Chapter 4.

THE DEFINITION PHASE

PART 2_2

Summary

6. Project Estimating Guidelines

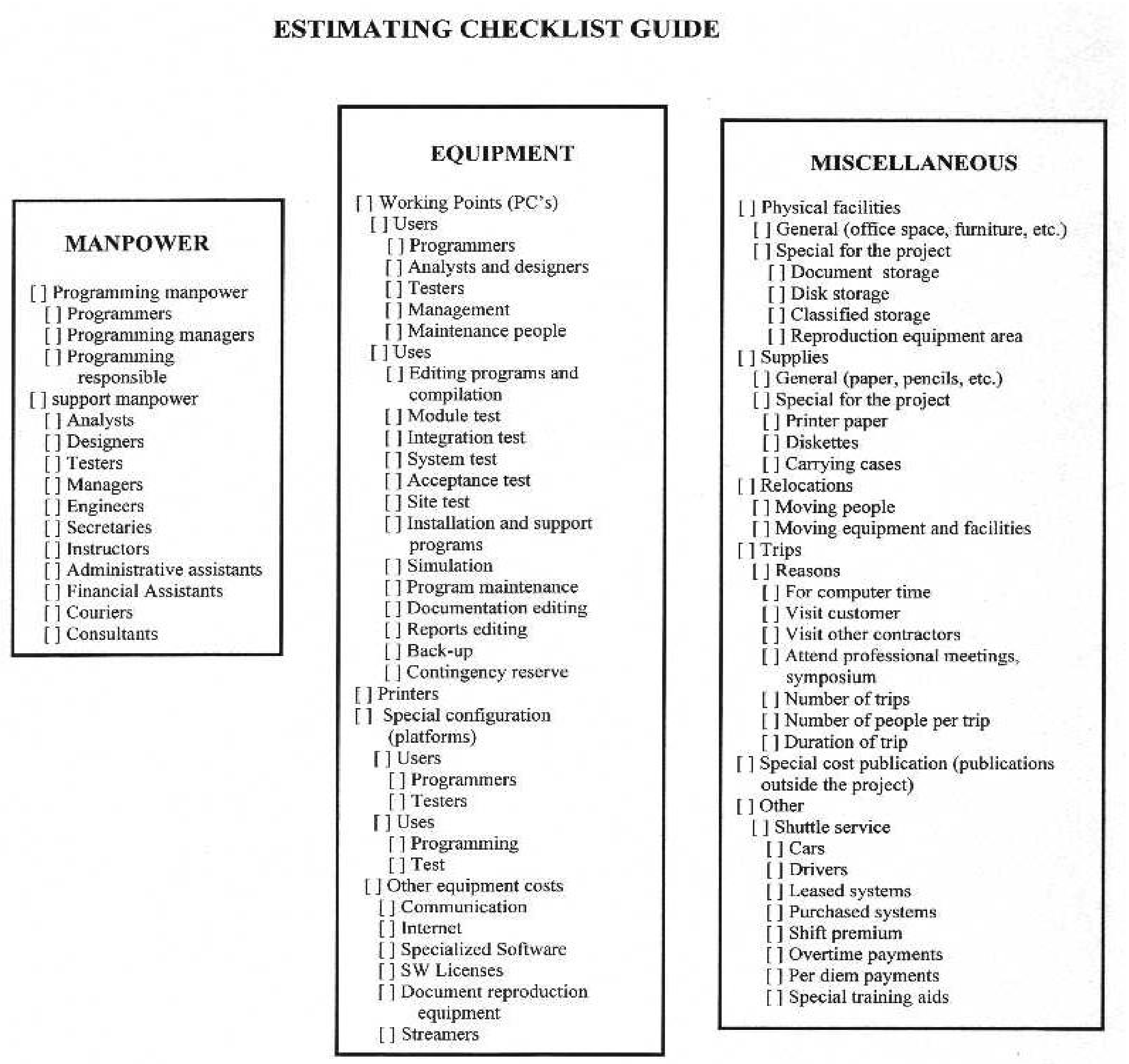
- Estimating the *size of the job* to be done is probably one of the toughest tasks a manager faces.
- Nobody has succeeded in coming up with a cookbook to make the estimating process mechanical or automatic
    - But there are some **steps** you can take to approach the problem with a reasonable degree of sanity.
- What is an **estimate**?
    - (1) An estimate is *your judgment of what a job will cost* in terms of man-years, calendar time, machine-hours, and other resources.
    - (2) An estimate is a *statement* of how you *plan* to expend resources in order to get the job done.
    - (3) When translated into money and calendar time, and adopted for your project, the estimate is called a *budget.*
- *Your **Project Plan** must evolve and change* whenever conditions change.
    - The same applies to your **estimate** or **budget**: If changes requiring more resources take place, you must reestimate and change the budget.
- Estimates will always be imperfect and will need *refining* as the project rolls along.
    - Ideally, your *contract* should be written to allow for estimating each new phase at the end of the preceding one.
- Here is one way to approach the estimating process.
    - Although this section logically is placed here, it will be more meaningful if you review it once you have studied the rest of the course.

- **STEP 1. *Design the program system*.**
    - "What does he mean, *design the system?*"
    - The truth is that you *can* design your system to *some level of detail*.
        - The design might be only four pages of charts requiring a day's effort but that's a whole lot better than nothing.

- You *must take the time to consider* the magnitude of the problem you're estimating.
    - It's impossible to estimate the cost of building something completely undefined.
- In fact, the key to successful estimating is understanding the problem.
    - The design you develop for your first estimating exercise *should be carried* to a level at which you have a feel for **all** the *major technical problems* that you will face.
- If you can take the time to design the program system in detail, that's great.
    - More likely, however, your design (at least for *initial* estimates) will be at a much broader level.

- **STEP 2. *Estimate the total size of the program system to be delivered*.**
    - By *size* we mean the *number of final*, deliverable **lines of code** (LOC) there will be, including code that defines **data files** as well as code that defines **executable machine instructions**.
    - For this purpose can be used the different *techniques and methods* we have disscused.
        - For **example**, based on *buttom-up technique* the size is determined this by estimating the size of each module in your design and then adding them together.
    - The more detailed your design, the more accurate this total estimate is likely to be.
        - If you plan to code in more than one language (for example, assembler language and a high level language), find a *separate total* for each.
    - If you are modifying an existing system, estimate *the lines of* ***new code*** that you will write.

- **STEP 3. *Estimate the programming manpower* required to produce the number of lines of code that you have just derived.** (fig. 6.a)



ESTIMATING CHECKLIST GUIDE

MANPOWER

- [ ] Programming manpower
  - [ ] Programmers
  - [ ] Programming managers
  - [ ] Programming responsible
- [ ] support manpower
  - [ ] Analysts
  - [ ] Designers
  - [ ] Testers
  - [ ] Managers
  - [ ] Engineers
  - [ ] Secretaries
  - [ ] Instructors
  - [ ] Administrative assistants
  - [ ] Financial Assistants
  - [ ] Couriers
  - [ ] Consultants

EQUIPMENT

- [ ] Working Points (PC's)
  - [ ] Users
    - [ ] Programmers
    - [ ] Analysts and designers
    - [ ] Testers
    - [ ] Management
    - [ ] Maintenance people
  - [ ] Uses
    - [ ] Editing programs and compilation
    - [ ] Module test
    - [ ] Integration test
    - [ ] System test
    - [ ] Acceptance test
    - [ ] Site test
    - [ ] Installation and support programs
    - [ ] Simulation
    - [ ] Program maintenance
    - [ ] Documentation editing
    - [ ] Reports editing
    - [ ] Back-up
    - [ ] Contingency reserve
- [ ] Printers
- [ ] Special configuration (platforms)
  - [ ] Users
    - [ ] Programmers
    - [ ] Testers
  - [ ] Uses
    - [ ] Programming
    - [ ] Test
  - [ ] Other equipment costs
    - [ ] Communication
    - [ ] Internet
    - [ ] Specialized Software
    - [ ] SW Licenses
    - [ ] Document reproduction equipment
    - [ ] Streamers

MISCELLANEOUS

- [ ] Physical facilities
  - [ ] General (office space, furniture, etc.)
  - [ ] Special for the project
    - [ ] Document storage
    - [ ] Disk storage
    - [ ] Classified storage
    - [ ] Reproduction equipment area
- [ ] Supplies
  - [ ] General (paper, pencils, etc.)
  - [ ] Special for the project
    - [ ] Printer paper
    - [ ] Diskettes
    - [ ] Carrying cases
- [ ] Relocations
  - [ ] Moving people
  - [ ] Moving equipment and facilities
- [ ] Trips
  - [ ] Reasons
    - [ ] For computer time
    - [ ] Visit customer
    - [ ] Visit other contractors
    - [ ] Attend professional meetings, symposium
    - [ ] Number of trips
    - [ ] Number of people per trip
    - [ ] Duration of trip
- [ ] Special cost publication (publications outside the project)
- [ ] Other
  - [ ] Shuttle service
    - [ ] Cars
    - [ ] Drivers
    - [ ] Leased systems
    - [ ] Purchased systems
    - [ ] Shift premium
    - [ ] Overtime payments
    - [ ] Per diem payments
    - [ ] Special training aids

Fig. 6.a. Estimating checklist

- Each organization has its own *rules of thumb* regarding *numbers of finished instructions per man-day* or *man-month*.
  - There is no published set of these numbers in the programming community better than the *informal rules* that you have in your own organization.
- If you use someone else's numbers, question them.
  - For ***example***, if someone says that a good rule of thumb is *five instructions per man-day*, you need to know several things about that number:

        - Does it cover only the programmer's time?
        - Does it include problem analysis and baseline design?
        - It includes module test?
        - Integration test?
        - System test?
        - Acceptance test?
        - Documentation?
        - Management time?
        - Support personnel?
        - What language is used?

- **STEP 4. *Estimate the support manpower requirements***
    - The support manpower items are presented in the fig.6.a, first column.
        - Not all the items listed in this and other checklists are necessarily applicable to *your job.*
    - Later the course will offer some *typical numbers of **support personnel***, but here again your own experience may be worth far more than any numbers you get from a book.

- **STEP 5. *Estimate equipment costs.***
    - Figure 6.a, second column, is intended to remind you of the various *categories of equipment* and users you might need to consider for your job.
        - Again, later chapters will have more to say on testing and machine time.

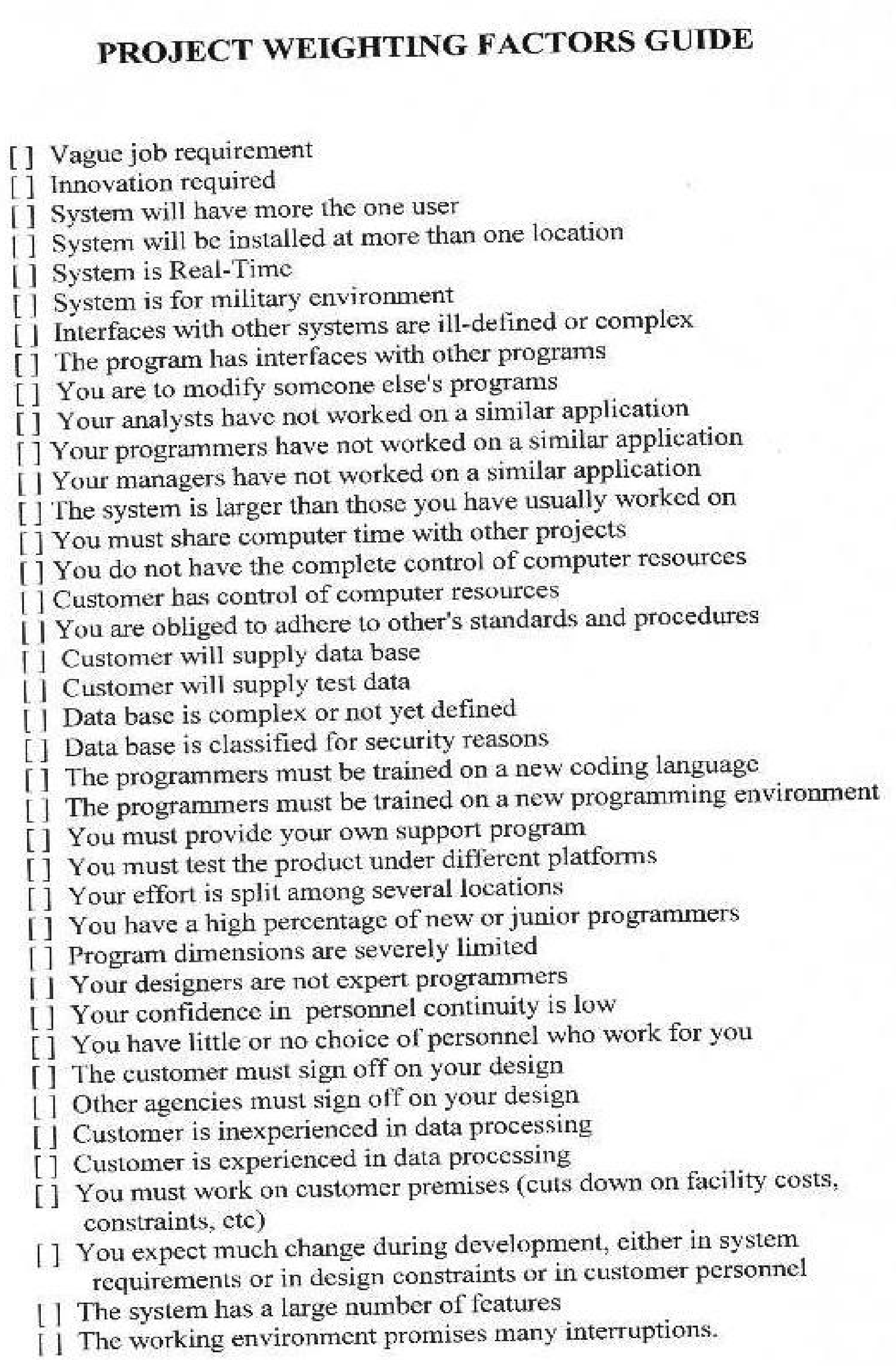
- **STEP 6. *Estimate the items referring to physical facilities, supplies, relocations, trips, etc.***
    - The main items of this category are listed in figure 6.a, column number 3.
    - Consider the items that apply to you.

- **STEP 7. Now that you've sized everything in a sort of bare-bones way, *go back and add whatever contingencies are appropriate for your organization.***

- For ***example***, you may have shown a requirement for *500 man-months of programmers' time*, but programmers *get sick, take vacations, quit, get fired, go on military duty, and so on*.
- These items may easily reduce the average person's effectiveness on your job by *20%*. If you *assume* 100% use of anybody's time, you'll be in trouble right off the bat.
- Decide what the *numbers* should be for your organization and use them to *modify* your estimates (either manpower or calendar time, or both).
- ***Write down*** *all your* ***contingency factors*** *so that others looking at your estimate will know what you did.*

- **STEP 8.** ***Consider weighting factors***

  - The Weighting Factors are listed din fig.6.b.



PROJECT WEIGHTING FACTORS GUIDE

[ ] Vague job requirement
[ ] Innovation required
[ ] System will have more the one user
[ ] System will be installed at more than one location
[ ] System is Real-Time
[ ] System is for military environment
[ ] Interfaces with other systems are ill-defined or complex
[ ] The program has interfaces with other programs
[ ] You are to modify someone else's programs
[ ] Your analysts have not worked on a similar application
[ ] Your programmers have not worked on a similar application
[ ] Your managers have not worked on a similar application
[ ] The system is larger than those you have usually worked on
[ ] You must share computer time with other projects
[ ] You do not have the complete control of computer resources
[ ] Customer has control of computer resources
[ ] You are obliged to adhere to other's standards and procedures
[ ] Customer will supply data base
[ ] Customer will supply test data
[ ] Data base is complex or not yet defined
[ ] Data base is classified for security reasons
[ ] The programmers must be trained on a new coding language
[ ] The programmers must be trained on a new programming environment
[ ] You must provide your own support program
[ ] You must test the product under different platforms
[ ] Your effort is split among several locations
[ ] You have a high percentage of new or junior programmers
[ ] Program dimensions are severely limited
[ ] Your designers are not expert programmers
[ ] Your confidence in personnel continuity is low
[ ] You have little or no choice of personnel who work for you
[ ] The customer must sign off on your design
[ ] Other agencies must sign off on your design
[ ] Customer is inexperienced in data processing
[ ] Customer is experienced in data processing
[ ] You must work on customer premises (cuts down on facility costs, constraints, etc)
[ ] You expect much change during development, either in system requirements or in design constraints or in customer personnel
[ ] The system has a large number of features
[ ] The working environment promises many interruptions.

Fig.6.b. Project Weighting Factors

- Check off *each item* that you think applies to your job.
- For each item checked, ask whether or not *you have already taken it into account* in your estimate.
  - If not, decide whether each is of sufficient value to cause you to increase your estimate.
- In each case there are various parts of your estimate that may be affected, but you can reasonably limit yourself to the two largest items:
  - (1) ***Manpower***
  - (2) ***Calendar time***.
- The list is rigged so that any item checked would normally suggest an *increase in your estimates.*
  - For any item not checked, it may mean that your estimate should be left alone, as far as that point is concerned, or it may suggest that negative weighting is in order —that is, you might *decrease* your original estimate.
  - But go easy — estimates in this business are very seldom too high.

- **STEP** ***9. Convert everything to money, by applying average salary rates, machine time rates, etc. for your organization.***
  - If pricing specialists handle this task for you, be sure that they *understand* what you have included and be sure you *know* what they may add on.
    - For ***example***, they may normally add the contingencies mentioned in Step 7.

- **STEP 10. The estimate you have so far is the** ***base cost for the job, sometimes called factory cost or direct cost.***
  - *Now, add your* **profit**, your **overhead expense**, and any **fees** not already included.
  - This is the ***completed cost estimate*** for the job. But you're not done yet.

- **STEP 11.** ***Write down the assumptions on which your estimate is based.***
  - Do not pass your estimate on to *anyone* without giving him the *assumptions*.
    - Nail the *estimate* and the *assumptions* together if you have to.

  - You may have made some assumptions that your boss can't live with or that are impossible to meet.
  - Your estimate may be *totally useless* if it's based on a bad assumption.
  - Don't include in very fine print an assumption that you *know is* impossible to meet in order to save your hide if the project gets in trouble later on. Besides being childish and unethical, it won't work.

- **STEP 12. As the job progresses through the development cycle, *reestimate all or portions of the job as you get better and better inputs.***
  - A major point for **reestimating** is at the end of the Design Phase.
    - At that time you should have not only a *complete system design* as the basis for your estimate but a solid *Project Plan* as well.
    - The process of writing the Project Plan will have greatly sharpened your awareness of all cost items you're likely to face.
  - Your **new estimate** may show a need for *simply rejuggling* your total resources (for example, using fewer people but more machine time); this is usually within a project manager's purview.
    - If, however, the new estimate shows a need for *additional resources*, you may have a problem.
  - Whether or not you can get the **additional resources** depends on the kind of *contract* you have and on the understanding and credibility existing between you and your customer.
    - In many cases you can demonstrate that there has been a change in the *work scope*, and the *customer* will amend *the contract* to reflect the changes.
    - Frequently an **agreement** can be reached in which the program requirements will be pared to fit the resources the customer can offer.
  - Probably just as frequently you will end up *doing more work* than you are paid for — a fact that underscores our *inability* to estimate **accurately enough** the first time around.

6.1 Project History

- Probably one of the most miserable failures in the data processing business is that of *keeping accurate records* of estimates versus actual expenditures.
- The difficulty in doing this industry wide is that *everyone operates differently* and it's tough to agree on any standard way of keeping records.

    - But there is no reason why a given organization cannot and should not compile such data, perhaps with the help of an automated control system.
- It's extremely important to *keep histories* for your projects and then *use them in* planning succeeding jobs.
    - It's worth your taking the time *to define* what records should be kept as part of a history,
    - It's also worth designating someone sharp as a *historian*.
- The fig.6.1.a. presents an **outline** for a **Project History.**

PROJECT HISTORY (TEMPLATE)

DEPARTMENT:

PROPJECT:

DOCUMENT NUMBER:

APPROVALS:

DATE OF ISSUE:

REALISED:

SECTION 1: **SCOPE**

A general statement telling the reader the intention of this document. It is a historical record of important events and data on the project for use in planning and estimating later phases of the project or entirely new projects. It is intended to be a crisp summary, not a huge collection of paper.

SECTION 2: **SIGNIFICANT EVENTS**

A chronological listing and very brief summary of important events during the life of a contract, including missed milestones, new estimates, contract changes, project reviews, equipment installation dates, important telephone agreements, meeting with the customer, and meeting with subcontractors, team member, or vendors.

SECTION 3: **MAJOR SUCCESES**

Point out major deadlines met, profitability, customer satisfaction, whatever was happy about the outcome.

SECTION 4: **MAJOR PROBLEMS**

Describe the major problems: missed schedules, budget overruns, technical performance, personnel, customer relations, management support, etc.

SECTION 5: **MANPOWER HISTORY**

Charts showing three major items:

1. *Total estimated manpower (in man-months) at the beginning of the contract in each of the categories listed in Figure 6.a.*
2. *A record of changes in the estimated numbers shown in (1) and notations explaining the reasons for the changes.*
3. *Total manpower actually used during the contract in each category listed in (1).*

Present all in a form of guidelines for estimating the next job.

SECTION 6: **MACHINE TIME HISTORY**

A series of charts similar to the manpower history. Keep one chart for each type of machine used. Each chart shows:

1. *Total hours estimated for this machine at the beginning of the contract in these categories:*

   - module/integration test

   - system test

   - acceptance test

   - other.
2. *A record of changes in the estimates shown in (1) and notations explaining the reasons for the changes, including differences between expected and actual configurations.*
3. *Total machine time actually used during the contract in each category listed in (1).*

A postmortem comparison between (1) and (3), taking (2) into account, should assist in future estimating.

SECTION 7: **CONCLUSIONS**

The main conclusions. What you would do differently if you were to repeat this job.

Fig.6.1.a. Project History template

- Use this model, or devise one of your own **Project History**.
    - But avoid simply *gathering* a lot of charts and documents, slapping a cover on them, and labeling the whole mess “Project History.”
    - Such *piles of paper* are useless.
- In order for a set of project history data to be helpful on your next job, you must be able to *relate* the two jobs:
    - (1) First, adopt a *development cycle* and stick to it from one project to the next.
    - (2) Then, when you list manpower or computer time costs for items such as “module test,” “integration test,” and “system test,” *those terms will mean* ***the same*** *thing* on the new job you’re estimating *as they did* on the previous jobs for which you have kept histories.

7. The Project Plan

7.1 Characteristics of a good plan

- *A plan is a* roadmap showing how to get from here to there.
    - Like a roadmap, it indicates alternate routes, landmarks, and distances.
- Here are some of the things a good plan is and is not:
- **(1)** It’s *in writing, not in the manager’s head.*
- **(2)** It describes:
    - What the job is.
    - How it will be attacked.
    - The resources required to do it.
- **(3)** It’s written with care so that it’s readable, not just an accumulation of papers whose relation to one another is obscure.
    - Pay a particular attention to *plan continuity*.
- **(4)** It allows for *contingencies: it states* actions to be taken in case something does not go as planned.
    - There are *two types* of ***contingency planning*** to consider:
    - (a) The first involves *specific, identifiable problems* which may arise.

        - For ***example***, if a planned computer is not available on the date specified, where will test time be obtained, and at what added cost in time and money?
    - (b) The second is tougher: what actions will be taken in case ***unforeseen*** *problems* develop?
        - Obviously, specific answers cannot be supplied because specific problems cannot be stated.
        - But the manager can prepare for such events by *being realistic* in *planning resources*.
        - Always *assume people* will get sick, leave the company, get pregnant, misunderstand a specification, make mistakes.
        - Always *assume machines* will break down.
        - Always *assume misunderstandings* with the customer.
        - Plan the *best* you can to make everything work perfectly, but understand that will never happen.
        - Allow *extra resources* to cover the unplanned obstacles.
- **(5)** It's *modular.*
    - *If books were not written in parts, chapters, and paragraphs, reading them would be exhausting.*
        - The same applies to any piece of writing, including a plan.
    - *The document must be logically subdivided* so that there is a *reasonable flow* from one section to the next, but so that each section still *retains its identity* and *is useful* on a stand-alone basis.
- **(6)** It's *brief enough that it won't turn people off.* Mounds of paper simply will not be read*.*
    - *The plan could be as small* as one page for each section.
        - More likely, it will require, for a medium size project, a notebook of perhaps forty or fifty pages. That's not too much to ask *each project member* to read.
    - After having read the plan once, most people will need only refer to *specific sections* of the plan from time to time.
        - How big your plan grows depends in part on how much *tutorial filler* you include.
    - You need enough discussion *to define terms* and *to guide* the reader through the plan, but remember that too much bulk in your plan will cause it to go unread and unused.
- **(7)** It has an *index.*

    - *It only takes a short time to construct a decent **alphabetical index***, and its inclusion will greatly enhance the usefulness of the document.
- In order to fullfill all the above requirements, it is *recommended* to begin with a ***model plan*** and you'll have a *good start* toward continuity.

7.2 Writing the Project Plan

- (1) Normally *planning* does not begin **until** you have a *signed contract* and your *project is under way*.
    - In practice, however, this is rarely the case.
    - Your project's Definition Phase will usually have been preceded by any number of *planning activities*:
        - Your own studies and proposal efforts;
        - Customer-funded studies;
        - Prior related contracts;
        - Discussions between you and your prospective customer.
    - *The planning activity* may be considered *only one step* in an evolutionary process, and this evolution does not end when Definition Phase ends.
    - No matter how good a plan you devise, it will change throughout the life of your project.
        - Even so, the plan you produce during the Definition Phase should be as complete and self-contained as you can make it *at that time.*
- (2) The *people* who contribute to your Project Plan should include:
    - Analysts, programmers, and managers, all of whom are to have continuing *responsibility* on the project.
    - The *user*, too, must either *participate* or at least provide *input.*
        - Leave out any one of them and the plan will probably be unreal.
    - One **individual** arbitrates and has the final say in areas of dispute: that's you the **Project Manager**.
    - If you go off politicking and tell your planning group to conceive up a plan for you to rubber-stamp, you're wasting everybody's time.
    - If you don't really believe in planning and don't intend to use the plan to help guide you during the project, admit it. You're a dynamic manager and you're going to "wing it." Good luck.
- (3) The planning team should be *as small as possible* in order to reduce *interactions* and produce a plan that hangs together.

    - The plan should not read as if it were written by a dozen different people who never talk to one another.
- (4) To get on with the planning job, first decide on a *rough* ***outline***.
    - We suggest that you simply lift the ***model plan outline*** presented in this course and then spend a couple of hours *adapting it* to suit your own ideas.
    - Then, decide how much time you can afford to spend in the *planning activity*.
- (5) Consider all the *individuals* you have available to do the planning — their talents, strengths, weaknesses — and assign them *appropriate pieces of the plan* along with a *schedule* of dates when you expect to see completed drafts.
    - The *model plan* presented in this course is divided into *sections*, or subplans, which provide a handy basis for doling out the work.
- (6) When you're ready to hand out planning assignments, have a *kickoff meeting* including everyone working for or with you on this project.
    - (a) With everyone (both planners and analysts) together in the same room at the same time, define and discuss the ***project's objectives***.
    - (b) Then discuss any ***constraints*** under which the project must operate, such as *fixed deadlines, customer-imposed milestones, funding,* and *work location*.
    - (c) Establish and analyze the ***risks*** related to the project.
    - (d) Having set the stage, now assign sections of the Project Plan to *individuals*.
    - (e) If you have already done this before the meeting, be sure now that everyone knows not only his assignment, but the assignments of the other project members.
    - (f) Write down *names, assignments*, and *dates* when material is due to you, and give everyone present *a copy* of the list.
- (7) Follow up the *kickoff meeting* with periodic status meetings in which planners and analysts) can describe their *progress* and in which you can *assess* overall status relative to approaching target dates.
- (8) The Definition Phase and the Design Phase together should be allotted from about *one-fourth* to *one-half* of the total contract time.
    - **Planning** is a *major activity* during both those phases. You will find it all too easy to slight your planning activity because you're anxious to get on with the programming.
    - But if you don't take the time to plan well, you'll more than pay for it *in later phases* when things fall apart:

        - Testing will falter, schedules will be missed, budgets will be overrun, and quite likely the quality of your product will suffer.
    - Maybe *you* don't mind the chaos of a poorly planned project, but how about your people?
        - If they are deprived of a pleasant, broadening work experience, or if their private lives are severely impacted because of poor project planning, the *management of the job* is a failure.

7.3 A Project Plan Outline

- **Metzger** tried many *different structures* before settling on **the outline** presented in this course.
    - Use it if you'll find it helpful, but by all means modify it to suit *your needs*, even your temperament. After all, you have to live with it.
- **The Project Plan** is divided into the *following sections*, which are summarized in succeeding paragraphs:
    - (1) Overview
    - (2) Phase Plan
    - (3) Organization Plan
    - (4) Test Plan
    - (5) Change Control Plan
    - (6) Documentation Plan
    - (7) Training Plan
    - (8) Review and Reporting Plan
    - (9) Installation and Operation Plan
    - (10) Resources and Deliverables Plan
    - (11) Index

7.3.1 Overview

- The **overview** section of the plan has some important *purposes*:
    - (1) First, it assumes that the reader *knows nothing* about the project and it introduces him to the job and to the customer.
    - (2) Second, it describes the *general organization* of the plan.
    - (3) Presents the *assumptions and restrictions* on which the plan is based.

- (4) Establishes a *gross schedule* for the project.
- (5) Refers for short to *technical aspects*.
- (6) Makes a *risk analyze* of the project.
- (7) It *summarizes* the entire plan by giving a *capsule description* of the detailed plan elements that follow the Overview. (fig.7.3.1.a).

SECTION 1

PLAN OVERVIEW (TEMPLATE)

DEPARTMENT:

PROJECT:

DOCUMENT NUMBER:

APPROVALS:

DATE OF ISSUE:

REALISED:

1.1 OBJECTIVE

The objective of this section is to summarise the entire Project plan .

1.2 DISCUSSION

- ***Set the stage. Identify the customer and his experience in the field. Describe in one or two paragraphs the job to be done***.
  - *Why is this project being done*
  - *What are the project deliverable?*
  - *Give any background necessary for a good understanding of the job environment. Then state your project's objectives under contract.*
- ***Next, explain how the plan is organized**. Promise the reader that he can gain a good understanding of the plan by reading each section's Objectives and Discussion, and the keep your promise.*
  - *Will a team approach be used or one individual assignment?*
  - *What will be the responsibilities for the team or individual ?*

    - *Who will the team or individual report to?*
    - *How and where will be subcontractors be used?*
    - *What hardware and software is required?*
    - *What other products are involved?*
    - *What other documents should be referenced?*
- ***Now list the assumptions and restrictions on which the plan is based**. They are very important. Don't hide them with a lot of words. State them honestly and simple.*
- ***Then establish a gross schedule for the project**. This schedule should show al major efforts bearing on this project, whether under your control or not.*
    - *What tasks will be involved?*
    - *Who will be assigned to each task?*
    - *When will each task start?*
    - *How long will each task take?*
    - *What order do the tasks have to be done in?*
    - *How will the schedule be kept up-to-date?*
- ***You had also to refer for short to the technical aspects:***
    - *What programming language will be used?*
    - *What platform will be used?*
    - *What development methods , tools, techniques, rules, practices and conventions will be followed?*
    - *Who will perform the code reviews?*
    - *How will code reviews be performed?*
    - *What are documentation requirements?*
- ***Finally make a risk analyze of the project.***
    - *What are the risks associated with the project. Detail where is necessary:*
        - *Unrealistic schedules and budgets*
        - *Over-engineering*
        - *Subcontractors not having the degree of quality assurance necessary*

    - *Shortfalls in externally performed tasks includes software and hardware tasks carried out by a subcontractor and shortfalls in tasks carried out by the customer*
    - *Staff deficiencies*
    - *Are personnel technically equipped to perform the tasks assigned to them?*
    - *Requirements volatility - Will requirements for the system change during the course of the project?*
    - *Performance problems / Scaling problems - Is the real-time response sufficient for projects that involve access to large databases? Strains on current computer technology - Does the project use unproven technology*
  - *What is the degree of the risk?*
  - *What actions will be taken to mitigate risk?*
  - *What skills do the developers need to have?*
  - *How will project progress be monitored?*

1.3 DETAIL

Give a brief description of the objectives of each of the remaining sections of the Project Plan.

Fig.7.3.1.a. Plan Overview template

7.3.2 Phase Plan

- The objective of this section is to define the *development cycle* for the project.
- The Phase Plan serves as a *foundation* for subsequent plan elements.
  - It is highly recommended to adopt a development cycle as the one presented in this course or another.
- For each phase of the **Life cycle** describe the **primary** and the **secondary objectives**.
  - The Phase Plan should end with a chart similar to fig. 7.3.2.a but with *dates included.*

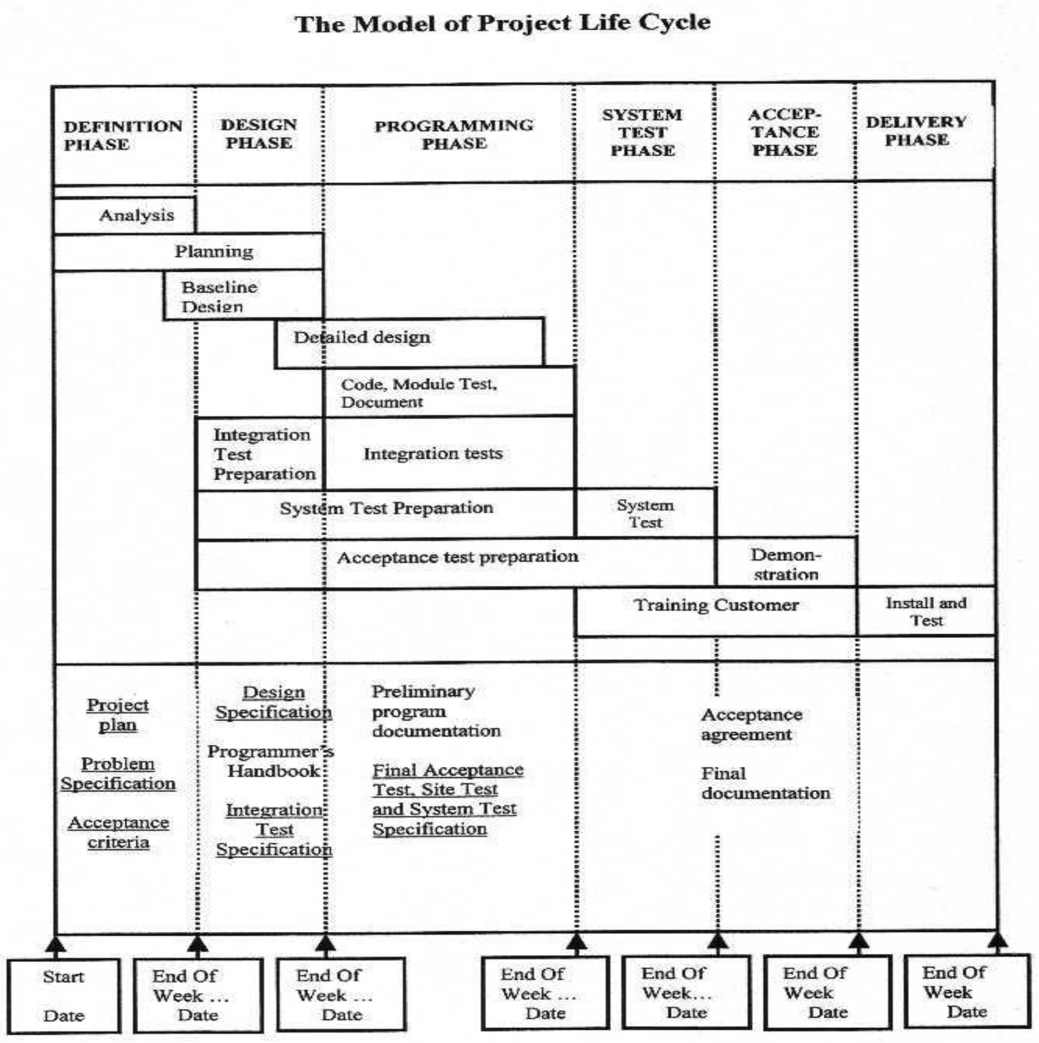


Fig. 7.3.2.a. The Chart Model of the Project Life Cycle

- The outline of the Phase Plan is presented in fig. 7.3.2.b.
- The Phase Plan provides you with a *base*, a *point of reference*.
    - For example, when you and your people talk about the System Test Phase, *you should all be talking about the* same thing.

- Unfortunately this *rarely happens* on a project, and that's a crime. It leads to much confusion and misunderstanding that could easily be avoided.

SECTION 2

PHASE PLAN (TEMPLATE)

DEPARTMENT:

PROJECT:

DOCUMENT NUMBER:

APPROVALS:

DATE OF ISSUE:

REALISED:

2.1 OBJECTIVE

The objective of this section is to define the programming development effort in terms of a series of time-slices called "phase".

2.2 DISCUSSION

Define your development cycle and, briefly, each phase making up the cycle. Include an illustration such in fig. 7.3.2.a. but include calendar dates. Establish basic definitions and points out that the remaining sections of the plan are tied to these definitions. If you are planning multiple releases, show how they are scheduled in a series of overlapping, essentially identical, development cycles.

2.3 DETAIL

For each phase list primary and secondary objectives and define each objective as rigorously as possible

2.3.1 **Definition Phase**

2.3.1.1 Primary Objectives

(a) Problem analysis

(b) Detailed project planning

(c) Defining acceptance criteria

2.3.1.2 Secondary Objectives

(a) Finding people

(b) Understanding the customer

(c) Forming tentative design ideas

2.3.2 Design Phase

2.3.2.1 Primary Objectives

(a) Baseline design for operational programs

(b) Baseline design for support programs

2.3.2.2 Secondary Objectives

(a) Preparation for integration testing

(b) Setting up change controls

(c) Constructing simulation models

(d) Planning for subsequent phases

(e) Preparation for programmer training

(f) Publication of Programmer's Handbook

(g) Initial preparation for system test

(h) Initial preparation for acceptance test

(i) Initial preparation for site test

(j) Setting up project libraries

2.3.3 Programming Phase

2.3.3.1 Primary Objectives

(a) Detailed design

(b) Coding

(c) Module test

(d) Integration test

(e) Program documentation

2.3.3.2 Secondary Objectives

(a) Detailed preparation for system test

(b) Detailed preparation for acceptance test

(c) Detailed preparation for site test

(d) Preparation for customer training

2.3.4 **System Test Phase**

2.3.4.1 Primary Objectives

(a) Testing against Problem Specification

(b) Testing as “live” as possible

(c) Testing not controlled by programmers

2.3.4.2 Secondary Objectives

(a) Completion of acceptance test preparations

(b) Customer training

(c) Correction or descriptive documentation

(d) Completion of user documentation

(e) Reassignment of people

2.3.5 **Acceptance Phase**

2.3.5.1 Primary Objectives

(a) Execution and analysis of acceptance tests

(b) Signing of formal acceptance agreement

2.3.5.2 Secondary Objectives

(a) Completion of customer training

(b) Cleanup of documentation

2.3.6 **Installation and Operation Phase**

2.3.6.1 Primary Objectives

(a) Assistance in installing system

(b) Assistance in beginning operation

2.3.6.2 Secondary Objectives

(a) Testing on-site

(b) Continuing maintenance and tuning

(c) Continuing operation

(d) Project evaluation

2.4 TASKS CHART

Introduce here a tasks chart for the project (as can be produced by MS Project for example).

Fig. 7.3.2.b. The Phase Plan template

7.3.3 Organization Plan

- Organization Plan element should define:
  - (1) The ***organization*** during the various *phases* of the project.
  - (2) The ***specific responsibilities*** of *each group* within the organization.
- The outline of the Phase Plan is presented in fig. 7.3.3.a.
  - (1) It presents first the groups and their general responsibilities.
  - (2) For **each phase** of the **development cycle**, the specific organization and groups' responsibilities are detailed.

SECTION 3

ORGANIZATION PLAN (TEMPLATE)

DEPARTMENT:

PROJECT:

DOCUMENT NUMBER:

APPROVALS:

DATE OF ISSUE:

REALISED:

3.1 OBJECTIVE

The objective of this section is to define the organisation of the project and the assignment of responsibilities.

3.2 DISCUSSION

Review the basic reasons for establishing an organisation: clarity of job assignment, minimising interactions, controlling change, establishing points of responsibility and focus. Sketch the main flow of work within the organisation, starting with problem analysis and design and running through programming, testing, documentation and delivery.

3.3 DETAIL

In the first subsection list the groups which will be found on the organisation charts and the general responsibilities of each group. Then show an organisation chart for each phase. The organisation will generally not be the same for all phases; for example, during the Definition Phase there will not yet exist a Programming Group.

3.3.1 Groups and General Responsibilities

3.3.1.1 Analysis and Design Group

(a) Writing Problem Specification

(b) Writing Design Specification

(c) Change control

(d) Data control

(e) Simulation modelling

(f) Design and code inspections

(g) Writing user documentation

3.3.1.2 Programming Group

(a) Detailed design

(b) Coding

(c) Module test

(d) Integration test

(e) Descriptive Documentation

3.3.1.3 Test Group

(a) Writing System Test Specifications

(b) Writing Acceptance and Site Test Specifications

(c) Validating test cases

(d) Gathering and generating test data

(e) Choosing and obtaining test tools

(f) Setting up test libraries

(g) Scheduling test resources

(h) Executing tests

(i) Analysing test results

(j) Documenting test results

3.3.1.4 Staff Group

(a) Library services

(b) Computer time control

(c) Supplying typing services

(d) Planning and installing terminals

(e) Issuing Programmer's Handbook

(f) Training

(g) Special technical assignments

(h) Technical liaison

(i) Document control

(j) Report control

(k) Contract change control

(l) Supplying clerical support

(m) Maintaining project history

3.3.2 Organisation and Responsibilities: Definition Phase

3.3.3 Organisation and Responsibilities: Design Phase

3.3.4 Organisation and Responsibilities: Programming Phase

3.3.5 Organisation and Responsibilities: System Test Phase

3.3.6 **Organisation and Responsibilities:**

Acceptance Phase

3.3.7 **Organisation and Responsibilities:**

Installation and Operation Phase

Fig. 7.3.3.a. Organization Plan template

- **Organization Plan** should be *reorganized* from time to time for the following reasons:
    - (1) As the project moves along from one phase to another, the *emphasis* shifts from **analysis** to **design** to **programming** and then to **system testing**.
        - The organization should shift with the work.
        - For **example**, there is no need for an installation group as early as the Definition Phase, and there may be no need for a requirements analysis group during the Installation and Operation Phase.
    - (2) You should organize around the *people* you have.
        - If a new manager begins working for you midway through the project, and if he has strong feelings about how his end of the project should be organized, listen to him and try to organize to suit him.
        - Nothing wrong with that, if it enhances his effectiveness and does not foul up someone else's.
    - (3) You use incremental and/or iterative development technologies.
        - In this case you may have a specific **Organization Plan** for **each iteration**.
- If the organization you planned for and adopted just isn't working smoothly, then, of course, change it.

7.3.4 Test Plan

- This section describes the *tools*, *procedures*, and *responsibilities* for conducting all testing levels on the project. (Fig. 7.3.4.a.)

- The Test Plan should clearly define:
  - (1)The levels of test (for example, *"module test," "integration test," "system test," "acceptance test," "site test)*.
  - (2) Responsibility for executing each level.
  - (3) Machine support required for each level.
  - (4) Support programs or tools required.
  - (5) The reporting of test results.
- For each **test level** the Test Plan must define:
  - (1) The test objectives.
  - (2) The test responsibility.
  - (3) The test procedures.
  - (4) The test entry criteria.
  - (5) The test exit criteria.
  - (6) The test tools.

SECTION 4
TEST PLAN (TEMPLATE)

DEPARTMENT:

PROJECT:

DOCUMENT NUMBER:

APPROVALS:

DATE OF ISSUE:

REALISED:

4.1 OBJECTIVE

The objective of this section is to define the tools, procedures, and responsibilities for conducting all levels of test of the program system.

4.2 DISCUSSION

A convenient way to write a test plan is to define each discrete level of test (for example, module test, integration test, system test, acceptance test, site test) and then describe the objectives, procedures, responsibilities, and tools for each level. In this subsection, briefly define each test level and show how the different levels fit together in a meaningful test hierarchy. Emphasise the need for modularity of the testing process and the need for certainty at one level before proceeding to the next.

4.3 DETAIL

4.3.1 Module Test

Module test is testing done on the lowest-level program modules before they are integrated with other modules.

4.3.1.1 Module Test Objectives

4.3.1.2 Module Test Responsibility

4.3.1.3 Module Test Procedures

4.3.1.4 Module Test Entry Criteria

4.3.1.5 Module Test Exit Criteria

4.3.1.6 Module Test Tools

4.3.2 Integration Test

Integration test is the process of combining tested modules into progressively more complex grouping, either top-down or bottom-up, and testing these groupings until the entire program system has been put together and tested.

4.3.2.1 Integration Test Objectives

4.3.2.2 Integration Test Responsibility

4.3.2.3 Integration Test Procedures

4.3.2.4 Integration Test Entry Criteria

4.3.2.5 Integration Test Exit Criteria

4.3.2.6 Integration Test Tools

4.3.3 System Test

System Test is the re-testing of the completed program system in as nearly a live environment as possible by personnel other than those who produced the programs.

4.3.3.1 System Test Objectives

4.3.3.2 System Test Responsibility

4.3.3.3 System Test Procedures

4.3.3.4 System Test Entry Criteria

4.3.3.5 System Test Exit Criteria

4.3.3.6 System Test Tools

4.3.4 **Acceptance Test**

Acceptance test is the exercising of the program system under conditions agreed to by the customer in order to demonstrate that the system satisfies the customer's requirements.

4.3.4.1 Acceptance Test Objectives

4.3.4.2 Acceptance Test Responsibility

4.3.4.3 Acceptance Test Procedures

4.3.4.4 Acceptance Test Entry Criteria

4.3.4.5 Acceptance Test Exit Criteria

4.3.4.6 Acceptance Test Tools

4.3.5 **Site Test**

Site test is testing of the program system in its ultimate operating environment to assure readiness for operation.

4.3.5.1 Site Test Objectives

4.3.5.2 Site Test Responsibility

4.3.5.3 Site Test Procedures

4.3.5.4 Site Test Entry Criteria

4.3.5.5 Site Test Exit Criteria

4.3.5.6 Site Test Tools

4.3.6 **Common Test Facilities**

Describe the facilities and tools common to several or all levels of test.

4.3.6.1 Development Support Library

4.3.6.2 Computer Facilities

4.3.6.3 Typing Services

4.3.6.4 Terminal Systems

4.3.6.5 Special Languages

4.3.6.6 Test Run Pickup and Drop Areas

4.3.7 **Testing Support Programs**

Describe anything unique about the testing of the test tools themselves.

Fig. 7.3.4.a. Test Plan template

7.3.5 Change Control Plan.

- *Controlling changes* in the developing program system is one of management's most vital functions.
- This section defines:
  - (1) The *kinds of changes* to be controlled.
  - (2) The *mechanism* for effecting that control. (fig. 7.3.5.a.).

SECTION 5

CHANGE CONTROL PLAN (TEMPLATE)

DEPARTMENT:

PROJECT:

DOCUMENT NUMBER:

APPROVALS:

DATE OF ISSUE:

REALISED:

5.1 OBJECTIVE

The objective of this section is to define the procedures to be used for controlling change in the program system.

5.2 DISCUSSION

Describe the customer's need to know that what is being developed by you is what was envisioned when the contract was signed, and your own need for knowing that what the programmers are producing is what was originally intended. A solution to this problem is to establish certain critical baseline documentation acceptable to both customer and you, and to control events always relative to those baselines. Whenever a question is raised, the baseline documents are reference point. Anything anyone wants that is not covered in the baselines is a change, and it must be negotiated. When a change is deemed necessary ,its cost and impact, if any, must be asserted, and the change must be written into the baseline document(s). A revised baseline document becomes the new baseline.

5.3 DETAIL

5.3.1 **Baseline**

Define the documents to be used as baselines on your project.

5.3.1.1. Problem Specification

5.3.1.2. Design Specification

5.3.2 **Proposing a Change**

5.3.2.1 Who May Propose a Change

(a) Project members

(b) Customer

(c) Other contractors

5.3.2.2 Change Proposal Document

5.3.3 **Investigating a Proposed Change**

5.3.3.1 Who, How, When?

5.3.3.2 The Investigator's Report

(a) Summary of proposed change

(b) Originator's name and organisation

(c) Classification of the change

(d) Impact on costs, schedules, other programs

(e) Recommendations

5.3.4 **Types of Changes**

5.3.4.1 Type 1

The change affects a baseline or would cause a cost, schedule, or other impact.

5.3.4.2 Type 2

The change affects no baseline and has negligible cost, schedule, or other impact.

5.3.5 **Change Control Board**

5.3.5.1 Membership

5.3.5.2 When It Meets

5.3.5.3 How It Operates

5.3.6 **Types of Recommendations**

5.3.6.1 Acceptance

5.3.6.2 Rejection

5.3.7 **Implementing a Change**

5.3.7.1 Estimating Cost of Change

5.3.7.2 Approvals

(a) Project management

(b) Customer

5.3.7.3 Documenting the Change

5.3.7.4 Testing the Change

OBS. *As result of the Change Control Plan, an associated Change Procedure must be defined as a component of the Software Quality Assurance (SQA).*

Fig. 7.3.5.a. Change Control Plan template

- In fact Change Control Plan defines:
    - (1)The baseline documents (the documents affected by changes).

- (2) The management procedure of the changes consisting in:
    - (2.1) Proposing a change.
    - (2.2) Investigating a proposed change.
    - (2.3) Types of changes.
    - (2.4) Change Control Board.
    - (2.5) Types of recommendations.
    - (2.6) Implementing a change.
- When you write a change control procedure, it's always a temptation to try to cover every conceivable kind of change no matter how minor.
    - Therein lies failure because these procedures quickly become so entangled in details that they become an administrative horror and collapse.

7.3.6 Documentation Plan.

- This is a key section, but it's usually missing.
- Its intent is to control the gush of paper that inevitably accompanies most projects.
    - One important cause of our so often getting buried under paper is that we don't take the time to define the documents we want to use on the project.
    - As a result, whenever a project member needs to write something, he dreams up his own format and suddenly there is a new kind of document to file and keep track of.
- The Documentation Plan is a gathering place in the Project Plan for the descriptions of *all paper work* to be used during the project. (fig.7.3.6.a).

SECTION 6

DOCUMENTATION PLAN (TEMPLATE)

DEPARTMENT:

PROJECT:

DOCUMENT NUMBER:

APPROVALS:

DAT OF ISSUE:

REALISED:

6.1 OBJECTIVE

The objective of this section is to define:

- *the name and type of the project documents*
- *project documents organisational structure*
- *publication procedure*

6.2 DISCUSSION

6.2.1 Generalities

The recommended standard documents for a project development are presented in Documentation Summary fig. 7.3.6.b. In dependence of your project (size, importance, price), only some of the recommended documents should be used. Emphasise that all project documents will be outlined in this section and no new kinds of documents are to be written unless management cam be shown why the currency planned documents are inadequate; in that case, a new document will be outlined and added to the plan

6.2.2 Name and type of the project documents

For each project document describe the name, the contents, preparation procedure (who writes, date of issue), approval procedure (person), deadline, physical location of document. In the same time, for each documents must be specified the access rights (R, RW, password) for different categories of people. Include a chart such as that shown on Documentation Summary and complete with details peculiar to your job.

6.2.2. Number of a document

Each document receives a ***unique number*** *which is registered in* Documentation Index*.* *This number appears in all places where the document is out lined.*

6.3. DETAILS

6.3.1 Project Documents Infrastructure (PDI)

This section describes the philosophy of the Project Documents Infrastructure ***(PDI)*** *for your project. The* ***PDI*** *consists in a number of* Project Repositories *containing documents, tables of documents and other information associate to the project, structured in an accessible manner.* ***PDI*** *is in fact a structured data collection containing all the information related to the project (documents, code, reports, etc.). The main purpose of this section is to describe this structure. The* ***PDI*** *is usually distributed in the local network. The recommended structure of the* ***PDI*** *consists in the following repositories.*

- *Documentation Index*
- *Project Plan Repository*
- *Project Specification Repository*
- *Project Code Repository*
- *Project Test Repository*
- *Project Bug Repository*
- *Project Change Repository*
- *Documentation Repository*
- *Manager Repository*

In dependence of your project (size, importance, price), the repositories configuration can be restructured. Include here a ***PDI Summary*** *describing your project repositories and their physical location as shown below.*

Nr.crt	Repository	Physical location	Obs
1	Documentation Index		
2	Project Plan Repository	C:\Prj1\….	
3	Project Specification Repository	D:\Users\…	
4	Project Code Repository		
5	Project Test Repository		
6	Project Bug Repository		
7	Project Change Repository		
8	Documentation Repository		

9	Manager Repository		

6.3.2 Repositories Description

6.3.2.1 Documentation Index

A file containing a list of all project's documents. It's the primary evidence of any project document. For any project documents, apart from the repository in which it is physically placed, there is an entrance in this table. Each document has a ***unique identification number*** *and a* ***name****. (See* Documentation Index Template).

6.3.2.2 Project Plan Repository

6.3.2.2.1 Repository Content

A description of the repository structure and a list of included files (See Project Plan Repository*). Include a table of repository contents as* Project Plan Repository Table.

6.3.2.2.2 Access rights

For each directory and file included in the repository specify the access right for different categories of people.

6.3.2.3 Project Specification Repository

6.3.2.3.1 Repository Content

A description of the repository structure and a list of included files (See Project Specification Repository*). Include a table of repository contents as* Project Specification Repository Table .

6.3.2.3.2 Access rights

For each directory and file included in the repository specify the access right for different categories of people.

6.3.2.4 Project Code Repository

6.3.2.4.1 Repository Content

A description of the repository structure and a list of included files (See Project Code Repository*). Include a table of repository contents as* Code Repository Table *. Project Code Repository can be implemented using specific tools (CVS).*

6.3.2.4.2 Access rights

For each directory and file included in the repository specify the access right for different categories of people.

6.3.2.5 Project Test Repository

6.3.2.5.1 Repository Content

A description of the repository structure and a list of included Repositories (See Project Test Repository structure). Include a table of repository contents as Project Test Repository Table.

6.3.2.5.2 Access Rights

For each specific repository included in the Test Repository specify the access right for different categories of people.

6.3.2.5.3 Integration Test Repository

6.3.2.5.3.1 Repository Content

A description of the repository structure and a list of included files (See Integration Test Repository structure). Include a table of repository contents as Integration Test Repository Table.

6.3.2.5.3.2 Access Rights

For each specific file included in the Repository specify the access right for different categories of people.

6.3.2.5.4 ***System Test Repository***

6.3.2.5.4.1 Repository Content

A description of the repository structure and a list of included files (See System Test Repository structure). Include a table of repository contents as System Test Repository Table.

6.3.2.5.4.2 Access Rights

For each specific file included in the Repository specify the access right for different categories of people.

6.3.2.5.5 ***Acceptance Test Repository***

6.3.2.5.5.1 Repository Content

A description of the repository structure and a list of included files (See Acceptance Test Repository structure). Include a table of repository contents as Acceptance Test Repository Table.

6.3.2.5.5.2 Access Rights

For each specific file included in the Repository specify the access right for different categories of people.

6.3.2.5.6 ***Site Test Repository***

6.3.2.5.6.1 Repository Content

A description of the repository structure and a list of included files (See Site Test Repository *structure). Include a table of repository contents as* Site Test Repository Table.

6.3.2.5.6.2 Access Rights

For each specific file included in the Repository specify the access right for different categories of people.

6.3.2.6 Project Bug Repository

6.3.2.6.1 Repository Content

A description of the repository structure and a list of included files (See Project Bug Repository *structure). Include a table of repository contents as* Project Bug Table .

6.3.2.6.2 Access rights

For each directory and file included in the repository specify the access right for different categories of people.

6.3.2.7 Project Change Repository

6.3.2.7.1 Repository Content

A description of the repository structure and a list of included files (See Project Change Repository *structure). Include a table of repository contents as* Project Change Table.

6.3.2.7.2 Access rights

For each directory and file included in the repository specify the access right for different categories of people.

6.3.2.8 SW Product Documentation Repository

6.3.2.8.1 Repository Content

A description of the repository structure and a list of included files (See SW Product Documentation Repository *structure). Include a table of repository contents as* SW Product Documentation Repository Table.

6.3.2.8.2 Access rights

For each directory and file included in the repository specify the access right for different categories of people.

6.3.2.9 Manager Repository

6.3.2.9.1 Repository Content

A description of the repository structure and a list of included files (See Manager Repository *structure). Include a table of repository contents as* Manager Repository Table.

6.3.2.9.2 Access rights

For each directory and file included in the repository specify the access right for different categories of people.

6.3.3 Publication Procedures

6.3.1.1 Preparation and Approval

6.3.1.2 Proofing and Editing

6.3.1.3 Visibility

(a) General

(b) Restricted

6.3.1.4 Reproduction

(a) Ordinary

(b) Specially

6.3.1.5 Distribution

(a) Within the project

(b) To the customers

(c) Other contractors

(d) Company management

(e) Vital records storage

Fig. 7.3.6.a. Documentation Plan template

- When someone wants to write something down, he should be able to find an ***appropriate** kind of document* clearly outlined in the plan.
    - If you miss a few in your initial planning, don't fret. Add new document descriptions whenever you find that they are needed.

- Keep your document descriptions as uncluttered and as flexible as possible so that writers will have freedom to express themselves.
    - Since pride of authorship is a very powerful motivating force in most people, whether they admit it or not, documentation guidelines that are too restrictive will be ignored.
- In addition to serving as an index of document descriptions the Documentation Plan includes a summary of publication procedures dealing with preparation, approval, reproduction, distribution, and filing.

Name of Document	Contents	Preparation			Approval		Repository Location
		Who writes	When finished	Formal proof	Who approves	Dead line	
PROJECT PLAN	The plans associated to the project	Project Manager	End of Definition Phase	Yes			Project Plan Repository
PROBLEM SPECIFICATION	A description of the requirements of the problem to be fulfilled by the product	Analysts	End of Definition Phase	Yes	Customer		Project Specification Repository
DESIGN SPECIFICATION	A description of the design to be used by the programmers in producing the product	Analysis and Design Group	End of Design Phase	Yes	Customer		Project Specification Repository
CODING SPECIFICATION	A detailed description of the modules produced by the programmers; the specifications describe the complete product	Individual programmers	Preliminary version at end of module test Final version at end of acceptance test	Yes	Customer		Project Specification Repository

CHANGE PROPOSAL	A description of proposed change to the Problem Specification and/or Design Specification	Anyone	Anytime	No			Project Change Repository
CHANGE RECOMMA-DATION	A description of the proposed change, classification of the change, the impact, adoption recommendation	Change Investigator	After a Change Proposal	Yes	Change Control Board		Project Change Repository
PROBLEM SPECIFICATION CHANGE NOTICE	A description of adopted changes to the current Problem Specification	Contractor, Analysis and Design Group	Anytime	No	Contractor Customer		Project Specification Repository
DESIGN SPECIFICATION CHANGE NOTICE	A description of adopted changes to the current Design Specification	Contractor, Analysis and Design Group	Anytime	No	Depends on change		Project Specification Repository
INTEGRATION TEST SPECIFICATION	A description of the philosophy, objectives and procedures involved in integration test; a matrix of test cases	Programmers	End of Design Phase	No	Contractor		Project Specification Repository
SYSTEM TEST SPECIFICATION	A description of the philosophy, objectives and procedures involved in system test; a matrix of test cases	Test Group	End of Programming Phase	Yes	Contractor		Project Specification Repository

ACCEPTANCE TEST SPECIFICATION	A description of the philosophy, objectives and procedures involved in acceptance test; acceptance criteria; a matrix of test cases	Test Group	Preliminary version at end of Definition Phase; Final version at end of Programming Phase	Yes	Customer		Project Specification Repository
SITE TEST SPECIFICATION	A description of the philosophy, objectives and procedures involved in site test; a matrix of test cases	Test Group	End of Programming Phase	Yes	Customer		Project Specification Repository
TEST CASE	Individual test script and data	Test Group and Programmers	Depends on type of test	No	Depends on type of test		Specific Test Repository
TEST REPORT	A report of any problem encountered during any formal test; integration, system, acceptance	Test conductor	After running any test cases	No			Specific Test Repository
TEST SCRIPT	A description of the steps to be followed in test. (Part of the Test Case)	Test Group/ Programmers	Depends on type of test	No	Depends on the type of test		Specific Test Repository
TEST CHECKLIST	A description of the items to be verified. (Part of the Test Case)	Test Group (Programmers)	Depends on type o test	No	Depends on type of test		Specific Test Repository
BUG DESCRIPTION	A bug description for each of the problem encountered during the any test. (Attached to the Test Report).	Tester	After running any test cases				Project Bugs Repository

BUG FIXING NOTE	A description of the bug fixing.	Debugging programmer	After debugging				Project Bugs Repository
TECHNICAL NOTE	Miscellaneous technical correspondence	Anyone	Anytime	No	Depends on the content		Manager Repository
ADMINISTRA-TIVE NOTE	Miscellaneous administrative correspondence	Anyone	Anytime	No	Depends on the content		Manager Repository
PROGRAMMER'S HAND BOOK	Collection of data needed by the programmer	Technical Staff	First version at the end of Design Phase	No	Contractor		Code Repository
TECHNICAL STATUS REPORT	A single form used in reporting technical status to the next management level within the contractor's organization	Each level	Biweekly or monthly	No			Manager Repository
PROJECT HISTORY	A set of charts showing • Significant events • Manpower (estimated/actual) • Machine time (estimated/actual)	Administra-tive Staff (Project manager)	At the end of the contract	No	Contractor		Manager Repository
DOCUMENTATION INDEX	A table containing entrances for all current project documents	Administra-tive Staff (Librarian)	Periodically published	No			Project Documents Infrastructure

Fig. 7.3.6.b. Documentation Summary

7.3.7 Training Plan.

- Generally, there are two categories of training required on a project:
    - (1) Internal - training your own people.
    - (2) External - training the customer and others.
- Training is often awarded little or no space in a plan, but this omission can be serious on some jobs.
    - The Training Plan defines all the *kinds* of *internal* and *external training* required, the *responsibility* for each, and the *resources* required (fig. 7.3.7.a.).

SECTION 7

TRAINING PLAN (TEMPLATE)

DEPARTMENT:

PROJECT:

DOCUMENT NUMBER:

APPROVALS:

DATE OF ISSUE:

REALISED:

7.1. OBJECTIVE

The objective of this section is to define the contractor's training responsibilities.

7.2. DISCUSSION

The programming contractor is responsible for two general categories of training: internal (training his internal people) and external (training the customer, the system contractor, and others).

7.3. DETAIL

7.3.1. Types of Training

7.3.1.1. Internal Training

(a) Understanding the overall project

(b) Technical

- Coding language
- Development environment
- Use of test tools
- The data processing
- Interfacing with other systems
- The problem
- The baseline design

(c) Nontechnical

- Management techniques
- Change control procedures
- Documentation control
- Reporting requirements
- Clerical procedures

7.3.1.2. External Training

(a) Installing the program

(b) Using the system

(c) Modifying the system

7.3.2. Resources

For each identified type of training show:

- *Training schedules*
- *Instructors required*
- *Training materials*
- *Facilities (class rooms, computers, etc.)*
- *Number of trainees*
- *Special computer programs for training*

Fig. 7.3.7.a. Training Plane template

7.3.8 Review and Reporting Plan

- The objective of this plan element is to define how project status will be communicated by:
    - Oral project reviews.
    - Written reports.
    - Structured walk-through.
    - Inspections.
- Structure Walk-through's and inspections are discussed in detail later.
    - They are not intended as a means of *reporting status* to management, but they are an important means of helping project members *assess the quality* of the products they are developing.
- The outline of Review and Reporting Plan is presented in fig. 7.3.8.a.

SECTION 8

REVIEW AND REPORTING PLAN (TEMPLATE)

DEPARTMENT:

PROJECT:

DOCUMENT NUMBER:

APPROVALS:

DATE OF ISSUE:

REALISED:

8.1. OBJECTIVE

The objective of this section is to describe the means of reviewing and reporting progress.

8.2. DISCUSSION

There is informal review and reporting going on at all levels more or less continuously. This plan address not the informal, but rather the formal reviewing and reporting. This Discussion subsection should describe in a general way the reporting structure. It should stress the importance of making financial and technical reports consistent with one another. It should describe the contractor's accounting system to which the project financial reports must conform.

8.3. DETAIL

8.3.1. Reviews

8.3.1.1. Internal Reviews

The Internal Reviews activities will be based on the developing model outlined in Review Procedure. Participants in each internal review include project members and outside reviewers.

(a) Definition Phase Review

When: *End of Definition Phase*

Objectives:

- *To review the Problem Specification and to determine readiness for the Design Phase;*
- *To review and assess the Project Plan;*
- *To review the Acceptance Criteria.*

(b) Preliminary Design Phase

When: *Midway in the Design Phase*

Objectives:

- *To review the baseline design, as far as it has been developed, in order to assure the validation of the design approach.*

(c) Design Phase Review

When: *End of Design Phase*

Objectives:

- *To review the completed Design Specification to determine whether or not it satisfies the Problem Specification and is reasonable and programmable;*
- *To review the Project Plan.* ***(Include outside reviewers)***

(See: Design Phase Review Outline).

(d) Programming Phase Review

When: *End of Programming Phase*

Objectives:

- *To review program integration results and determine readiness for the System Test Phase;*
- *To review program documentation*

(e) System Test Phase Review

When: *End of System Test Phase*

Objectives:

- *To review system test results and determine readiness for the Acceptance Phase;*
- *To review program documentation*

(f) Postmortem Review

When: *End of Acceptance Phase*

Objectives:

- *To review and approve the Project History document.*

8.3.1.2. External Reviews

Participants in each of these reviews include representatives of the contractor and the customer.

(a) Preliminary Design Review

When: *Midway in the Design Phase, after internal review.*

Objectives:

- *To review the validity of the design approach.*

(b) Design Phase Review

When: *At the end of the Design Phase, after internal review.*

Objectives:

- *To review in detail the Design Specification;*
- *To review the contractor's Project Plan for the entering the Programming Phase.*

(c) Acceptance Review

When: *End of Acceptance Phase*

Objectives:

- *To review the results of the competed acceptance tests and determine any remaining problems that must be corrected before the customer will formally accept the programs.*
-

8.3.1.3. Structured Walk-Through

These are held whenever there is a product (a design, code, test plan, user's manual, anything) ready for a close look by other project members guided ("walked through") by the developer of the product.

Objectives:

· *To find error not to report status.*

8.3.2. Reports

8.3.2.1. Generated by Nonmanagers

(a) Frequency: bi-weekly.

(b) To: immediate manager.

(c) Format: Technical Status Report.

(d) Scope: one report for each task assigned.

8.3.2.2. Generated by Managers

(a) Frequency: bi-weekly.

(b) To: immediate manager;

(c) Format: Technical Status Report.

(d) Scope: one report for each milestone task.

8.3.2.3. Generated by Project Manager

(a) Frequency: monthly and quarterly. A quarterly report should replace the monthly normal report.

(b) To: company management and customer.

(c) Format: Depends on company and customer requirements. (See Project Manager Report).

8.3.2.4. Generated by Company Staff

Describe the fed back to management by the company. These reports are usually financial. See Staff Report.

Fig. 7.3.8.a. Review and Reporting Plan template

- The **Review and Reporting Plan** describe the following activities:
  - Internal reviews.
  - External reviews.
  - Structured walk-through.
  - Reports.
- Avoid transforming project reviews in shows.
  - For them to be useful you must avoid the temptation to trumpet your successes and smother the problems — a difficult task no matter how objective and honest you think you are.

- One way to help dig out the problems is to include competent outside reviewers, that is, people who have no personal involvement in the project.
    - Reviews are discussed later in the text and a suggested set of reviews is included in templates.
- Written project reports, like other documents, have a way of growing more and more voluminous and less and less useful.
- The plan should lay out exactly:
    - What reports are *required*.
    - The reports *organization*.
    - The reports *frequency*.
    - *Responsibility* for writing the reports.
    - The reports *distribution*.
    - The reports *relation* to one another.

7.3.9 Installation and Operation Plan

- This describes the *procedure* for getting your finished, “accepted” program system installed and operating properly in its intended environment, perhaps at some missile defense site, perhaps in the computing center down the hall.
- The **outline** of this plan is presented in fig. 7.3.9.a.
- The plan contains to chapters:
    - (1) **Installation.**
    - (2) **Operation.**
- Even the simplest of programs can become *snarled* in such problems as how to convert from an existing, perhaps manual, system to the new, computerized system.

SECTION 9

INSTALLATION AND OPERATING PLAN (TEMPLATE)

DEPARTMENT:

PROJECT:

DOCUMENT NUMBER:

APPROVALS:

DATE OF ISSUE:

REALISED:

9.1. OBJECTIVE

The objective of this section is to define the contractor's responsibilities in installing and operating the accepted program system.

9.2. DISCUSSION

The amount of participation by a contractor in installing and operating a system he has delivered is a variable from one project to the next. Discussion subsection describe the degree of this involvement for your project.

9.3. DETAIL

9.3.1. Installation

9.3.1.1. Responsibility

9.3.1.2. Schedule

9.3.1.3. Conversion

(a) Method

Parallel operation, immediate replacement, etc.

(b) Cut-over criteria

How the decision is to be made to cut off the old system and relay on the new.

(c) Who makes cut-over decision

(d) Fallback position if system fails

9.3.1.4. Introduction of Data

(a) Who gathers data

(b) Who validates data

9.3.1.5. Multiple-Site Considerations

(a) Site installation team

(b) Site-to-site coordination

9.3.2. Operation

9.3.2.1. Responsibility for Operation

9.3.2.2. Responsibility for Maintenance and Tuning

(a) Change control procedures

(b) Work location

(c) Founding

9.3.2.3. Duration of Responsibilities

Fig. 7.3.9.a. Installation and Operation Plan template

7.3.10 Resources and Deliverables Plan.

- This plan element brings together in one place the *critical details* associated with your plan:
  - (1) Manpower.
  - (2) Computer time.
  - (3) Other resources.
  - (4) Delivery schedules - A summary of all items that you are to deliver under your contract.
  - (5) Milestone chart - A summary of project milestones.
  - (6) Budget.
- The **outline** of this plan is presented in fig. 7.3.10.a.
- These data are among the *most frequently changed or consulted*, so they should be gathered in one place to make them easier to find and easier to change.

SECTION 10

RESOURCES AND DELIVERABLES PLAN (TEMPLATE)

10.1. OBJECTIVE

The objective of this section is to gather in one place a summary of all resources estimates and a schedule for all a deliverable.

10.2. DISCUSSION

Various resources, schedules and deliverable items are mentioned or implied in other sections of the Project Plan. Here they are all tied together and made explicit. See Project Estimating Checklist Guide *and* Project Weighting Factors Guide.

10.3. DETAIL

10.3.1. Manpower

- *A **general chart** showing total manpower planned for the project on a monthly basis.*
- *A **main chart** should show two broad categories: programming and non programming manpower. Included in the first are programmers and the first-level managers; in the second are all other kind s of manpower.*
- ***Supporting charts** should break down the two categories into more detail.*
- *If the project is large and if there are a number of major program subsystems, show **separate manpower charts** for each*
- *If the project plans a number of releases of the complete program system, show manpower for each release separately.*

10.3.2. Computer time

- *Show monthly computer time requirements broken down by program release, by major program subsystem within release and by use category: module/integration test, system test, acceptance test, site test.*
- *If more than one type of computer installation is used (different platforms) show separate estimates.*
- *Show computer time separately for other categories, such as administrative uses.*

10.3.3. Other resources

10.3.3.1. Publication Costs

(a) Reports

(b) Problem Specification

(c) Design Specification

(d) Coding Specification

(e) User documents

(f) Test documents

10.3.3.2. Travel Costs

(a) To contractor's own facilities

(b) To customer facilities

(c) To other contractor's own facilities

(d) To test sites

10.3.3.3. Relocation of Employees and Equipment

10.3.3.4. Equipment and Supplies

The normal office equipment and supplies plus any special items (extra diskettes, hard disks,cd's etc.).

10.3.3.5. Special Purchases or Rentals

Such items as extra office space or temporary quarters in trailers.

10.3.4. Delivery Schedules

- *Chart(s) showing dates and deliverables called for in the contract or in any subsequent agreements.*
- *Accompanying the chart should be a set of narrative capsule descriptions of each item shown on the chart.*

10.3.5. Milestone Chart

- *A chart showing all milestone against which reports to the customer are to be made. A good base for this chart would be a variation of* The Model of Life Cycle*. It's helpful to show milestones overlaid on a development cycle, so that one can better relate each milestone to the planned major activities, that is the phases.*
- *Include a separate sheet giving a capsule description of each milestone indicated on the chart. See* Project Milestones Guide.

10.3.6. Budget

A copy of the financial budget showing how founds are allocated to each of the cost categories shown in the preceding sections. As estimates are reconsidered and changed, the budget must change. When that happens, this subsection must be updated to reflect the change.

Fig. 7.3.10.a. Resources and Deliverables Plan template

7.3.11 Plan Index

- It’s one of the most effective way to make your Project Plan much more attractive and usable to the reader.

8 Writing Acceptance Criteria

- Discussion of acceptance testing will be included under the **Acceptance Phase** in Chapter 8.
- But the actual work of preparing for **acceptance** begins here, in the Definition Phase.
- What deserves particular emphasis now is that the *criteria for acceptance must be* agreed to early and in writing.
- Don’t labor through an entire project without knowing exactly what **conditions** your product must satisfy in order to be acceptable to the customer.

Exercise #6

1. What are the *resources* of a SW project? What are the *cost elements* of a SW project?

2. Describe the most known *SW costs estimating techniques*: description, advantages, limitations.

3. What kind of *Decomposition SW Costs Evaluation Model* do you know? Describe their main characteristics.

4. What is *COCOMO 81*? Explain it's philosophy and the using manner: levels, inputs, processing, outputs, support.

5. What are the *enhancement* introduced by COCOMO II? Describe the main *stages* of COCOCMO II.

6. Describe the main characteristics of the PRICE S and SEER SEM parametric cost evaluation models.

7. What are the *steps of the estimation* of a SW Project?

8. What is a *project plan*? What are the *characteristics* of a good plan?

9. Describe the steps of the *Plan writing* activity.

10. What is the *structure* of a SW project Plan? Describe at detailed level the contents of the *Plan Sections*.