network in terms of the total duration of activities. The time to complete the critical path is the time to complete the project.

The float time of a work package is equal to the difference between the earliest and latest work package start (or end) times, and is the amount of time each activity can be moved without affecting the total time to complete the project. Activities on the critical path therefore have zero float time.

In general, activity networks should only include work packages that depend upon other work packages for input, or produce output for other work packages to use.

Activities that are just connected to the start and end nodes should be excluded from the network. This simplifies it without affecting the critical path. Further, the activity network of complex projects should be broken down into subnetworks (e.g. one per phase). Such a modular approach makes the project easier to manage, understand and evaluate.

2.4.5 Define schedule and total cost

The last stage of planning defines the schedule and resource requirements, and finally the total cost.

The activity network constrains the schedule but does not define it. The project manager has to decide the actual schedule by setting the start and end times of work packages so as to:

- comply with time and resource constraints;
- minimise the total cost;
- minimise the fragmentation of resource allocations;

- allow for any risks that might delay the project.

The start and end dates of work packages affected by time constraints (e.g. delivery dates) and resource constraints (e.g. staff and equipment availability) should be set first, as they reduce the number of possibilities to consider. If the total time to complete the project violates the time constraints, project managers should return to the define activities stage, redefine work packages, re-estimate resources and duration, and then modify the activity network.

The minimum total cost of the project is the sum of all the work packages. However labour costs should be calculated from the total amount of time spent by each person on the project. This ensures that the time spent waiting for work packages to start and end is included in the cost estimate. Simply summing the individual costs of each bottom-level work package excludes this overhead. Project managers adjust the schedule to bring the actual cost of the project as close as possible to the minimum.

Activities with high risk factors should be scheduled to start at their earliest possible start times so that the activity float can be used as contingency.

Minimising the fragmentation of a resource allocation is often called 'resource smoothing'. Project managers should adjust the start time of activities using the same resource so that it is used continuously, rather than at discrete intervals. This is because:

- interruptions mean that people have to refamiliarise with the current situation, or relearn procedures that have been forgotten;
- equipment may be charged to the project even when it is not in use.

2.5 LEADING THE PROJECT

Leadership provides direction and guidance to the project team, and is an essential management function. There are numerous studies on leadership in the management literature, and the interested reader should consult them directly.

2.6 TECHNICAL MANAGEMENT OF PROJECTS

Besides the managerial issues, a software project manager must also understand the project technically. He or she is responsible for the major decisions concerning:

- the methods and tools;
- the design and coding standards;
- the logical model;
- the software requirements;
- the physical model;
- the architectural design;
- the detailed design;
- configuration management;
- verification and validation;
- quality assurance.

In medium and large projects the project manager often delegates much of the routine technical management responsibilities to team leaders.

2.7 MANAGEMENT OF PROJECT RISKS

All projects have risks. The point about risk management is not to run away from risks, but to reduce their ability to threaten the success of a project. Project managers should manage risk by:

- continuously looking out for likely threats to the success of the project;
- adjusting the plan to minimise the probability of the threats being realised;
- defining contingency plans for when the threats are realised;
- implementing the contingency plans if necessary.

Risk management never starts from the optimistic premise that 'all will go well'. Instead project managers should always ask themselves 'what is most likely to go wrong?' This is realism, not pessimism.

Project plans should identify the risks to a project and show how the plan takes account of them.

The following sections discuss the common types of risk to software projects, with possible actions that can be taken to counteract them. There are four groups:

- experience factors;
- planning factors;
- technology factors;
- external factors.

2.7.1 Experience factors

Experience factors that can be a source of risk are:

- experience and qualifications of the project manager;
- experience and qualifications of staff;
- maturity of suppliers.

2.7.1.1 Experience and qualifications of the project manager

The project manager is the member of staff whose performance has a critical effect on the outcome of a project. Inexperienced project managers are a significant risk, and organisations making appointments should match the difficulty of the project to the experience of the project manager. Part-time project managers are also a risk, because this reduces the capability of a project to respond to problems quickly.

Project management is a discipline like any other. Untrained project managers are a risk because they may be unaware of what is involved in management. Staff moving into management should receive training for their new role.

Project management should be the responsibility of a single person and not be divided. This ensures unity of command and direction.

2.7.1.2 Experience and qualifications of staff

Staff will be a risk to a project if they have insufficient experience, skills and qualifications for the tasks that they are assigned.

Project managers should avoid such risks by:

- assessing staff before they join the project;

- allocating staff tasks that match their experience, skills and qualifications;
- retaining staff within the project who have the appropriate skills, experience and qualifications to cope with future tasks.

2.7.1.3 Maturity of suppliers

The experience and record of suppliers are key factors in assessing the risks to a project. Indicators of maturity are the:

- successful development of similar systems;
- use of software engineering standards;
- the possession of an ISO 9000 certificate;
- existence of a software process improvement programme.

Experience of developing similar systems is an essential qualification for a project team. Lack of experience results in poor estimates, avoidable errors, and higher costs (because extra work is required to correct the errors).

Standards not only include system development standards such as ESA PSS-05-0, but also coding standards and administrative standards. Standards may exist within an organisation but ignorance of how best to apply them may prevent their effective use. Project managers should ensure that standards are understood, accepted and applied. Project managers may require additional support from software quality assurance staff to achieve this.

A software process improvement programme is a good sign of maturity. Such a programme should be led by a software process group that is staffed with experienced software engineers. The group should collect data about the current process, analyse it, and decide about improvements to the process.

2.7.2 Planning factors

Planning factors that can be a source of risk are:

- accuracy of estimates;
- short timescales;
- long timescales;
- single-point failures;

- location of staff;
- definition of responsibilities;
- staffing profile evolution.

2.7.2.1 Accuracy of estimates

Accurate estimates of resource requirements are needed to complete a project successfully. Underestimates are especially dangerous if there is no scope for awarding additional resources later. Overestimation can result in waste and can prevent resources from being deployed elsewhere.

Some of the activities which are often poorly estimated for are:

- testing;
- integration, especially of external systems;
- transfer phase;
- reviews, including rework.

Some contingency should always be added to the estimates to cater for errors. The amount of contingency should also be related to the other risks to the project. Estimating the effort required to do something new is particularly difficult.

2.7.2.2 Short timescales

Short timescales increase the amount of parallel working required, resulting in a larger team. Progressive reduction in the timescale increases this proportion to a limit where the project becomes unmanageable [Ref 9]. Project managers should not accept unrealistic timescales.

Project managers should avoid artificially contracting timescales by attempting to deliver software before it is required. They should use the time available to optimise the allocation of resources such as staff.

When faced with accumulating delays and rapidly approaching deadlines, project manager's should remember Brooks' Law: 'adding manpower to a late project makes it later' [Ref 10].

2.7.2.3 Long timescales

Projects with long timescales run the risk of:

- changes in requirements;
- staff turnover;
- being overtaken by technological change.

Long timescales can result from starting too early. Project managers should determine the optimal time to start the project by careful planning.

Long projects should consider using an incremental delivery or evolutionary development life cycle approach. Both approaches aim to deliver some useful functionality within a timescale in which the requirements should be stable. If this cannot be done then the viability of the project should be questioned, as the product may be obsolete or unwanted by the time it appears.

Ambitious objectives, when combined with constraints imposed by annual budgets, often cause projects to have long timescales. As management costs are incurred at a fixed rate, management consumes a larger proportion of the project cost as the timescale lengthens. Furthermore, change, such as technical and economic change, can make the objectives obsolete, resulting in the abandonment of the project before anything is achieved!

2.7.2.4 Single-point failures

A single-point failure occurs in a project when a resource vital to an activity fails and there is no backup. Project managers should look for single-point failures by examining each activity and considering:

- the reliability of the resources;
- whether backup is available.

Project managers should devise contingency plans to reallocate resources when failures occur.

2.7.2.5 Location of staff

Dispersing staff to different locations can result in poor communication. Project managers should co-locate staff in contiguous accommodation wherever possible. This improves communication and allows for more flexible staff allocation.

2.7.2.6 Definition of responsibilities

Poor definition of responsibilities is a major threat to the success of a project. Vital jobs may not be done simply because no-one was assigned to do them. Tasks may be repeated unnecessarily because of ignorance about responsibilities.

Projects should be well organised by defining project roles and allocating tasks to the roles. Responsibilities should be clearly defined in work package descriptions.

2.7.2.7 Staffing profile evolution

Rapid increases in the number of staff can be a problem because new staff need time to familiarise themselves with a project. After the start of the project much of the project knowledge comes from the existing staff. These 'teaching resources' limit the ability of a project to grow quickly.

Rapid decreases in the number of staff can be a problem because the loss of expertise before the project is finished can drastically slow the rate at which problems are solved. When experienced staff leave a project, time should be set aside for them to transfer their knowledge to other staff.

The number of staff on a software project should grow smoothly from a small team in the software requirements definition phase to a peak during coding and unit testing, and fall again to a small team for the transfer phase. Project managers avoid sudden changes in the manpower profile and ensure that the project can absorb the inflow of staff without disruption.

2.7.3 Technological factors

Technological factors that can be a source of risk are:

- technical novelty;
- maturity and suitability of methods;
- maturity and efficiency of tools;
- quality of commercial software.

2.7.3.1 Technical novelty

Technical novelty is an obvious risk. Project managers should assess the technical novelty of every part of the design. Any radically new component can cause problems because the:

- feasibility has not been demonstrated;
- amount of effort required to build it will be difficult to estimate accurately.

Project managers should estimate the costs and benefits of technically novel components. The estimates should include the costs of prototyping the component.

Prototyping should be used to reduce the risk of technical novelty. The cost of prototyping should be significantly less than the cost of developing the rest of the system, so that feasibility is demonstrated before major expenditure is incurred. More accurate cost estimates are an added benefit of prototyping.

2.7.3.2 Maturity and suitability of methods

New methods are often immature, as practical experience with a method is necessary to refine it. Furthermore, new methods normally lack tool support.

Choosing an unsuitable method results in unnecessary work and, possibly, unsuitable software. Analysis and design methods should match the application being built. Verification and validation methods should match the criticality of the software.

Project managers should evaluate the strengths (e.g. high suitability) and the weaknesses (e.g. lack of maturity) of a method before making a choice. Project managers should always check whether the method has been used for similar applications.

2.7.3.3 Maturity and efficiency of tools

Tools can prove to be a hindrance instead of a benefit if:

- staff have not been trained in their use;
- they are unsuitable for the methods selected for the project;
- they are changed during the project;

- they have more overheads than manual techniques.

The aim of using tools is to improve productivity and quality. Project managers should carefully assess the costs and benefits of tools before deciding to use them on a project.

2.7.3.4 Quality of commercial software

The use of proven commercial software with well-known characteristics is normally a low-risk decision. However, including new commercial software, or using commercial software for the first time in a new application area, can be a risk because:

- it may not work as advertised, or as expected;
- it may not have been designed to meet the quality, reliability, maintainability and safety requirements of the project;

New commercial software should be comprehensively evaluated as early as possible in the project so that surprises are avoided later. Besides the technical requirements, other aspects to evaluate when considering the acquisition of commercial software are:

- supplier size, capability and experience;
- availability of support;
- future development plans.

2.7.4 External factors

External factors that can be a source of risk are the:

- quality and stability of user requirements;
- definition and stability of external interfaces;
- quality and availability of external systems.

2.7.4.1 Quality and stability of user requirements

A project is unlikely to succeed without a high quality User Requirements Document. A URD is a well-defined baseline against which to measure success. Project managers should, through the User Requirements Review process, ensure that a coherent set of requirements is available. This may require resources very early in the project to help users define their requirements (e.g. prototyping).

2.7.4.2 Definition and stability of external interfaces

External interfaces can be a risk when their definitions are unstable, poorly defined or non-standard.

Interfaces with external systems should be defined in Interface Control Documents (ICDs). These are needed as early as possible because they constrain the design. Once agreed, changes should be minimised, as a change in an interface implies rework by multiple teams.

Standard interfaces have the advantage of being well-defined and stable. For these reasons a standard interface is always to be preferred, even if the system design has to be modified.

2.7.4.3 Quality and availability of external systems

External systems can be a problem if their quality is poor or if they are not available when required either owing to late delivery or to unreliability.

Project managers should initiate discussions with suppliers of external systems when the project starts. This ensures that suppliers are made aware of project requirements as early as possible. Project managers should allocate adequate resources to the integration of external systems.

The effects of late delivery of the external system can be mitigated by:

- adding the capability of temporarily 'bypassing' the external system;
- development of software to simulate the external system.

The cost of the simulator must be traded-off against the costs incurred by late delivery.

2.8 MEASURING PROJECT PROCESSES AND PRODUCTS

Project managers should as far as possible adopt a quantitative approach to measuring software processes and products. This is essential for effective project control, planning future projects and improving the software development process in the organisation.

The Application of Metrics in Industry (AMI) guide [Ref 18] describes the following measurement steps:

1. assess the project environment and define the primary goals (e.g. financial targets, quality, reliability etc);
2. analyse the primary goals into subgoals that can be quantitatively measured and define a metric (e.g. man-months of effort) for each subgoal;
3. collect the raw metric data and measure the performance relative to the project goals;
4. improve the performance of the project by updating plans to correct deviations from the project goals;
5. improve the performance of the organisation by passing the metric data and measurements to the software process improvement group, who are responsible for the standards and procedures the project uses.

Steps one to four are discussed in the following sections.

2.8.1 Assess environment and define primary goal

The purpose of assessing the environment is to understand the context of the project. An audit aimed at measuring conformance to ESA PSS-05-0 is one means of assessment. Other methods for assessing software development organisations and projects include the Software Engineering Institute’s Capability Maturity Model [Ref 4] and the European Software Institute’s ‘BOOTSTRAP’ model, which uses the ESA Software Engineering Standards as a definition of basic practices. Whatever the approach taken, the assessment step should define what is needed and what is practical.

The primary software project goal is invariably to deliver the product on time, within budget, with the required quality.

2.8.2 Analyse goals and define metrics

Common subgoals related to the primary goal are:

- do not exceed the planned effort on each activity;
- do not exceed the planned time on each activity;
- ensure that the product is complete;
- ensure that the product is reliable.

A metric is a 'quantitative measure of the degree to which a system, component or process possesses a given attribute' [Ref 2]. Project managers should define one or more metrics related to the subgoals defined for the project. Metric definitions should as far as possible follow organisational standards or industry standards, so that comparisons with other projects are possible [Ref 18].

2.8.2.1 Process metrics

Possible metrics for measuring project development effort are the:

- amount of resources used;
- amount of resources left.

Possible metrics for measuring project development time are the:

- actual duration of activities in days, weeks or months;
- slippage of activities (actual start - planned start).

Possible metrics for measuring project progress are:

- number of work packages completed;
- number of software problems solved.

2.8.2.2 Product metrics

The amount of product produced can be measured in terms of:

- number of lines of source code produced, excluding comments;
- number of modules coded and unit tested;
- number of function points implemented [Ref 11, 15];
- number of pages of documentation written.

Actual values should be compared with target values to measure product completeness.

Possible metrics for measuring product reliability are:

- test coverage;
- cyclomatic complexity of source modules;
- integration complexity of programs;
- number of critical Software Problem Reports;
- number of non-critical Software Problem Reports;
- number of changes to products after first issue or release.

2.8.3 Collect metric data and measure performance

Project managers should ensure that metric data collection is a natural activity within the project. For example the actual effort expended should always be assessed before a Work Package is signed off as completed. Counts of Software Problem Reports should be accumulated in the Configuration Status Accounting process. Cyclomatic complexity can be a routine measurement made before code inspection or unit test design.

2.8.4 Improving performance

Metric data should be made available for project management reporting, planning future projects, and software process improvement studies [Ref 4].

The project manager should use the metric data to improve processes and products. For example:

- comparisons of predicted and actual effort early in a project can quickly identify deficiencies in the initial estimates, and prompt major replanning of the project to improve progress;
- analysis of the trends in software problem occurrence during integration and system testing can show whether further improvement in quality and reliability is necessary before the software is ready for transfer.

2.9 PROJECT REPORTING

Accurate and timely reporting is essential for the control of a project. Project managers report to initiators and senior managers by means of progress reports. Initiators and senior managers should analyse progress reports and arrange regular meetings with the project manager to review the project. The progress report is an essential input to these management reviews.

Project managers should have frequent discussions with team members so that they are always up-to-date with project status. Team members should formally report progress to project managers by means of work package completion reports and timesheets.

2.9.1 Progress reports

Project managers should submit routine (e.g. monthly) progress reports that describe:

- technical status;
- resource status;
- schedule status;
- problems;
- financial status.

Progress reports are discussed in detail in Chapter 6.

2.9.2 Work package completion reports

As each work package is completed, the responsible team member (called the 'work package manager') should notify the project manager by a 'work package completion report'. Project managers accept work package deliverables by issuing 'work package completion certificates'. One way to document this process is to add completion and acceptance fields to the work package description form.

2.9.3 Timesheets

Organisations should have a timesheet system that project managers can use for tracking the time spent by staff on a project. A timesheet describes what each employee has done on a daily or hourly basis during the reporting period. The project manager needs this data to compile progress reports.

A good timesheet system should allow staff to record the work package associated with the expenditure of effort, rather than just the project. This level of granularity in the record keeping eases the collection of data about project development effort, and provides a more precise database for estimating future projects.

This page is intentionally left blank

CHAPTER 3
SOFTWARE PROJECT MANAGEMENT METHODS

3.1 INTRODUCTION

Various methods and techniques are available for software project management. This chapter discusses some methods for the activities described in Chapter 2. The reader should consult the references for detailed descriptions. Methods are discussed for:

- project planning;
- project risk management;
- project reporting.

3.2 PROJECT PLANNING METHODS

Methods are available for supporting the following project planning activities:

- process modelling;
- estimating resources and duration;
- defining activity networks;
- defining schedules.

The following sections discuss these methods.

3.2.1 Process modelling methods

The objective of a process modelling method is to construct a model of the software process roles, responsibilities, activities, inputs and outputs. This is often referred to as the 'workflow'.

Process models have been traditionally defined using text or simple diagrams. The Software Lifecycle Model defined in Part 1, Figure 1.2 of ESA PSS-05-0 is an example. Process models have also been defined using notations derived from systems analysis. SADT diagrams can be used for illustrating the flow of products and plans between activities [Ref 6].

Specialised process modelling methods based upon programming languages, predicate logic and Petri Nets are recent developments in software engineering [Ref 7]. These methods allow dynamic modelling of the software process. Dynamic models are required for workflow support.

3.2.2 Estimating methods

Chapter 2 recommends that labour costs of each work package be estimated mainly by expert opinion. The total labour cost is calculated from the total time spent on the project by each team member. The non-labour costs are then added to get the cost to completion of the project. The labour cost estimates should be verified by using another method. This section discusses the alternative methods of:

- historical comparison;
- COCOMO;
- function point analysis;
- activity distribution analysis;
- Delphi methods;
- integration and system test effort estimation;
- documentation effort estimation.

Some of these methods use formulae. Anyone using formula approaches to software cost estimation should take note of the following statement from Albrecht, the originator of Function Point Analysis [Ref 11]:

"It is important to distinguish between two types of work-effort estimates, a primary or 'task analysis' estimate and a 'formula' estimate. The primary work-effort estimate should always be based on an analysis of the tasks , thus providing the project team with an estimate and a work plan... It is recommended that formula estimates be used only to validate and provide perspective on primary estimates".

Boehm gives similar advice, which is all too often ignored by users of his COCOMO method [Ref 9].

3.2.2.1 Historical comparison

The historical comparison method of cost estimation is simply to compare the current project with previous projects. This method can work quite well when:

- the costs of the previous projects have been accurately recorded;
- the previous project has similarities to those of the current project.

If the projects are all identical, then a simple average of the costs can be used. More commonly, minor differences between the projects will lead to a range of actual costs. Estimates can be made by identifying the most similar project, or work package.

Historical comparisons can be very useful for an organisation which develops a series of similar systems, especially when accounts are kept of the effort expended upon each work package. There is no substitute for experience. However, possible disadvantages of the approach are that:

- new methods and tools cannot be accounted for in the estimates;
- inefficient work practices are institutionalised, resulting in waste.

To avoid these problems, it is very important that special features of projects that have influenced the cost are documented, enabling software project managers to adjust their estimates accordingly.

Part of the purpose of the Project History Document (PHD) is to pass on the cost data and the background information needed to understand it.

3.2.2.2 COCOMO

The Constructive Cost Model (COCOMO) is a formula-based method for estimating the total cost of developing software [Ref 9]. The fundamental input to COCOMO is the estimated number of lines of source code. This is its major weakness because:

- the number of lines of code is only accurately predictable at the end of the architectural design phase of a project, and this is too late;
- what constitutes a 'line of code' can vary amongst programming languages and conventions;
- the concept of a line of code does not apply to some modern programming techniques, e.g. visual programming.

COCOMO offers basic, intermediate and detailed methods. The basic method uses simple formulae to derive the total number of man months of effort, and the total elapsed time of the project, from the estimated number of lines of code.

The intermediate method refines the basic effort estimate by multiplying it with an 'effort adjustment factor' derived from fifteen 'effort multipliers'. Wildly inaccurate estimates can result from a poor choice of effort multiplier values. COCOMO supplies objective criteria to help the estimator make a sensible choice. The detailed COCOMO method uses a separate set of effort multipliers for every phase.

When estimating by means of COCOMO, the intermediate method should be used because the detailed method does not appear to perform more accurately than the intermediate method. The basic method provides a level of accuracy that is only adequate for preliminary estimates.

3.2.2.3 Function Point Analysis

Function Point Analysis (FPA) estimates the cost of a project from the estimates of the delivered functionality [Ref 11, 13]. FPA can therefore be applied at the end of the software requirements definition phase to make cost estimates. The method combines quite well with methods such as structured analysis, and has been used for estimating the total effort required to develop an information system.

The FPA approach to cost estimation is based upon counting the numbers of system inputs, outputs and data stores. These counts are weighted, combined and then multiplied by cost drivers, resulting in a 'Function Point' count. The original version of FPA converts the number of function points to lines of code using a language dependent multiplier. The number of lines of code is then fed into COCOMO (see Section 3.2.2.2) to calculate the effort and schedule.

Mark Two Function Point Analysis simplifies the original approach, is more easily applicable to modern systems, and is better calibrated [Ref 15]. Furthermore it does not depend upon the use of language-dependent multipliers to produce an effort estimate. When estimating by means of FPA, the Mark Two version should be used.

3.2.2.4 Activity distribution analysis

Activity distribution analysis uses measurements of the proportion of effort expended on activities in previous projects to:

- derive the effort required for each activity from the total effort;
- verify that the proportion of effort expended upon each activity is realistic;
- verify that the effort required for future activities is in proportion to the effort expended upon completed activities.

Project phases are the top-level activities in a project. The activity distribution technique is called the 'phase per cent method' when it is used to estimate the amount of effort required for project phases.

The activity distribution method requires data about completed projects to be analysed and reduced to derive the relative effort values. As with all methods based upon historical data, the estimates need to be adjusted to take account of factors such as staff experience, project risks, new methods and technology and more efficient work practices. For example the use of modern analysis and design methods and CASE tools requires a higher proportion of effort to be expended in the SR and AD phases.

3.2.2.5 Delphi method

The 'Delphi' method is a useful approach for arriving at software cost estimates [Ref 9]. The assumption of the Delphi method is that experts will independently converge on a good estimate.

The method requires several cost estimation experts and a coordinator to direct the process. The basic steps are:

1. the coordinator presents each expert with a specification and a form upon which to record estimates;
2. the experts fill out forms anonymously, explaining their estimates (they may put questions to the coordinator, but should not discuss the problem with each other);
3. if consensus has not been achieved, the coordinator prepares a summary of all the estimates and distributes them to the experts and the process repeats from step 1.

The summary should just give the results, and not the reasoning behind the results.

A common variation is to hold a review meeting after the first pass to discuss the independent estimates and arrive at a common estimate.

3.2.2.6 Integration and System Test effort estimation

Integration and system test effort depends substantially upon the:

- number and criticality of defects in the software after unit testing;
- number and criticality of defects acceptable upon delivery;

- number and duration of integration and system tests;
- average time to repair a defect.

The effort required for integration and system testing can therefore be estimated from a model of the integration and testing process, assumptions about the number of defects present after unit tests, and an estimate of the mean repair effort per defect [Ref 14].

A typical integration and test process model contains the following cycle:

1. the integration and test team adds components, discovers defects, and reports them to the software modification team;
2. the software modification team repairs the defects;
3. the integration and test team retests the software.

Reference 14 states that twenty to fifty defects per 1000 lines of code typically occur during the life cycle of a software product, and of these, five to fifteen defects per 1000 lines of code remain when integration starts, and one to four defects per 1000 lines of code still exist after system tests.

Repair effort values will vary very widely, but a mean value of half a man-day for a code defect is typical.

3.2.2.7 Documentation effort estimation

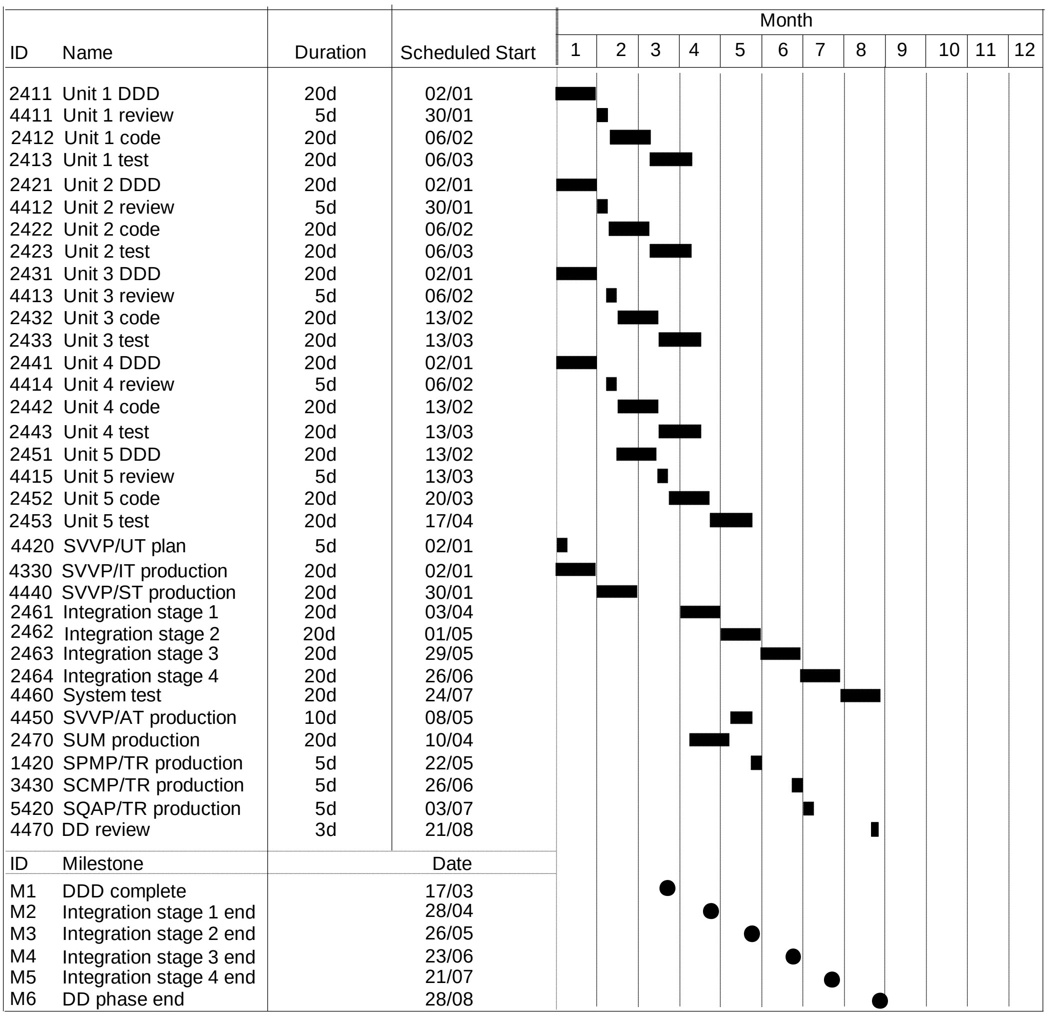
Software consists of documentation and code. Documentation is a critical output of software development, not an overhead. When estimating the volume of software that a project should produce, project managers should define not only the amount of code that will be produced, but also the amount of documentation.

Boehm observes that the documentation rates vary between two and four man hours per page [Ref 9]. Boehm also observes that the ratio between code volume and documentation volume ranges from about 10 to about 150 pages of documentation per 1000 lines of code, with a median value of about 50. These figures apply to all the life cycle documents.

The average productivity figure of 20 lines of code per day includes the effort required for documentation. This productivity figure, when combined with the documentation rates given above, implies that software engineers spend half their time on documentation (i.e. the median of 50 pages of documentation per 1000 lines of code implies one page of documentation per 20 lines of code; one page of documentation requires four hours, i.e. half a day, of effort).

3.2.3 Activity network methods

The Program Evaluation and Review Technique (PERT) is the usual method for constructing an activity network [Ref 9]. Most planning tools construct a PERT chart automatically as the planner defines activities and their dependencies.



ID	Name	Duration	Scheduled Start
2411	Unit 1 DDD	20d	02/01
4411	Unit 1 review	5d	30/01
2412	Unit 1 code	20d	06/02
2413	Unit 1 test	20d	06/03
2421	Unit 2 DDD	20d	02/01
4412	Unit 2 review	5d	30/01
2422	Unit 2 code	20d	06/02
2423	Unit 2 test	20d	06/03
2431	Unit 3 DDD	20d	02/01
4413	Unit 3 review	5d	06/02
2432	Unit 3 code	20d	13/02
2433	Unit 3 test	20d	13/03
2441	Unit 4 DDD	20d	02/01
4414	Unit 4 review	5d	06/02
2442	Unit 4 code	20d	13/02
2443	Unit 4 test	20d	13/03
2451	Unit 5 DDD	20d	13/02
4415	Unit 5 review	5d	13/03
2452	Unit 5 code	20d	20/03
2453	Unit 5 test	20d	17/04
4420	SVVP/UT plan	5d	02/01
4330	SVVP/IT production	20d	02/01
4440	SVVP/ST production	20d	30/01
2461	Integration stage 1	20d	03/04
2462	Integration stage 2	20d	01/05
2463	Integration stage 3	20d	29/05
2464	Integration stage 4	20d	26/06
4460	System test	20d	24/07
4450	SVVP/AT production	10d	08/05
2470	SUM production	20d	10/04
1420	SPMP/TR production	5d	22/05
3430	SCMP/TR production	5d	26/06
5420	SQAP/TR production	5d	03/07
4470	DD review	3d	21/08

ID	Milestone	Date
M1	DDD complete	17/03
M2	Integration stage 1 end	28/04
M3	Integration stage 2 end	26/05
M4	Integration stage 3 end	23/06
M5	Integration stage 4 end	21/07
M6	DD phase end	28/08

Figure 3.2.4: Example Gantt chart for the DD phase of a project

3.2.4 Scheduling presentation methods

The project schedule defines the start times and duration of each work package and the dates of each milestone. The 'Gantt chart' is the usual method for presenting it. Work packages are marked on a vertical axis and time along the corresponding horizontal axis. Work package activities are shown as horizontal bars (giving rise to synonym of 'bar chart'). Time may be expressed absolutely or relative to the start of the phase.

Figure 3.2.4 shows the schedule for DD phase of the example project described in Chapter 2. Planning tools normally construct the Gantt chart automatically as tasks are defined.

3.3 PROJECT RISK MANAGEMENT METHODS

Two simple and effective methods for risk management are:

- risk table;
- risk matrix.

3.3.1 Risk table

The steps of the risk table method are :

1. list the risks in the project:
2. define the probability of each risk;
3. define the risk reduction action or alternative approach;
4. define the decision date;
5. define the possible impact.

An example is provided in Table 3.3.1.

Risk	Description	Prob.	Action	Decision Date	Impact
1	New user requirements	High	Change to evolutionary development	1/Jun/1997	High
2	Installation of air conditioning	Medium	Relocate staff while work is done	1/April/1998	Medium

Table 3.3.1: Example of a Risk Table

3.3.2 Risk matrix

The steps of the risk matrix method are:

1. list the risks in the project:

2. define the probability of each risk;
3. define the possible impact;
4. plot the risks according to their probability and impact.

An example is provided in Figure 3.3.2, using the data in the risk table in Figure 3.3.1. Risks with high probability and high impact cluster in the top right hand corner of the matrix. These are the risks that should receive the most attention.

	high			**1**
probability	medium		**2**	
	low			
		low	medium	high
			impact	

Figure 3.3.2: Example of a Risk Matrix

3.4 PROJECT REPORTING METHODS

Some important methods for progress reporting are:

- progress tables
- progress charts;
- milestone trend charts.

3.4.1 Progress tables and charts

Progress tables and charts are used to report how much has been spent on each work package and how much remains to be spent. For each work package in the WBS an initial estimate of the resources is made. The initial estimates should be in the SPMP. At the end of every reporting period, the following parameters are collected:

1. previous estimate;
2. expenditure for this period;
3. cumulative expenditure up to the end of the reporting period;
4. estimate to completion.

The sum of three and four generates the new estimate, i.e. item one for the following month. The estimate to completion should be re-evaluated each reporting period and not obtained by subtracting the cumulative expenditure from the previous estimate.

Work package expenditure should be summarised in progress tables as shown in Table 3.4.1. Work package identifiers and names are listed in columns one and two. The estimate for the work package in the current SPMP is placed in column three. Subsequent columns contain, for each month, the values of the previous estimate (PrEst), expenditure for the period (ExpP), cumulative expenditure (Cum) and estimate to completion (ToGo). The total in the last column states the current estimated effort for the work package.

Table 3.4.1 shows that the effort required for work packages 2210 and 2220 was underestimated. This underestimate of 10 man days for WP 2210 was detected and corrected in January. The estimate for WP 2220 had to be revised upwards by 5 man days in January and increased again by another 5 man days in February.

WPid	Name	Plan	January		February		March		Total
			PrEst	ToGo	PrEst	ToGo	PrEst	ToGo	
			ExpP	Cum	Exp	Cum	Exp	Cum	
2210	Logical	20	20	10	30	0	30	0	30
	Model		20	20	10	30	0	30	
2220	Prototype	30	30	15	35	5	40	0	40
			20	20	15	35	5	40	
2230	SRD	20	20	20	20	10	20	0	20
	draft		0	0	10	10	10	20	
2240	SRD	5	5	5	5	5	5	0	5
	final		0	0	0	0	5	5	
2200	SR phase	75	75	50	90	20	95	0	95
	total		40	40	35	75	20	95	

Table 3.4.1: Example Progress Table for March

Progress charts plot the estimates against time. They are best used for summarising the trends in the costs of high level work packages. Figure 3.4.1 is an example of a plot showing the trends in the estimates for WP 2200.

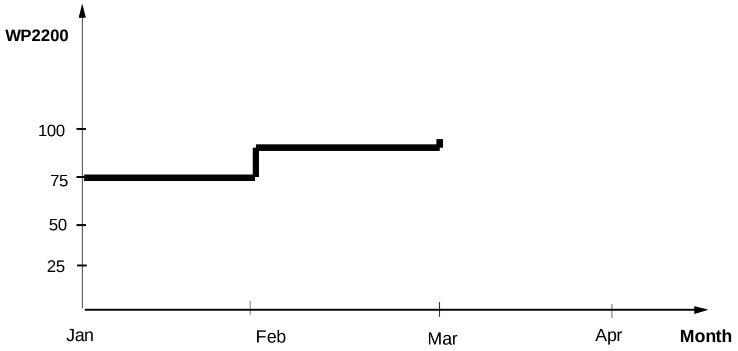


Figure 3.4.1: Example of a Progress Chart

The construction of progress tables and charts every reporting period enables:

- the accuracy of estimates to be measured;
- the degree of completion of each work package to be assessed.

The process of assessing every month what has been accomplished and re-estimating what remains is essential for keeping good control of project resources. Overspend trends are revealed and pinpointed early enough for timely corrective action to be taken.

3.4.2 Milestone trend charts

Milestone trend charts are used to report when milestones have been, or will be, achieved. First, an initial estimate of each milestone date is made. This estimate is put in the schedule section of the SPMP. Then, at the end of every reporting period, the following data are collected:

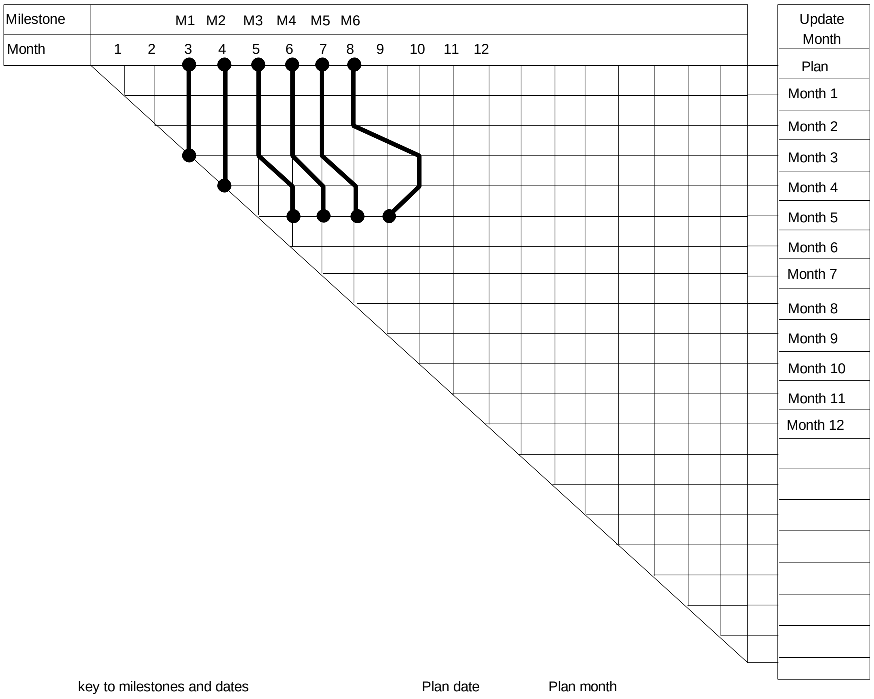
1. previous estimates of milestone achievement dates;
2. new estimate of milestone achievement dates.

The dates should be plotted in a milestone trend chart to illustrate the changes, if any, of the dates. A sample form is provided in Appendix D.

A milestone trend chart for the example project described in Figure 3.4.1 is shown in Figure 3.4.2. The vertical scale shows that the reporting period is one month. Figure 3.4.2 shows the status after five months. Milestones M1 and M2 have been achieved on schedule. Milestones M3, M4 and M5 have been slipped one month because 'Integration stage 2' took one month longer than expected. Milestone M6, the end of phase, was slipped two months when it was announced that the simulator required for system testing was going to be supplied two months late. Complaints to the supplier resulted in the delivery date being brought forward a month. M6 then moved forward one month.

The example shows how vertical lines indicate that the project is progressing according to schedule. Sloping lines indicate changes. Milestone trend charts are a powerful tool for understanding what is happening in a project. A milestone that slips every reporting period indicates that a serious problem exists.

Milestone trend charts should show milestone dates relative to the current approved plan, not obsolete plans. They should be reinitialised when the SPMP is updated.



key to milestones and dates		Plan date	Plan month
M1	DDD complete	17/03	3
M2	Integration stage 1 end	28/04	4
M3	Integration stage 2 end	26/05	5
M4	Integration stage 3 end	23/06	6
M5	Integration stage 4 end	21/07	7
M6	DD phase end	28/08	8

Figure 3.4.2: Milestone trend chart for the DD phase of a project

This page is intentionally left blank.

CHAPTER 4
SOFTWARE PROJECT MANAGEMENT TOOLS

4.1 INTRODUCTION

This chapter discusses tool support for the methods described in chapters 2 and 3. Tools are available for project planning, project risk analysis, project reporting and process support.

4.2 PROJECT PLANNING TOOLS

As well as the general purpose project planning tools that support activity definition, PERT and scheduling, specialised project planning tools are available for constructing process models and estimating software project costs.

4.2.1 General purpose project planning tools

General purpose project planning tools normally support:

- defining work packages;
- defining resources;
- defining resource availability;
- allocating resources to work packages;
- defining the duration of work packages;
- constructing activity networks using the PERT method;
- defining the critical path;
- defining the schedule in a Gantt chart.

Managers enter the project activities and define the resources that they require. They then mark the dependencies between the activities. The tool constructs the activity network and Gantt chart. The tools should also include the ability to:

- compute the total amount of resources required;
- provide resource utilisation profiles;
- fix the duration of an activity;
- divide a project into subprojects;
- highlight resource conflicts, or over-utilisation;
- integrate easily with word processors.

Common deficiencies of general purpose project planning tools are:

- fixed rate scheduling, so that work packages have to be split when someone takes a holiday;
- inability to handle fractional allocations of resources (e.g. resource conflicts occur when staff have to share their time between concurrent work packages);
- identifiers that change every time a new work package is inserted;
- too much line crossing in activity networks.

Advanced project planning tools permit variable rate scheduling. The project manager only needs to define:

- the total effort involved in the work package;
- the resources to be allocated to the work package;
- the availability of the allocated resources.

The project planning tool then decides what the resource utilisation profile should be.

4.2.2 Process modelling tools

Process modelling methods are relatively new, and the tools that support them are consequently immature. Process modelling tools should:

- support the definition of procedures;
- contain a library of 'process templates' that can be tailored to each project;
- make the process model available to a process support tool (see Section 4.5).

There are few dedicated software process modelling tools. Analysis tools that support the structured analysis techniques of data flow diagrams and entity relationship diagrams can be effective substitutes.

4.2.3 Estimating Tools

Project costs should be stored in a spreadsheet or database for access during software cost estimating. Project managers estimate by comparing their project with those in the database and extracting the cost data for the most similar project.

Some software cost estimation tools use measurements of software size (e.g. number of lines of code, number of function points) to produce the initial estimate of effort. They then accept estimates of cost drivers (e.g. required reliability, programmer experience) and use them to adjust the initial estimate.

In addition, software cost estimation tools may:

- permit 'what-if' type exploration of costs and timescales;
- produce estimates of the optimum timescale;
- allow input of a range of values instead of a single value, thereby enabling the accuracy to be estimated;
- provide explanations of how estimates were arrived at;
- provide estimates even when some information is absent;
- allow predicted values to be replaced by actual values, thereby allowing progressive refinement of an estimate as the project proceeds;
- permit backtracking to an earlier point in the cost estimation process and subsequent input of new data.

4.3 PROJECT RISK ANALYSIS TOOLS

Risk analysis tools have been applied in other fields, but difficulties in accurately quantifying software risks limits their usefulness.

One approach to risk analysis is to use heuristics or 'rules-of-thumb', derived from experience. Tools are available that have been programmed with the rules related to the common risks to a project. The tools ask a series of questions about the project and then report the most likely risks. Tools of this kind have rarely been used in software development.

4.4 PROJECT REPORTING TOOLS

Project planning tools (see Section 4.2) that are capable of recording the actual resources used, and the dates that events occurred, can be useful for progress reporting. Such tools should use the information to:

- mark up the Gantt chart to indicate schedule progress;
- constrain replanning.

Spreadsheets are useful for constructing progress tables and charts.

4.5 PROCESS SUPPORT TOOLS

Process support tools help guide, control and automate project activities [Ref 8]. They need a formally defined process model in a suitable machine readable format. The absence of complete software process models has limited the application of process support tools.

Process support tools may be integrated into the software engineering environment, coordinating the use of other software tools and guiding and controlling developers through the user interface [Ref 5]. Outside software engineering, process support tools have been used for controlling the work flow in a business process.

Process support tools should provide the following capabilities:

- instantiation of the process model;
- enactment of each process model instance (i.e. the process support tool should contain a 'process engine' that drives activities)
- viewing of the process model instance status.

In addition process support tools should provide:

- interfaces between the process model instance and the actors (i.e. the people playing the project roles);
- interfaces to project planning tools so that coverage of the plan, and the resource expenditure, can be tracked.

CHAPTER 5
THE SOFTWARE PROJECT MANAGEMENT PLAN

5.1 INTRODUCTION

All software project management activities must be documented in the Software Project Management Plan (SPMP) (SPM01). The SPMP is the controlling document for managing a software project.

The SPMP contains four sections dedicated to each development phase. These sections are called:

- Software Project Management Plan for the SR phase (SPMP/SR);
- Software Project Management Plan for the AD phase (SPMP/AD);
- Software Project Management Plan for the DD phase (SPMP/DD);
- Software Project Management Plan for the TR phase (SPMP/TR).

Each section of the SPMP must:

- define the project organisation (SPM01);
- define the managerial process (SPM01);
- outline the technical approach, in particular the methods, tools and techniques (SPM01);
- define the work-breakdown structure (SPM01, SPM10);
- contain estimates of the effort required to complete the project (SPM01, SPM04, SPM06, SPM07, SPM09);
- contain a planning network describing when each work package will be started and finished (SPM01, SPM11);
- allow for the risks to the project (SPM01).

The table of contents for each section of the SPMP is described in Section 5.6. This table of contents is derived from the IEEE Standard for Software Project Management Plans (ANSI/IEEE Std 1058.1-1987).

5.2 STYLE

The SPMP should be plain, concise, clear, consistent and modifiable.

5.3 RESPONSIBILITY

The developer is normally responsible for the production of the SPMP. The software project manager should write the SPMP.

5.4 MEDIUM

It is usually assumed that the SPMP is a paper document. There is no reason why the SPMP should not be distributed electronically to people with the necessary equipment.

5.5 SERVICE INFORMATION

The SR, AD, DD and TR sections of the SPMP are produced at different times in a software project. Each section should be kept separately under configuration control and contain the following service information:

a - Abstract
b - Table of Contents
c - Document Status Sheet
d - Document Change records made since last issue

5.6 CONTENTS

ESA PSS-05-0 recommends the following table of contents for each phase section of the SPMP:

1 Introduction
    1.1 Project overview
    1.2 Project deliverables
    1.3 Evolution of the SPMP
    1.4 Reference materials
    1.5 Definitions and acronyms

2 Project Organisation
    2.1 Process model
    2.2 Organisational structure
    2.3 Organisational boundaries and interfaces
    2.4 Project responsibilities

3 Managerial Process
    3.1 Management objectives and priorities
    3.2 Assumptions, dependencies and constraints
    3.3 Risk management
    3.4 Monitoring and controlling mechanisms
    3.5 Staffing plan

4 Technical Process
    4.1 Methods, tools and techniques
    4.2 Software documentation
    4.3 Project support functions

5 Work Packages, Schedule, and Budget
    5.1 Work packages
    5.2 Dependencies
    5.3 Resource requirements
    5.4 Budget and resource allocation
    5.5 Schedule

Material unsuitable for the above contents list should be inserted in additional appendices. If there is no material for a section then the phrase 'Not Applicable' should be inserted and the section numbering preserved.

5.6.1 SPMP/1 Introduction

5.6.1.1 SPMP/1.1 Project overview

This section of the SPMP should provide a summary of the project:

- objectives;
- deliverables;
- life cycle approach;
- major activities;
- milestones;
- resource requirements;
- schedule;
- budget.

This section outlines the plan for whole project and must be provided in the SPMP/SR (SPM01). The overview may be updated in subsequent phases.

5.6.1.2 SPMP/1.2 Project deliverables

This section should list the deliverables of the phase. All ESA PSS-05-0 documents, plans and software releases that will be delivered should be listed. Any other deliverable items such as prototypes, demonstrators and tools should be included.

5.6.1.3 SPMP/1.3 Evolution of the SPMP

This section should summarise the history of the SPMP in this and previous phases of the project.

This section should describe the plan for updating the SPMP in this and subsequent phases of the project.

5.6.1.4 SPMP/1.4 Reference materials

This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included.

5.6.1.5 SPMP/1.5 Definitions and acronyms

This section should provide the definitions of all terms, acronyms, and abbreviations used in the plan, or refer to other documents where the definitions can be found.

5.6.2 SPMP/2 Project Organisation

5.6.2.1 SPMP/2.1 Process model

This section should define the activities in the phase and their inputs and outputs. The definition should include the major project functions (i.e. activities that span the entire duration of the project, such as project management, configuration management, verification and validation, and quality assurance) and the major production activities needed to achieve the objectives of the phase. The definition of the process model may be textual or graphic.

5.6.2.2 SPMP/2.2 Organisational structure

This section should describe the internal management structure of the project in the phase. Graphical devices such as organigrams should be used to show the lines of reporting, control and communication.

Roles that often appear in the internal management structure are:

- project manager;
- team leader;
- programmers;
- software librarian;
- software quality assurance engineer.

5.6.2.3 SPMP/2.3 Organisational boundaries and interfaces

This section should describe the relationship between the project and external groups during the phase such as:

- parent organisation;
- client organisation;
- end users;
- subcontractors;
- suppliers;
- independent verification and validation organisation;
- independent quality assurance organisations.

The procedures and responsibilities for the control of each external interface should be summarised. For example:

- name of Interface Control Document (ICD);
- those responsible for the agreement of the ICD;
- those responsible for authorising the ICD.

5.6.2.4 SPMP/2.4 Project responsibilities

This section should define the roles identified in the organisational structure and its boundaries. Each role definition should briefly describe the purpose of the role and list the responsibilities.

5.6.3 SPMP/3 Managerial process

5.6.3.1 SPMP/3.1 Management objectives and priorities

This section should define the management objectives of this project phase and their relative priorities. They should discuss any trade-offs between the objectives.

5.6.3.2 SPMP/3.2 Assumptions, dependencies and constraints

This section should state for the phase the:

- assumptions on which the plan is based;
- external events the project is dependent upon;
- constraints on the project.

Technical issues should only be mentioned if they have an effect on the plan.

Assumptions, dependencies and constraints are often difficult to distinguish. The best approach is not to categorise them but to list them. For example:

- limitations on the budget;
- schedule constraints (e.g. launch dates);
- constraints on the location of staff (e.g. they must work at developer's premises);
- commercial hardware or software that will be used by the system;
- availability of simulators and other test devices;
- availability of external systems with which the system must interface.

5.6.3.3 SPMP/3.3 Risk management

This section of the plan should identify and assess the risks to the project, and describe the actions that will be taken in this phase to manage them. A risk table (see Section 3.3.1) may be used.

5.6.3.4 SPMP/3.4 Monitoring and controlling mechanisms

This section of the plan should define the monitoring and controlling mechanisms for managing the work. Possible monitoring and controlling mechanisms are:

- work package descriptions;
- work package completion reports;
- progress reports;
- reviews;
- audits.

This section should define or reference the formats for all documents and forms used for monitoring and controlling the project.

This section should specify the:

- frequency of progress meeting with initiators and management;
- frequency of submission of progress reports;
- modifications (if any) to the progress report template provided in Chapter 6;
- general policy regarding reviews and audits (the details will be in the SVVP).

5.6.3.5 SPMP/3.5 Staffing plan

This section of the plan should specify the names, roles and grades of staff that will be involved in the phase. The utilisation of the staff should be given for each reporting period. A staff profile giving the total number of staff on the project each month may also be given.

5.6.4 SPMP/4 Technical Process

5.6.4.1 SPMP/4.1 Methods, tools and techniques

This section should specify the methods, tools and techniques to be used to produce the phase deliverables.

5.6.4.2 SPMP/4.2 Software documentation

This section should define or reference the documentation plan for the phase. For each document to be produced, the documentation plan should specify:

- document name;
- review requirements;
- approval requirements.

Some documents will be deliverable items. Others will be internal documents, such as technical notes, or plans, such as the SCMP. All documents should be listed.

The documentation plan may contain or reference a 'style' guide that describes the format and layout of documents.

5.6.4.3 SPMP/4.3 Project support functions

This section should contain an overview of the plans for the project support functions of:

- software configuration management;
- software verification and validation;
- software quality assurance.

This section should reference the SCMP, SVVP and SQAP.

5.6.5 SPMP/5 Work Packages, Schedule, and Budget

5.6.5.1 SPMP/5.1 Work packages

This section should describe the breakdown of the phase activities into work packages.

This section may begin with a Work Breakdown Structure (WBS) diagram to describe the hierarchical relationships between the work packages. Each box should show the title and identifier of the work package. Alternatively, the work breakdown may be described by listing the work package titles and identifiers as shown in Section 2.4.2.2.

The full Work Package Descriptions may be contained in this section or put in an appendix. Each Work Package description should define the:

- work package title;
- work package reference number;
- responsible organisation;
- major constituent activity;
- work package manager;
- start event;
- end event;
- inputs;
- activities;
- outputs.

Work packages should be defined for all activities, including project functions such as project management, configuration management, verification and validation and quality assurance.

5.6.5.2 SPMP/5.2 Dependencies

This section should define the ordering relations between the work packages. This may be done by using a planning network technique, such as the Program Evaluation and Review Technique (PERT), to order the execution of work packages according to their dependencies (see Section 3.5). Dependency analysis should define the critical path to completion of the project and derive the 'float' for activities off the critical path.

5.6.5.3 SPMP/5.3 Resource requirements

This section should describe, for each work package:

- the total resource requirements;
- the resource requirements as a function of time.

Labour resources should be evaluated in man-hours, man-days or man-months. Other resources should be identified (e.g. equipment).

5.6.5.4 SPMP/5.4 Budget and resource allocation

The project budget allocation should be specified by showing how the available financial resources will be deployed on the project. This is normally done by providing a table listing the amount of money budgeted for each major work package. The manner in which this section is completed depends upon the contractual relationship.

5.6.5.5 SPMP/5.5 Schedule

This section should define when each work package starts and ends. This is normally done by drawing a Gantt chart (see Section 3.6).

This section should describe the milestones in the project, providing for each milestone:

- an identifier;
- a description (e.g. a list of deliverables);
- the planned date of achievement;
- the actual date of achievement (for plan updates).

Milestones should be marked on or alongside the Gantt chart.

5.7 EVOLUTION

Project planning is a continuous process. Project managers should review their plan when:

- making progress reports;
- new risks are identified;
- problems occur.

It is good practice to draft sections of the plan as soon as the necessary information is available, and not wait to begin planning until just before the start of the phase that the SPMP section applies to. Some parts of the SPMP/DD need to be drafted at the start of the project to arrive at a total cost estimate, for example.

The evolution of the plan is closely connected with progress reporting (see Chapter 6). Reports are produced and reviewed. Reports may propose changes to the plan. Alternatively, changes to the plan may be defined during the progress meeting.

5.7.1 UR phase

By the end of the UR review, the SR phase section of the SPMP must be produced (SPMP/SR) (SPM02). The SPMP/SR describes, in detail, the project activities to be carried out in the SR phase.

As part of its introduction, the SPMP/SR must outline a plan for the whole project (SPM03), and provide a rough estimate of the total cost. The project manager will need to draft the work breakdown, schedule and budget section of the SPMP/AD, SPMP/DD and SPMP/TR at the start of the project to provide the overview and estimate the total cost of the project.

A precise estimate of the effort involved in the SR phase must be included in the SPMP/SR (SPM04). Specific factors affecting estimates for the work required in the SR phase are the:

- number of user requirements;
- level of user requirements;
- stability of user requirements;
- level of definition of external interfaces;
- quality of the URD.

An estimate based simply on the number of user requirements might be very misleading - a large number of detailed low-level user requirements might be more useful, and save more time in the SR phase, than a few high-level user requirements. A poor quality URD with few requirements might imply that a lot of requirements analysis is required in the SR phase.

5.7.2 SR phase

During the SR phase, the AD phase section of the SPMP must be produced (SPMP/AD) (SPM05). The SPMP/AD describes, in detail, the project activities to be carried out in the AD phase.

As part of its introduction, the SPMP/AD must include an outline plan for the rest of the project, and provide an estimate of the total cost of the whole project accurate to at least 30% (SPM06). The project manager will need to draft the work breakdown, schedule and budget section of the SPMP/DD and SPMP/TR at the end of the SR phase to provide the overview and estimate the total cost of the project.

A precise estimate of the effort involved in the AD phase must be included in the SPMP/AD (SPM07). Specific factors that affect estimates for the work required in the AD phase are the:

- number of software requirements;
- level of detail of software requirements;
- stability of software requirements;
- level of definition of external interfaces;
- quality of the SRD.

If an evolutionary development life cycle approach is to be used, then this should be stated in the SPMP/AD.

5.7.3 AD phase

During the AD phase, the DD phase section of the SPMP must be produced (SPMP/DD) (SPM08). The SPMP/DD describes, in detail, the project activities to be carried out in the DD phase.

An estimate of the total project cost must be included in the SPMP/DD (SPM09). An accuracy of 10% should be aimed at.

The SPMP/DD must contain a WBS that is directly related to the decomposition of the software into components (SPM10).

The SPMP/DD must contain a planning network (i.e. activity network) showing the relationships between the coding, integration and testing activities (SPM11).

5.7.4 DD phase

As the detailed design work proceeds to lower levels, the WBS and job schedule need to be refined to reflect this. To achieve the necessary level of visibility, software production work packages in the SPMP/DD must not last longer than one man-month (SPM12).

During the DD phase, the TR phase section of the SPMP must be produced (SPMP/TR) (SPM13). The SPMP/TR describes, in detail, project activities until final acceptance, in the OM phase.

CHAPTER 6
THE SOFTWARE PROJECT PROGRESS REPORT

6.1 INTRODUCTION

Accurate and timely reporting is essential for the control of a project. Software project managers should produce progress reports at regular intervals or when events occur that meet agreed criteria. The progress report is an essential tool for controlling a project because:

- preparing the progress report forces re-evaluation of the resources required to complete the project;
- the progress report provides initiators and senior management with visibility of the status of the project.

Progress reports describe the project status in relation to the applicable Software Project Management Plan. They should cover technical status, resource status, schedule status, problems and financial status (which may be provided separately).

Progress reports should be distributed to senior management and the initiator. The progress report sent to senior management should contain the information described in Section 6.6. Although the progress report sent to the initiator should have the same structure, the level of detail provided may not be the same because of the contractual situation. For example in a fixed price contract, it is acceptable to omit detailed information about resource status and financial status.

6.2 STYLE

The progress report should be plain, concise, clear, complete and consistent. Brevity, honesty and relevance are the watchwords of the good progress report.

6.3 RESPONSIBILITY

The software project manager is normally responsible for the production of the progress report.

6.4 MEDIUM

It is usually assumed that the progress report is a paper document. There is no reason why it should not be distributed electronically to people with the necessary equipment.

6.5 SERVICE INFORMATION

Progress reports should contain the following service information:

a - Abstract
b - Table of Contents

6.6 CONTENTS

The following table of contents is recommended for the progress report:

1 Introduction
 1.1 Purpose
 1.2 Summary
 1.3 References
 1.4 Definitions and acronyms

2 Technical status
 2.1 Work package technical status
 2.2 Configuration status
 2.3 Forecast for next reporting period

3 Resource status
 3.1 Staff utilisation
 3.2 Work package resource status
 3.3 Resource summary

4 Schedule status
 4.1 Milestone trends
 4.2 Schedule summary

5 Problems

6 Financial status report
    6.1 Costs for the reporting period
    6.2 Cost to completion
    6.3 Limit of liability
    6.4 Payments

6.6.1 Progress Report/1 Introduction

6.6.1.1 Progress Report/1.1 Purpose

This section should describe the purpose of the report. This section should include:

- the name of the project;
- a reference to the applicable SPMP;
- a reference to the applicable contract (if any);
- a statement of the reporting period.

6.6.1.2 Progress Report/1.2 Summary

This section should summarise the main activities and achievements during the reporting period.

Any change in the total cost to completion should be stated in this section. Changes to work package resource estimates should be summarised.

6.6.1.3 Progress Report/1.3 References

This section should list any documents referenced in the report or applicable to it (e.g. this guide, the SPMP etc).

6.6.1.4 Progress Report/1.4 Definitions and acronyms

This section should list any special terms, acronyms or abbreviations used in the report and explain their meaning.

6.6.2 Progress Report/2 Technical status

6.6.2.1 Progress Report/2.1 Work package technical status

This section should list the work packages completed during the reporting period. There should be some indication of whether each work package was completed successfully or not.

This section should list the work packages continued or started during the reporting period. The outstanding tasks for each work package should be identified.

6.6.2.2 Progress Report/2.2 Configuration status

This section should summarise the changes to the configuration status in the reporting period, e.g:

- RID statistics;
- issues and revisions of documents;
- SCR, SPR and SMR statistics;
- releases of software.

6.6.2.3 Progress Report/2.3 Forecast for next reporting period

This section should forecast what technical progress is expected on each work package in the next reporting period.

Any events that are expected to take place in the reporting period should be described (e.g. milestones, deliveries).

6.6.3 Progress Report/3 Resource status

6.6.3.1 Progress Report/3.1 Staff utilisation

This section should comprise:

- a table of hours booked by each member of the team;
- the hours staff have booked for a number of past reporting periods (e.g. from the beginning of the current phase, or beginning of the year, or beginning of the project, as appropriate);
- a forecast for the man-hours staff will book for future reporting periods (e.g. up to the end of the current phase, or the end of the project, or other milestone).

6.6.3.2 Progress Report/3.2 Work package resource status

This section should contain a work package progress table (see Section 3.7.1) summarising for each work package:

- previous estimate of the effort required for completion;
- effort expended in this period;
- cumulative effort expenditure up to the end of the reporting period;
- estimate of remaining effort required for completion.

6.6.3.3 Progress Report/3.3 Resource status summary

This section should present the aggregated effort expenditure for the reporting period for:

- the project;
- subsystems (for medium and large-size projects).

6.6.4 Progress Report/4 Schedule status

6.6.4.1 Progress Report/4.1 Milestone trend charts

This section should provide an updated milestone trend chart (see Section 3.4.2) for all the milestones described in the SPMP section to which this progress report applies. All changes to milestone dates made in this reporting period should be explained and justified.

6.6.4.2 Progress Report/4.2 Schedule summary

This section should provide an updated bar chart (i.e. Gantt chart) marked to show work packages completed and milestones achieved. Progress on partially completed work packages may also be shown.

6.6.5 Progress Report/5 Problems

This section should identify any problems which are affecting or could affect progress. These may include technical problems with the software under development, environmental problems (e.g. computers, communications, and accommodation) and resource problems (e.g. staffing problems). This list is not exhaustive.

6.6.6 Progress Report/6 Financial status report

The Financial Status Report (sometimes called the Cost Report) provides the total amounts, in currency units, to be billed during the reporting period. The Financial Status Report may be provided separately.

6.6.6.1 Progress Report/6.1 Costs for the reporting period

This section should provide a table listing the hours worked, rates and costs (hours worked multiplied by rate) for each team member. Total labour costs should be stated.

For ESA contracts, labour costs may have to be calculated in base rates and price variation rates, determined by applying a contractually specified price-variation formula to the base rate.

Non-labour costs should be listed and a total stated.

6.6.6.2 Progress Report/6.2 Cost to completion

This section should state the accumulated cost so far and the planned cost to completion of the project and project phase.

A progress chart plotting the cost to completion values reported since the start of the project or project phase may be provided (see Section 3.4.1).

6.6.6.3 Progress Report/6.3 Limit of liability

This section states:

- the Limit Of Liability (LOL);
- how much has been spent;
- how much there is to go.

6.6.6.4 Progress Report/6.4 Payments

This section should state what payments are due for the reporting period (e.g. contractual stage payments on achievement of a given milestone).

APPENDIX A
GLOSSARY

Terms used in this document are consistent with ESA PSS-05-0 [Ref 1] and ANSI/IEEE Std 610.12-1990 [Ref 2].

A.1 LIST OF ACRONYMS

AD	Architectural Design
AD/R	Architectural Design Review
ADD	Architectural Design Document
AMI	Application of Metrics in Industry
ANSI	American National Standards Institute
AT	Acceptance Test
BSSC	Board for Software Standardisation and Control
CASE	Computer Aided Software Engineering
COCOMO	Constructive Cost Model
DCR	Document Change Record
DD	Detailed Design and production
DD/R	Detailed Design and production Review
DDD	Detailed Design and production Document
ESA	European Space Agency
FPA	Function Point Analysis
ICD	Interface Control Document
IEEE	Institute of Electrical and Electronics Engineers
IT	Integration Test
LOL	Limit Of Liability
PERT	Program Evaluation and Review Technique
PSS	Procedures, Specifications and Standards
QA	Quality Assurance
RID	Review Item Discrepancy
SCMP	Software Configuration Management Plan
SCR	Software Change Request
SMR	Software Modification Report
SPR	Software Problem Report
SR	Software Requirements
SR/R	Software Requirements Review
SRD	Software Requirements Document
ST	System Test
SUT	Software Under Test
SVV	Software Verification and Validation

SVVP	Software Verification and Validation Plan
UR	User Requirements
UR/R	User Requirements Review
URD	User Requirements Document
UT	Unit Test

APPENDIX B
REFERENCES

1. ESA Software Engineering Standards, ESA PSS-05-0 Issue 2, February 1991.

2. IEEE Standard Glossary of Software Engineering Terminology, ANSI/IEEE Std 610.12-1990.

3. IEEE Standard for Software Project Management Plans, ANSI/IEEE Std 1058.1-1987.

4. Managing the Software Process, Watts S. Humphrey, SEI Series in Software Engineering, Addison-Wesley, August 1990.

5. Reference Model for Frameworks of Software Engineering Environments, European Computer Manufacturer's Association, TR/55, 1991.

6. Evolution of the ESA Software Engineering Standards, J.Fairclough and C.Mazza, in Proceedings of the IEEE Fourth Software Engineering Standards Application Workshop, 1991.

7. Software Process Themes and Issues, M. Dowson, in Proceedings of 2nd International Conference on the Software Process, IEEE Computer Society Press, 1993.

8. Tool Support for Software Process Definition and Enactment, C. Fernström, Proceedings of the ESA/ESTEC workshop on European Space Software Development Environment, ESA, 1992.

9. Software Engineering Economics, B.Boehm, Prentice-Hall, 1981

10. The Mythical Man-Month, F. Brooks, Addison-Wesley, 1975

11. Software Function, source lines of code and development effort prediction - a software science validation, A. Albrecht and J. Gaffney Jr, IEEE Transactions on Software Engineering, SE-9, (6), 1983.

12. SOCRAT, Software Cost Resources Assessment Tool, ESA Study Contract Report, J.Fairclough, R. Blake, I. Alexander, 1992.

13. Function Point Analysis: Difficulties and Improvements, IEEE Transactions on Software Engineering, Vol 14, 1, 1988.

14. Managing Computer Projects, R. Gibson, Prentice-Hall, 1992.

15. Software Sizing and Estimating, C.R. Symons, Wiley, 1991.

16. Project Control Requirements and Procedures for Medium-Size Projects, ESA PSS-38, Issue 1, December 1977.

17. The Nature of Managerial Work, H. Mintzberg, Prentice-Hall, 1980.

18. Applications of Metrics in Industry Handbook, a quantitative approach to software management, Centre for Systems and Software Engineering, South Bank University, London, 1992.

19. Guide to the Software Engineering Standards, ESA PSS-05-01, Issue 1, October 1991.

APPENDIX C
MANDATORY PRACTICES

This appendix is repeated from ESA PSS-05-0, appendix D.8.

SPM01 All software project management activities shall be documented in the Software Project Management Plan (SPMP).

SPM02 By the end of the UR review, the SR phase section of the SPMP shall be produced (SPMP/SR).

SPM03 The SPMP/SR shall outline a plan for the whole project.

SPM04 A precise estimate of the effort involved in the SR phase shall be included in the SPMP/SR.

SPM05 During the SR phase, the AD phase section of the SPMP shall be produced (SPMP/AD).

SPM06 An estimate of the total project cost shall be included in the SPMP/AD.

SPM07 A precise estimate of the effort involved in the AD phase shall be included in the SPMP/AD.

SPM08 During the AD phase, the DD phase section of the SPMP shall be produced (SPMP/DD).

SPM09 An estimate of the total project cost shall be included in the SPMP/DD.

SPM10 The SPMP/DD shall contain a WBS that is directly related to the decomposition of the software into components.

SPM11 The SPMP/DD shall contain a planning network showing relationships of coding, integration and testing activities.

SPM12 No software production work packages in the SPMP/DD shall last longer than 1 man-month.

SPM13 During the DD phase, the TR phase section of the SPMP shall be produced (SPMP/TR).

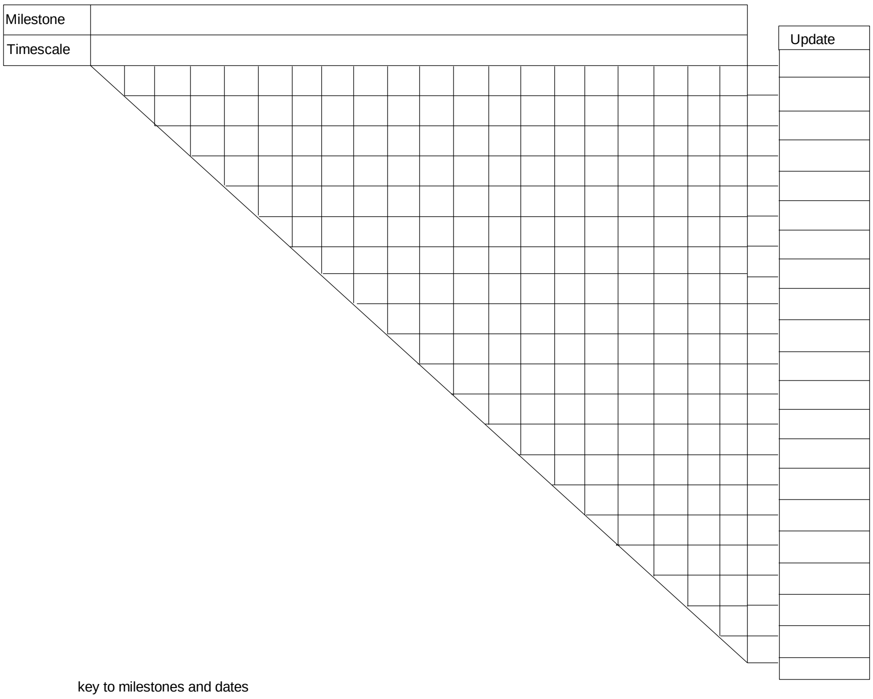
This page is intentionally left blank.

APPENDIX D
PROJECT MANAGEMENT FORMS

D.1 WORK PACKAGE DESCRIPTION

PROJECT:	PHASE:	W.P. REF:
W.P. TITLE:		
CONTRACTOR:		SHEET 1 OF 1
MAJOR CONSTITUENT:		
START EVENT:	PLANNED DATE:	ISSUE REF:
END EVENT:	PLANNED DATE:	ISSUE DATE:
W.P. MANAGER		
Inputs		
Activities		
Outputs		

D.2 MILESTONE TREND CHART



key to milestones and dates

M1

M2

M3

M4

M5

M6

M7

This page is intentionally left blank.

APPENDIX E
INDEX